2019-11-11

Enabling Autonomous Operation of Micro Aerial Vehicles Through GPS to GPS-Denied Transitions

James Scott Jackson
Brigham Young University

Enabling Autonomous Operation of Micro Aerial Vehicles Through GPS to GPS-Denied Transitions

James Scott Jackson

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

ABSTRACT

Enabling Autonomous Operation of Micro Aerial Vehicles
Through GPS to GPS-Denied Transitions

James Scott Jackson
Department of Mechanical Engineering, BYU
Doctor of Philosophy

Micro aerial vehicles and other autonomous systems have the potential to truly transform life as we know it, however much of the potential of autonomous systems remains unrealized because reliable navigation is still an unsolved problem with significant challenges.

This dissertation presents solutions to many aspects of autonomous navigation. First, it presents ROSflight, a software and hardware architecture that allows for rapid prototyping and experimentation of autonomy algorithms on MAVs with lightweight, efficient flight control. Next, this dissertation presents improvements to the state-of-the-art in optimal control of quadrotors by utilizing the error-state formulation frequently utilized in state estimation. It is shown that performing optimal control directly over the error-state results in a vastly more computationally efficient system than competing methods while also dealing with the non-vector rotation components of the state in a principled way. In addition, real-time robust flight planning is considered with a method to navigate cluttered, potentially unknown scenarios with real-time obstacle avoidance.

Robust state estimation is a critical component to reliable operation, and this dissertation focuses on improving the robustness of visual-inertial state estimation in a filtering framework by extending the state-of-the-art to include better modeling and sensor fusion. Further, this dissertation takes concepts from the visual-inertial estimation community and applies it to tightly-coupled GNSS, visual-inertial state estimation. This method is shown to demonstrate significantly more reliable state estimation than visual-inertial or GNSS-inertial state estimation alone in a hardware experiment through a GNSS-GNSS denied transition flying under a building and back out into open sky.

Finally, this dissertation explores a novel method to combine measurements from multiple agents into a coherent map. Traditional approaches to this problem attempt to solve for the position of multiple agents at specific times in their trajectories. This dissertation instead attempts to solve this problem in a relative context, resulting in a much more robust approach that is able to handle much greater initial error than traditional approaches.

Keywords: GPS degradation, GPS denied, navigation, state estimation, observability, error state, sensor fusion, vision-aided INS, consistency, multirotor, micro air vehicle, indoor flight, outdoor flight, simultaneous localization and mapping (SLAM), pose graph optimization, obstacle avoidance, visual odometry, Moving Horizon Estimation, Pseudorange, Linear Quadratic Regulator, LQR, Moving Horizon Estimation, MHE, Sliding Window, Optimization

ACKNOWLEDGMENTS

Performing this research was a wonderful adventure. I am grateful for the opportunity I had to grow as an engineer, mathematician, writer and programmer. I am grateful for the chance I had to chase down interesting problems and the patience my advisors and lab mates had with me as I broke things and made mistakes, and I will never forget the incredible mentorship I received from my advisors and the more senior graduate students that gave me confidence to pursue some of my more ambitious ideas, and the other students who helped me make them happen. I am also deeply grateful to parents and my wonderful wife who supported me through these years, including the many late nights and long hours I spent working. Finally, I must acknowledge my Savior, Jesus Christ, who through His atonement has allowed me to repent of my many sins and allowed me to taste of His rightousness. All that I am I owe to Him.

Much of this research was supported by the NSF Center for Unmanned Aircraft Systems (C-UAS).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The world is quickly approaching a revolution that could potentially change the way we live our lives at a fundamental level. This revolution will likely create new jobs and opportunities that we can only dream of today, much like someone living in the 1960s could never have predicted our daily activities in the 2000s after the advent of personal computing.

The first computers were the experiments of university and government researchers. They originally cost millions of dollars and were used for military applications. Today, however, they are ubiquitous, with 77 percent of the United States population carrying a smartphone daily [1]. Recent advancements in autonomous micro aerial vehicles (MAVs) and other autonomous systems are driving a potentially similar revolution as autonomous systems enter the workplace, our transportation systems and our daily lives.

1.1 Micro Aerial Vehicles

MAVs are already being used in aerial photography, photogrammetry, surveillance, inspection and reconnaissance, however, they are still limited by the environments they can operate in reliably. The vast majority of MAV operations require clear GNSS reception, a controlled environment, and narrowly defined objectives. However, some studies speculate that autonomous MAVs could give rise to several billion-dollar markets in delivery, infrastructure monitoring, security, precision agriculture, and transportation [2]. Some of these potential markets, such as using MAVs to inspect bridges, dams, chemical plants, and refineries are particularly motivating as they would take the place of dangerous human inspections. Unfortunately, these markets are still unrealized because reliable autonomous MAV navigation is still an unsolved problem with significant challenges.

Before we can achieve the full potential of autonomous MAVs, they must not only be able to operate reliably and accurately in all kinds of environments, but they must also respond to unexpected scenarios and work together to accomplish larger missions than a single agent can alone. Specifically, we must improve our understanding of control, obstacle avoidance, state estimation, and mapping to the point that it is fast, reliable and robust.

This dissertation presents solutions to some aspect of all of these problems. In chapter 3, several shortcomings in the current methods for performing hardware experimentation that have prevented higher-level research objectives are addressed. The next two chapters present improvements on the real-time planning and control of quadrotors to enable more robust autonomy in unknown areas. The next two chapters focus on the problem of GPS-denied and GPS-degraded operations and finally, chapter 8 deals with improving the robustness of multi-agent map fusion and mapping with degraded GPS.

1.2 Real-Time Control of Micro Aerial Vehicles

Unlike ground-based autonomous systems, control of a MAV presents several unique challenges. The first is that MAVs are constrained by the size, weight and power available for processing of sensor information and planning. In some cases, heavy processing can be offloaded to a nearby ground station with wireless communication, but this introduces a point of failure in the system and jeopardizes reliability. Therefore, in designing algorithms for use on MAVs, computational efficiency becomes paramount. The second unique challenge posed by MAVs is their fast, unstable dynamics. A ground vehicle has the advantage that system dynamics are not only well constrained, but also typically much slower than a MAV. Furthermore, a ground vehicle can often remain at rest while waiting for heavy computation to complete, while this is impossible for a MAV.

A robust design for an autonomous MAV must take this into account, and real-time guarantees must be placed around the critical control and estimation loops. This becomes complicated when vision processing is involved, as the processors typically capable of performing these computations

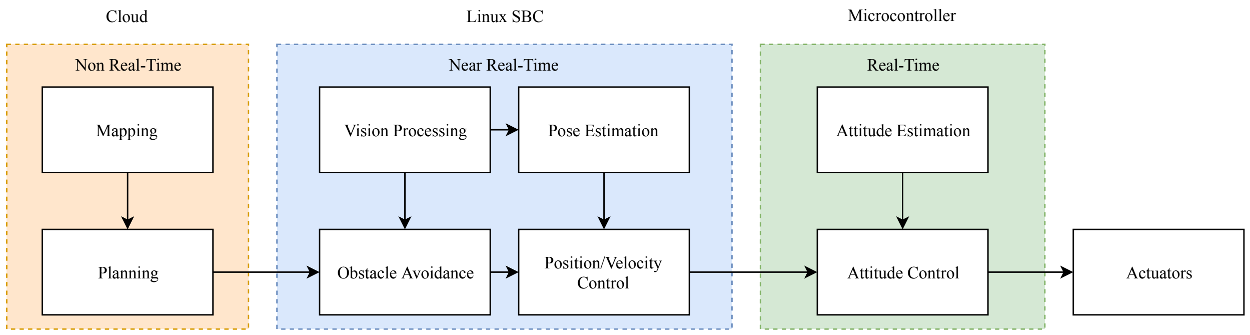


Figure 1.1: Three-tier architecture of MAV autonomy software and hardware

are not easily configured for running real-time software. One solution to this problem is to divide processes between real-time critical processes, near real-time processes, and non real-time processes while guaranteeing high-bandwidth, low-latency communication between the three groups. A potential configuration of this design is given in Figure 1.1

1.3 State Estimation of Micro Aerial Vehicles

Robust autonomy is only possible when built on a foundation of robust state estimation. Modern state estimation typically takes the form of a Bayesian inference problem. The objective is to find the most likely value of some state $\mathbf{x}$, which in the case of a multirotor, typically includes the position $\mathbf{p}$, attitude $\mathbf{q}$, and velocity $\mathbf{v}$, as well as the accelerometer and rate gyroscope biases, $\boldsymbol{\beta}_a$ and $\boldsymbol{\beta}_\omega$ respectively. We do this given sensor measurements, which we assume are related to our state through some measurement model $\mathbf{z} = h(\mathbf{x})$, and a state transition function $\mathbf{x}[t + \delta t] = f(\mathbf{x}[t], \mathbf{u}[t])$, where $\mathbf{u}$ are measured inputs.

There are several methods to accomplish this goal, the most common of which assume that both the state and the measurement are normally distributed random variables. Even if this assumption is not always exactly true, it makes real-time performance possible because we can leverage powerful linear algebra techniques to manipulate the covariance matrix associated with the normal PDF.

This maximum-likelihood estimation (MLE) can take place in several forms. The first attempts to solve for the most likely configuration of every state in the known history using all available information. This is the most powerful form of MLE, but it can also become computationally intractable as the number of measurements and states grow. The second method addresses this unbounded growth in complexity by limiting inference to some recent window of previous states which are simultaneously estimated from a group of measurements. This is known as sliding window, or moving horizon estimation (MHE). Finally, the most common probabilistic framework for state estimation is commonly known as the Extended Kalman Filter (EKF). The EKF can actually be viewed as a special case of MHE, with a sliding window of size one, and given the assumption that all involved random variables are distributed normally, inference can be performed using linear least-squares instead of the nonlinear least-squares techniques required when performing MLE more generally.

1.4 GNSS and GNSS Degradation

The earliest demonstrations of autonomous MAV operations relied on global satellite positioning and inertial sensor fusion (G-INS). GPS, Galileo, GLONASS and Beidou, (collectively known as global navigation satellite systems, or GNSS) are systems of satellites orbiting the earth between 20-30 km above the surface. These satellites are constantly reporting both a pseudo-random sequence with a known seed and their trajectory to the earth. A GNSS receiver is able to lock onto these signals and determine where in the pseudo-random sequence the satellite was when the signal was sent, and uses this information to determine precisely when the signal was transmitted. This measurement is known as the pseudorange, and with a pseudorange measurement to four or more satellites, an accurate measurement of time and position are possible with least-squares methods.

Most modern commercial, industrial and military use of MAVs is built on this technology, and therefore depends strongly on good GNSS signal reception. Unfortunately, many of the most useful potential applications of MAVs require operations in areas where GNSS cannot be guaranteed. GNSS-denied and GNSS-degraded state estimation is an active area of research, and this dissertation improves upon the state-of-the art methods in both GNSS-denied and GNSS-degraded applications.

As noted earlier, most of the autonomous operation of drones in commercial, industrial and military applications today rely heavily on a good GNSS signal. While this tremendous capability has been a boon for MAVs, relying too much on GPS has its disadvantages. GNSS-reliant systems are especially vulnerable to jamming, spoofing and multipath. In 2010 the United States Joint Chief of Staff, Norton Schwartz, stated,

> "It seems critical to me that the Joint Force should reduce its dependence on GPS-aided precision navigation and timing, allowing it to ultimately become less vulnerable, yet equally precise, and more resilient." [3]

While jamming and spoofing are typically due to adversarial conditions, multipath is a common phenomenon in normal operations. Multipath is when a signal from a GNSS satellite is reflected off a building, effectively extending the time offset the receiver observes from the satellite. If this erroneous measurement is fused naively then it can cause large jumps in position estimates as shown in Figure 1.2.

1.5 Vision-Aided Navigation of MAVs

In many cases, where GNSS is either unavailable or degraded, state estimation can be performed with a monocular camera and IMU. This sensor suite is also inexpensive and lightweight, but more importantly, it functions well indoors and near buildings where traditional GNSS estimation does not.

Visual-inertial navigation (V-INS) typically operates on the premise that stationary landmarks can be tracked over subsequent camera frames. The rotation and translation direction between frames up to a scale factor can be resolved from the imagery alone, and the scale factor can be

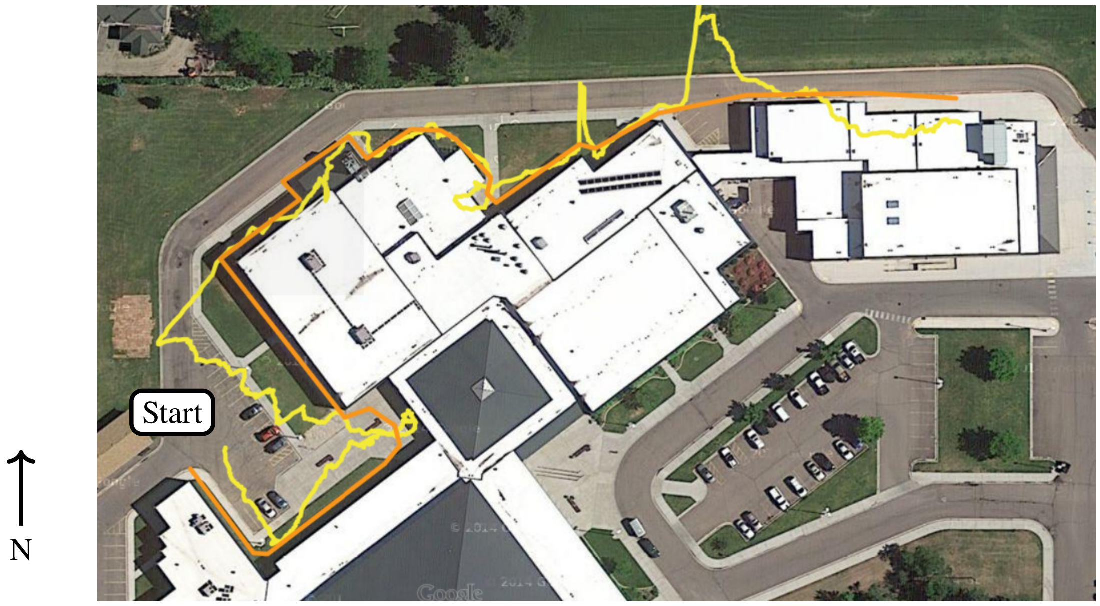


Figure 1.2: Comparison of true MAV trajectory (orange) with reported GPS measurements (yellow). At the start, the view of east-west GPS satellites are occluded by the building, leading to a significant multipath bias. Image first published in [4].

resolved by integrating inertial measurements. Unfortunately, because the accelerometer and gyro have an unknown, wandering bias, the monocular V-INS problem is stable only given sufficient excitation in all axes. Even with sufficient excitation, however, the global position and heading are still unobservable [5–7]. This leads to an unbounded error in global position and heading over time which also must be handled appropriately. Global position and heading unobservability can be mitigated to some extent with localization techniques such as loop closure. However this information is not always straight-forward to incorporate as will be discussed later.

Like G-INS, there are environments where V-INS does not perform well. The first, and most obvious, is when there is insufficient texture in the images to track features. This is often the case at high altitudes, over water, in the dark, or other feature-scarce environments. The other condition in which V-INS struggles is during significant lighting changes. This is because most cameras have some non-negligible delay in adjusting exposure levels to match lighting conditions. As a result, transitions from indoor to outdoor and vice versa can introduce a loss of all features being tracked in the image.

1.6 Pose Graph Optimization

Given the complementary strengths and weaknesses of V-INS and G-INS, it seems appropriate to leverage them together. However, they supply different kinds of information, which makes fusing them in a unified way difficult. GNSS measurements and localization events are inherently global, while visual and inertial information is relative. The EKF and MHE approaches are not well-suited to fusing both modalities of information. In contrast, however, a form of batch MLE, operating on a reduced set of information is well-posed to incorporate both global and relative information in a unified way [8, 9]. This is known as pose graph optimization.

As mentioned earlier, state estimation over our entire history can become computationally intractable if we try to perform inference over too much information. Therefore, we can marginalize out most of the information, reducing the pose graph problem to a set of *keyframes*. If we apply global information to these keyframes, and consider only the relative transforms between each one, then the problem becomes much more tractable.

Splitting the pose graph optimization from the real-time state estimation problem also allows us to leverage our three-tier architecture shown in Figure 1.1. Even with the reduced form of the pose graph problem, we have no guarantee that it can operate within real-time constraints. However, this does not pose a safety risk because the near-real time parts of the stack are able to operate indefinitely without feedback from the global map and planner.

1.7 Putting it all Together

The unrealized potential of MAVs hinges on each of the key ingredients just described. Working backwards through the architecture in Figure 1.1, we see that the low-level control of the MAV must be robust, and able to operate at real time. Next, we see that from a state-estimation perspective, the MAV must be able to perform both vision-aided and GNSS-aided navigation in near real-time, and from a control perspective, we must be able to not only execute control commands, but avoid unforeseen obstacles that may arise during operation. Finally, we must be able to robustly incorporate global information such as loop closures and GNSS measurements that can be used by our near real-time processes.

This dissertation seeks to solve these problems starting at the lowest levels and working up to the mapping problem, all with a focus on improving the robustness of MAVs in the GNSS-degraded and GNSS-denied zones. Increasing robustness in these areas will hopefully speed the realization of many of the exciting applications of MAVs in commercial, industrial and military applications. Nobody knows what the future holds, but history has shown us what potential a revolution like this could mean for human life. Fielding autonomous systems at scale could potentially free people from dangerous and menial tasks and enable them to work on more important issues, in fields that may have yet to be discovered, as we embark on a new chapter of human history.

CHAPTER 2: CONTRIBUTIONS

This dissertation describes three types of contributions. The first type are hardware and software architectural improvements to state-of-the-art. These contributions address several issues common to performing research on MAVs at the lowest levels of flight control by presenting a new hardware and software architecture used for rapid prototyping of autonomy algorithms. This work is an important contribution, as it lays the bedrock for experimental validation of the rest of the dissertation.

The second, and most significant kind of contribution focus on theoretial and algorithmic methods for improving robustness in autonomous operations in both control and estimation. Finally, all the work in this dissertation was ultimately demonstrated in hardware experimentation to validate the proposed theoretical contributions.

Much of the research has been published or is currently being prepared for publication and Chapters 3, 5, 6, 7 and 8 are presented in that form.

2.1 Real-Time Flight Control for Experimentation

Chapter 3

ROSflight: a Lightweight, Inexpensive MAV Research and Development Tool. *Jackson, Koch and McLain*. Currently being prepared for publication.

The development required to improve robustness of autonomous MAV operations in GNSS-degraded areas required a flexible platform for prototyping autonomy software. Originally, efforts relied on a number of autopilot options available, some of them supported by commercial entities and others supported by open-source organizations. However, integration efforts into these autopilots were frustrating. As many of these options had become more of push-button operations in open-air situations, they have become difficult to integrate with at the low levels required for high-performance autonomy research.

Therefore, to ease the research and development efforts required to perform the research presented in this dissertation, ROSflight was developed as an open-source flight controller with fundamental research needs at its core. This focus led to a design that includes:

1. Small, easy to understand code base.
2. High-bandwidth, low-latency communication with the flight controller.
3. Familiar, easy-to-use interface.
4. Robust software-in-the-loop (SIL) simulation capability.
5. Robust safety pilot integration.

The end result is a platform that enables the rapid development and testing of high-performance autonomy software at all levels of the autonomy stack. ROSflight was used in all hardware experimentations in this dissertation, and also as the base of most of the simulation experiments in a software-in-the-loop fashion.

Furthermore, ROSflight has now become an integral part of many projects currently going on at Brigham Young University, was used to perform novel research on the model-predictive control of multirotors at Luleå University of Technology [10], undergraduate engineering projects at the University of California, Berkley, and graduate research pending publication at the Massachusetts Institute of Technology and Texas A&M University. Several simulation environments are currently leveraging the powerful SIL capability of ROSflight in Gazebo [11] and the Unreal 4 [12] and Unity video game engines.

ROSflight is a small but growing open-source community. The website `www.rosflight.org` currently sees approximately 240 unique users per month, with users from all over the world.

Chapter 4

Optimal Control of a Multirotor Using An Error-State Formulation.

Optimal control of multirotors has been the focus of a lot of research effort because it has the potential of maximizing performance from the multirotor platform. Most optimal control strategies that have been presented rely on Euler decomposition of the attitude state of the quadrotor, which leads to linearization errors and inefficiencies in computing control outputs. Chapter 4 explores the use of the Lie group theory to perform optimization on the rotation manifold directly. This method avoids Euler decompositions and therefore results in both smaller linearization errors and much more efficient computation. Chapter 4 presents both on-manifold LQR and MPC of a quadrotor, and presents simulation results of both methods. The work presented in this section was further explored by Farrell et. al [13] where it was demonstrated in hardware experimentation and shown to be much more efficient than other state-of-the-art methods.

Chapter 5

Cushioned Extended-Periphery Avoidance: a Reactive Obstacle Avoidance Plugin. *Jackson, Wheeler, and McLain*. Published at the International Conference on Unmanned Aircraft Systems (ICUAS) in 2016.

As noted in Figure 1.1, obstacle avoidance and flight stability must be performed in a near real-time fashion. Many existing planners focus on planning optimal paths through a static, known environment, and can often take significant time to replan paths if changes in the environment are discovered. However, before we can fully realize the usefulness of MAVs, they must be able to quickly react to unforeseen disturbances and unknown changes to the environment. Chapter 5 first introduces the reactive obstacle avoidance plugin (ROAP) framework as a method for leveraging map-based algorithms while providing low-latency, high-bandwidth response to obstacles. Further, we propose and demonstrate the effectiveness of the cushioned extended-periphery avoidance (CEPA) algorithm.

CEPA uses a scanning laser to provide a fast, effective method for reactive obstacle avoidance, and makes use of two main ideas. First, to improve efficiency, planning and mapping are performed directly in a polar coordinate frame to avoid transforming the laser scan returns into a Cartesian coordinate system. Second, CEPA builds a 360 degree map of the local environment by combining recent laser scans. This extends the field-of-view of the avoidance algorithm, which improves robustness. In Chapter 5, CEPA is demonstrated in simulation, and on hardware in a GPS-denied environment using stricly onboard computation and sensing.

2.2 State Estimation in GNSS-Degraded/GNSS-Denied Environments

Chapter 6

Improving the Robustness of Visual-Inertial Extended Kalman filtering. *Jackson, Nielsen, McLain, and Beard*. Published at the International Conference of Robotics and Automation (ICRA) in 2019.

Before MAVs can be deployed in autonomous missions in areas where GNSS measurements may not be reliable, we must develop robust methods of navigating with lightweight, low-cost sensors. Visual-inertial (VI) navigation methods have shown promise in this area, however, even state-of-the-art VI approaches suffer from observability and consistency issues. Filtering approaches have the advantage of being computationally efficient, however, if a Kalman filter becomes inconsistent, then it can no longer fuse information optimally and accuracy degrades.

Monocular visual-inertial filtering approaches also must maintin a sensitive balance between accelerometer bias estimation, velocity estimation, and the estimation of depth to features. In many cases, both the accelerometer biases and depth to features are only partially observable, which leads to error in the velocity estimates that are commonly used for feedback control. Avoiding or reducing sensitivity to these issues are key to robust operation of autonomous MAVs.

Chapter 6 modifies a state-of-the art VI Kalman Filtering approach to improve robustness when used on multirotor agents. First, a linear drag term is added to the velocity dynamics. This is shown to improve estimation accuracy, but comes at the cost of estimator consistency. Second, a partial-update formulation is used to limit the effect of linearization errors in partially-observable states, such as the aforementioned drag term, depth to features and accelerometer biases. The partial update is shown to restore consistency to the filter and limits the effect of linearization errors in the partially observable states. Finally, a keyframe reset step is added, which enforces consistency and observability of the normally unobservable position states. This further improves consistency and accuracy, especially in the position and heading states.

These modifications are first derived, then implemented and Monte Carlo simulation experiments are performed to demonstrate the effectiveness of the additions. The combination of all three modifications is shown to significantly improve both accuracy and consistency.

Chapter 7

GV-INS: Fusing GNSS, Visual, and Inertial Sensors in a Moving-Horizon Estimation Framework. *Jackson, McLain, and Nielsen.* To be submitted to The International Journal of Robotics Research.

While VI methods have shown a lot of promise in indoor environments, they are not robust to significant lighting changes or texture-less scenes and they cannot reliably resolve global position and heading without additional information. GNSS-based state estimation, on the other hand, easily resolves global information so long as reliable measurements are used. Both methods struggle in the GNSS-degraded zone, where visual information may not be available, and GNSS suffers from multipath measurements.

Furthermore, work in visual-inertial (VI) state estimation in particular has driven significant advancements in the tools and understanding of state estimation in general, however, many of these advancements have been focused solely on visual applications and not applied to other sensing modalities. GNSS-inertial (GI) fusion has been largely left alone by some of these advancements, but could significantly benefit from many of the ideas developed by the VI community.

Chapter 7, takes the ideas of moving-horizon estimation (MHE), IMU preintegration and robust optimization with switching parameters from the VI community and applies it to fusing raw GNSS signals, inertial measurements, and visual information simultaneously in real-time onboard a MAV. The resulting GV-INS system demonstrates that by using all three modalities, GNSS multipath can be reliably detected and rejected, which allows for reliable operations in and around buildings. Chapter 7 also shows how to fuse both global and relative information in a unified way in an MHE framework. GV-INS is demonstrated both in simulation and in hardware and is shown to be superior to VI and GI alone.

Chapter 7 also provides a clear derivation of factor graphs, the IMU preintegration factor, and moving horizon estimation in terms of the notion of error states. The error state is a concept commonly utilized by more established Kalman filter literature, but it is typically not referenced by more generalized Bayesian inference such as MHE and batch MLE. The derivation in Chapter 7 provides a bridge between these modern state estimation methods and more established error-state Kalman filter literature.

2.3 Robust Pose Graph Optimization

Chapter 8

Direct Relative Edge Optimization, a Robust Alternative for Pose Graph Optimization. *Jackson, Brink, Forsgren, Wheeler, and McLain.* published in Robotics and Automation Letters in Jan, 2019 and was presented at at the International Conference of Robotics and Automation in May 2019.

As MAVs operate in larger environments, over larger timescales and potentially in collaborative groups, fusing information at scale becomes a difficult problem. Pose graph optimization is an effective method for fusing large-scale information, however it suffers from a lack of robustness to large initialization errors. Because pose graph optimization methods typically linearize about the current pose estimate for the entire map, if the map is incorrectly initialized, this linearization can be performed arbitrarily far from the truth and result in divergent behavior. Furthermore, there is no obvious method of recovery, which makes autonomous operations difficult in this situation.

Chapter 8 re-parameterizes the classic pose-graph problem into a relative context and shows how the pose graph problem is better conditioned in the relative parameterization, and therefore more robust to initialization errors. This work is distinct from other work that also identified the relative parameterization by showing how to perform optimization over the relative constraints in an entire pose graph, instead of some subset of simply-connected edges, which was all that was possible in previous work. The relative parameterization is shown to be significantly more robust to initialization errors in both simulation and hardware experiments, and therefore a strong candidate for real-life situations where initialization errors may cause unacceptable performance in the traditional solution.

2.4 Summary

In summary, this dissertation describes the following contributions:

- Develops a method to use relative parameterization of non-trivial pose graph optimization to perform robust optimization. This was previously impossible on graphs with cycles, and significantly improves robustness of pose graph optimization to real-world initialization errors.
- Describes the first-ever tightly-coupled GV-INS where GNSS, visual, and inertial information are fused in a unified moving-horizon estimation framework. This framework is shown to

be able to operate reliably in GNSS-degraded environments, which is a significant problem area for current methods.

- Derives optimal control of multirotor agents in an error-state formulation, and describes a principled way to perform on-manifold control. Because of the numerical efficiencies of optimizing directly on-manifold, the resulting LQR controller was subsequently demonstrated in [13] to require about 1/50th the computation of the current state-of-the art LQR [14]. Further, as the described MPC controller is shown to require only twice as much computation time as the LQR formulation, it is likely that hardware demonstrations of the MPC formulation could resulted in a computational cost of only 1/25th the aforementioned state-of-the-art LQR.
- Details the hardware and software architecture that has been used for rapid prototyping and experimentation of advanced autonomy applications on MAVs not only in this dissertation, but in organizations across the globe.

2.5 Additional Publications

Besides the work included in this dissertation, I was also involved in several other publications and am listed as a contributing author. These are listed here for reference:

- Jackson, James and Ellingson, Gary and McLain, Tim "ROSflight: A lightweight, inexpensive MAV research and development tool", ICAUS 2016 [15]
- Wheeler, David O. and Koch, Daniel P. and Jackson, James S. and McLain, Timothy W. and Beard, Randal W. "Relative Navigation: A keyframe-based approach for GPS-degraded navigation," IEEE Control Systems Magazine, Aug 2018 [16]
- Wheeler, David and Koch, Daniel and Jackson, James and Ellingson, Gary and Nyholm, Paul and Mclain, Tim and Beard, Randal, "Relative Navigation of Autonomous GPS-Degraded Micro Air Vehicles", Currently being prepared for publication [17]

- Farrell, Michael and Jackson, James and Nielsen, Jerel and Bidstrup, Craig and McLain, Timothy, "Error-State LQR Control of a Multirotor UAV", ICUAS 2019 [13]
- Nielsen, Jerel, and Jackson, James, and Beard, Randal, and McLain, Timothy, "A Visual-Inertial Extended Kalman Filter Using Image Coordinates," Currently being prepared for publication.

CHAPTER 3: ROSFLIGHT: A LIGHTWEIGHT, INEXPENSIVE MAV RESEARCH AND DEVELOPMENT TOOL[1]

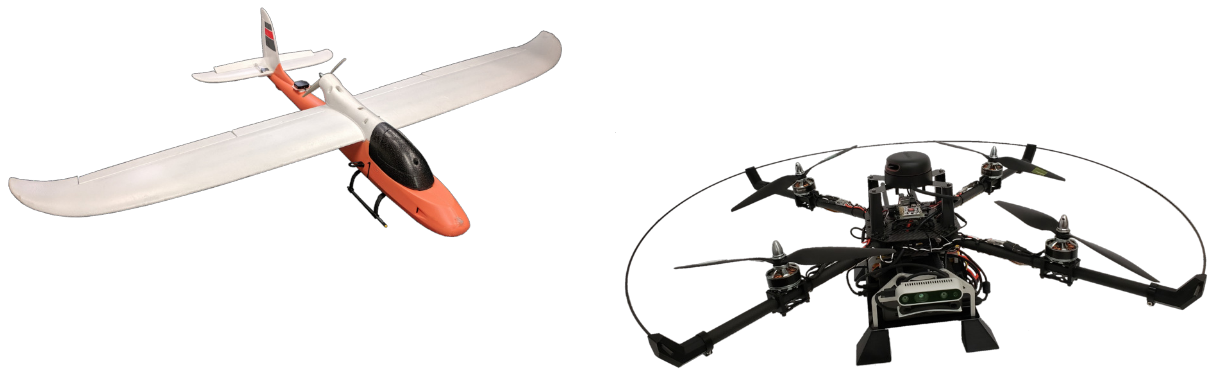


Figure 3.1: Quadcopter and Fixedwing MAVs using the ROSflight flight controller

3.1 Introduction

In recent years, we have seen a tremendous amount of research in the autonomous operation of micro aerial vehicles (MAVs). MAVs have been demonstrated in disaster response, inspection, monitoring, mapping and in other support capacities. Many of these recent advancements have relied on original research performed by academic institutions, government organizations and other research organizations inside of commercial entities. To aid these organizations in rapidly developing basic MAV technology, we present ROSflight as a basic, lean, open-source autopilot that has been developed specifically with the needs of researchers in mind.

Along with the advancement in MAV technology, there have been several autopilot technologies made available to researchers. Some of these have been developed and maintained by commercial entities [18–23] while others have been developed by communities of volunteers [24–27]. Most of

[1]This paper was was written by James Jackson and Dan Koch and Tim McLain and is being prepared for publication.

these autopilots are designed to enable push-button operation of MAVs in open-air situations with GPS, or under tight control from a human pilot. As a result of both the intense competition and the incredible energy in the autopilot space, many of these autopilots have become quite feature-rich and demonstrate state-of-the-art capabilities in terms of autonomy or real-time performance.

Unfortunately, many of these autopilots have feature-creeped their way out of being useful to researchers. Many researchers of MAV technology are working on improving low-level state estimation or control algorithms [10, 13]. Others require a powerful processor for vision processing for GPS-denied autonomy [5, 17, 28] Open-source autopilots such as [25, 27] have become so feature rich that it takes an unreasonable amount of time for a researcher to fully understand the implications of changing aspects of the way the code works, and how to properly integrate their research code into the autopilot, from both the physical systems integration and the control systems perspective.

Researchers often require specific interfaces to the MAV. For example, a researcher in model predictive control (MPC) of a quadrotor might wish to supply angular rate commands at high rate while compensating for throttle response, while another researcher might wish to supply attitude commands in roll in pitch and an angular rate command in heading. In general, most researchers want to maximize control and estimation bandwidth with the MAV, which means prioritizing offboard control and sensor streaming at the lowest levels. Exposing these interfaces in a push-button autopilot can not only be difficult, but even dangerous, so they are generally not made available in more fully-featured autopilots.

To ease autopilot research and development efforts, we propose ROSflight. ROSflight is a flight controller designed with researcher needs as the primary focus. This has led the the following aspects of the autopilot design:

1. Small, easy-to-understand code base.

2. High-bandwidth, low-latency communication with the flight controller.
3. Familiar, easy-to-use interface.
4. Robust Software-In-The-Loop (SIL) simulation capability.
5. Robust safety pilot integration

We believe that these features are critical to a flexible and powerful autopilot platform to be used in a variety of research projects and have found ROSflight to be useful in our work, and in the work of others.

The rest of the paper is organized as follows: First, we will describe the overall vision, intended use case and organization of the ROSflight project. Next, we will describe in detail the design of the autopilot, including each of the parts and their communication. Third, we will discuss integration of the autopilot for both hardware experimentation and simulation and finally, we will discuss a few research projects that have used ROSflight to successfully complete and publish novel research.

3.2 Overview

In this section we describe the vision and long-term goals for the ROSflight project, then provide an overview of how we envision the system typically being used. We also provide a brief overview of the organization of the ROSflight project.

One of the primary goals of the project is that the code base will remain lean and easy to understand. In our experience, complex black-box systems have not been conducive to research activities because they make it difficult to debug the full-system behavior when the details of the inner-loop operation are not well-understood. In addition, highly complex systems can be difficult to configure for the unique requirements of research applications, and can be difficult to modify when required.

To avoid these pitfalls, we have chosen to adopt the philosophy that the embedded ROSflight firmware will implement only the minimum functionality required to achieve safe and stable flight. This functionality includes sensor and actuator input/output, high-bandwidth communication with a companion computer, attitude or attitude-rate control when operating a multirotor aircraft, and supporting functionality such as configuration management and safety features. All higher-level functionality is left for the user to implement, typically on a companion Linux computer. While this requires effort from the user, we believe that it makes ROSflight a much more flexible and easy-to-use tool for researchers who often write highly-customized application code. We have also chosen to adopt a hard stance against feature creep; we welcome users who wish to incorporate additional functionality into the embedded flight controller to fork our project and to share their successes, but in general we will not merge these extensions into the core code base.

Another key goal of the project is to enable useful simulation capabilities, including true software-in-the-loop (SIL) simulation. Some autopilot systems use flags in the firmware to change the behavior of the code when running in SIL mode. The ROSflight firmware instead uses a hardware abstraction layer to implement SIL, so that the core flight-stack code that runs in SIL is identical to the code that runs in hardware, and has no knowledge of which mode it is running in. This approach also allows ROSflight be be incorporated into a variety of simulation environments. Additionally, the interface with application code is identical between simulation and hardware, which allows many integration issues to be debugged in simulation before moving to hardware. These features are discussed in more depth in Section 3.4.

3.2.1 Typical Use Case

The intended use case for ROSflight is illustrated by the diagram in Figure 3.2. There are three main components to the system: the embedded flight controller, the companion computer, and the safety pilot.

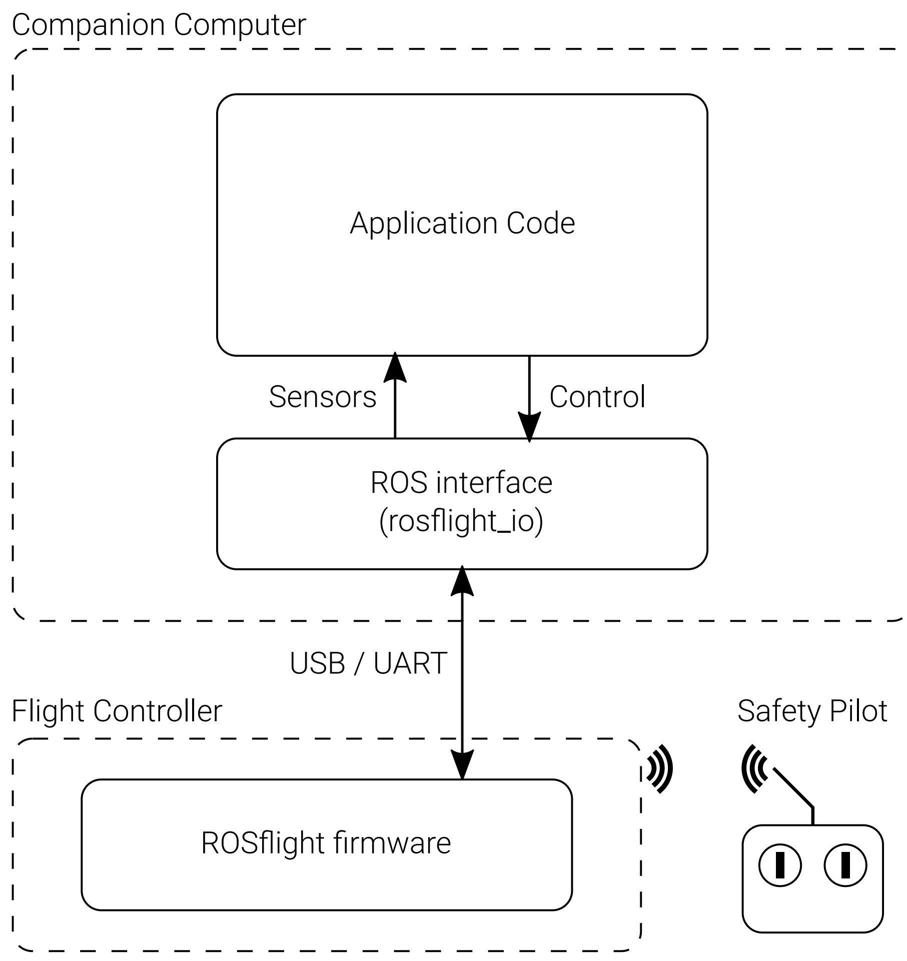


Figure 3.2: Intended use case. The embedded flight controller provides sensor and actuator input/output. The application code runs on a companion computer, and communicates with the flight controller through a provided ROS interface. A safety pilot is able to override computer control if needed.

The *flight controller* runs the ROSflight firmware,[2] and provides low-level input/output for sensors and actuators (servos and electronic speed controllers (ESCs)). For multirotor vehicles, the flight controller also performs attitude or attitude-rate control. For most research applications, the embedded firmware should not need to be modified, unless new low-level multirotor attitude or attitude-rate controllers are being developed.

The *companion computer* is a Linux computer—such as an Intel NUC, NVIDIA Jetson, Odroid, or other small-form-factor computer—that is mounted on the vehicle and is connected to the flight controller via USB. Connections over UART serial are also supported if required. For researchers using the Robot Operating System (ROS)[3], the `rosflight_io` node in the `rosflight` package[4] provides a ROS interface to communicate with the flight controller. All configuration of the flight controller is also performed via service calls provided by `rosflight_io`.

It is intended that most research code will run on the companion computer, as indicated by the *application code* block in Figure 3.2. The application code can use the high-rate sensor data streams exposed by `rosflight_io`, and sends control setpoints to the flight controller. These control setpoints can consist of attitude and throttle commands, attitude rate and throttle commands, or direct throttle and servo commands. We refer to these setpoints as *offboard control* setpoints, because they are "offboard" from the perspective of the flight controller (even though the companion computer is also mounted on the vehicle).

The *safety pilot* is an integral part of the system, and interacts with the flight controller using a standard radio control (RC) transmitter. Due to the nature of research code, it is important that the safety pilot have the ability to quickly override the offboard control setpoints at any time. The flight-controller firmware makes three mechanisms available to accomplish this:

[2]`https://github.com/rosflight/firmware`
[3]`http://www.ros.org/`
[4]`http://wiki.ros.org/rosflight`, `https://github.com/rosflight/rosflight`

1. The safety pilot may lock out the offboard setpoints completely by flipping a switch on the transmitter,
2. The flight controller will follow the minimum of the throttle setpoints coming from the safety pilot and offboard setpoints, allowing the safety pilot to quickly kill the throttle if necessary,
3. The safety pilot may temporarily and independently override the roll, pitch, or yaw-rate channels by deviating the corresponding transmitter stick from center.

These override mechanisms have proven to be valuable in our experience, although they may be independently disabled if desired. For safety reasons, the flight controller can only be armed from the safety pilot's RC transmitter.

3.2.2 Project Organization

The ROSflight project is hosted on GitHub[5], and consists of two separate but related code bases: the embedded flight-controller firmware (`https://github.com/rosfligh/firmware`), and the ROS interface (`https://github.com/rosflight/rosflight`). Documentation is maintained as part of the project. Links to the code repositories, documentation, and other resources are provided at our website, `http://rosflight.org/`.

3.3 Design

The core ROSflight firmware flight stack is made up of several modules. This design is intended to limit the scope of each module so that the implications of changes to any one module can be easily understood. The module-level architecture of the flight stack is shown in Figure 3.3. In this section, we will describe each module, its role in the flight stack and the relevant algorithms used in its runtime processing.

[5]`https://github.com/rosflight`

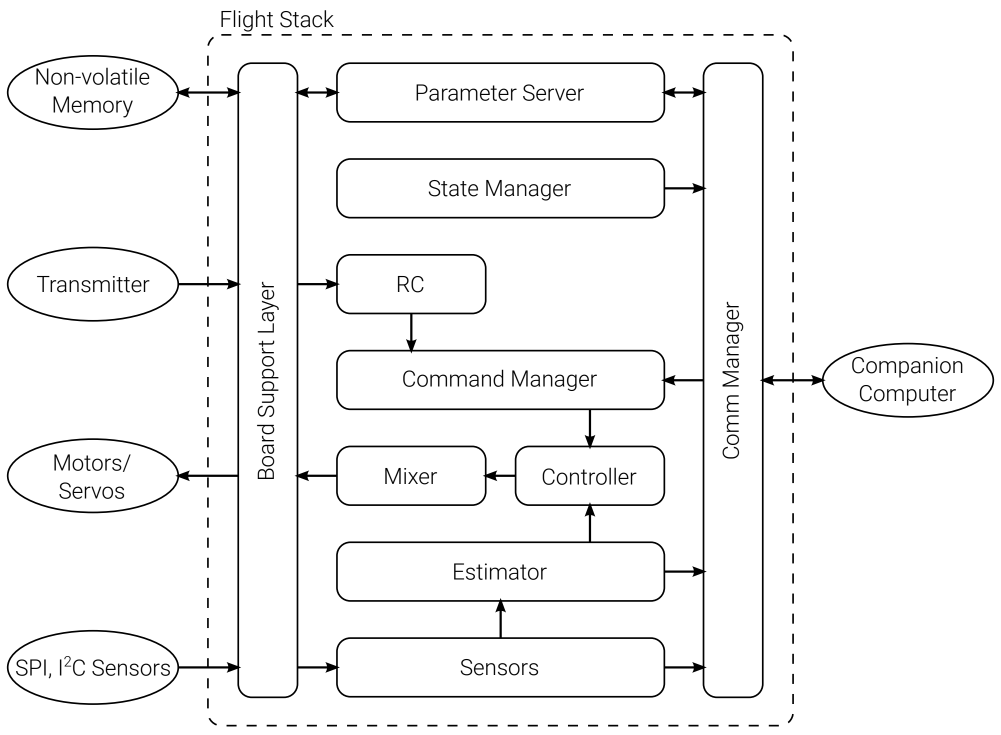


Figure 3.3: Flight stack architecture

3.3.1 Flight Management Modules

The first three modules we will discuss are the state manager, the parameter server and the communications manager. These modules are responsible for providing an efficient and safe environment for the control and state estimation to occur. Because of their role in managing the operating environment, they tend to have larger scope than the other modules, and can be a little less straight-forward in implementation. However, significant attempts have been made to limit their complexity while still maintaining an efficient implementation.

State Manager The state manager is responsible for handling external events and system errors, and managing the system state. "State" in this context refers to whether the system is experiencing errors, if the system is armed, or in a failsafe condition, as opposed to the current attitude and angular rate of the MAV, as will be discussed later. The state manager also informs all the other

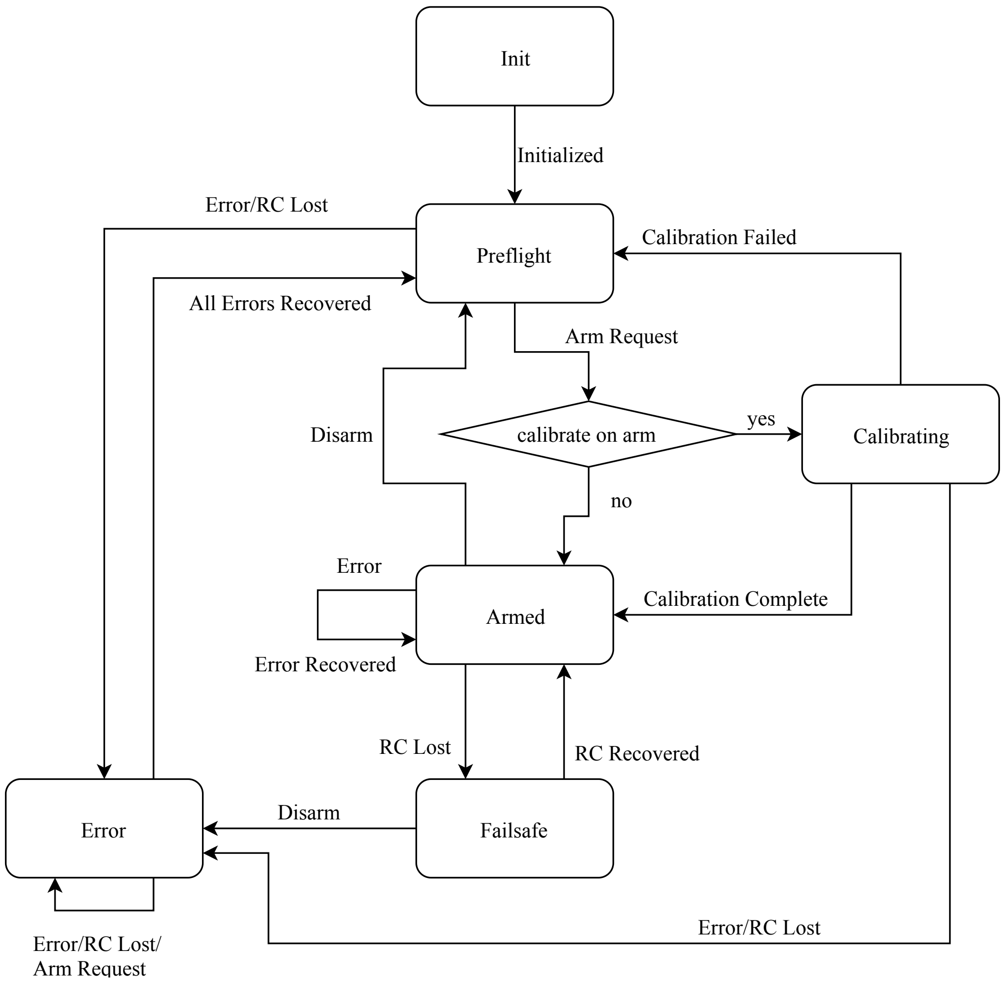


Figure 3.4: State Machine Diagram

modules of the current state of the flight controller so they can behave appropriately. There are six discrete states and seven transition events. These states and their associated transitions are diagrammed in Figure 3.4

Parameter Server The flight controller behavior is controlled by a number of parameters that are configurable by the user while in the setup phase of flight, but remain constant during operation. These parameters include controller gains, motor and remote control configuration, and sensor stream rates. Parameters are accessible to all modules in the flight stack at any time, however, the parameter server includes a mechanism that informs all the other modules of changes to each parameter to improve efficiency during runtime. The parameter server is also responsible for informing the user of the current value of any requested parameter via the communications manager.

As a convenience, parameters are also saved to non-volatile memory so they persist through reboots. However, to avoid loading either a corrupt or an out-of-date parameter set, a 32-bit checksum, the length of the parameter set, and a hash of the firmware code version history are saved with the parameter file. Upon loading the file, these values are checked against pre-compiled constants and if they are found to be out-of-sync, a default parameter set is loaded instead, and an associated warning is displayed to the user.

Communications Manager The communications manager is responsible for receiving commands from the companion computer and streaming sensor data. While the actual communication protocol is abstracted to enable easy replacement, the current implementation uses MAVlink as the serial protocol [29]. An assocated MAVlink parser is supplied for the companion computer to decode and encode the sensor information and commands.

Communications between the computer and the flight controller take one of three forms. The first is streaming information from the flight controller to the companion computer, such as sensor information, or the current motor commands. This information is simply streamed to the companion computer at some specified rate, and makes up the bulk of communication between the two devices. The second kind of information are commands from the companion computer to the flight controller. Some of these commands include commands to calibrate certain sensors, change a parameter, or reboot the processor, and require an acknowledgement from the flight controller that the command has been received and executed properly. Finally, there are streaming commands from the computer to the flight controller. These include external attitude updates and control commands, such as the desired roll, pitch, yaw-rate and throttle values from the companion computer. These commands are streamed to the flight controller at some specified rate, and do not require an acknowledgement.

All three of these kinds of commands can take place simultaneously in this system, and emphasis has been put into enabling very high streaming rates, including streaming IMU sensor measurements at more than 1000 Hz. To ensure that streaming messages are prioritized over other communications during flight, many acknowledgement-type commands are disabled while the aircraft is in the armed state. This is also a safety feature of the autopilot, as many of these commands could have potentially disasterous conseqences in flight, such as changing the motor mixer type or rebooting the processor.

3.3.2 Flight Control Modules

The rest of this section will focus on the rest of the modules shown in Figure 3.3. These modules are primarily responsible for the real-time state estimation and control of the aircraft. At a high-level, control commands are received from both the radio control (RC) transmitter and the companion computer. These commands are merged together by the command manager, taking into account safety pilot integration and other user-specified constraints, and given to the controller. Meanwhile, sensor information is collected from the board abstraction layer and safety pilot integration is provided to the state estimator, which provides the current state estimate to the controller as well. The controller determines the desired forces and torques in the body frame, and the mixer takes these body-frame commands and transforms them into individual motor commands, based on the geometry of the MAV. These commands are passed back down through the board layer to the actuators on the aircraft.

Command Manager The command manager is responsible for combining commands from the companion computer and the remote control operator, enabling safety pilot integration and safe testing of unproven control and estimation algorithms. The resulting command from the command manager may be solely from the RC receiver, the companion computer, or a mixture of both (see Figure 3.5).

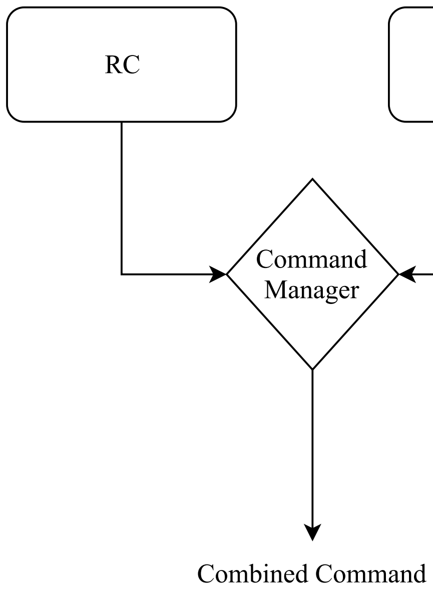


Figure 3.5: The combined command from the command manager is a mixture of the command from the RC and the companion computer.

By default, if both RC and companion computer commands are present, the companion computer command is chosen as the combined output. However, if the RC sticks are deviated or a lockout switch on the RC transmitter is activated, then the RC safety pilot command is chosen instead. In the case that the companion computer supplies no command, or no new command has arrived within a chosen time window, then the combined command defaults to RC. Finally, if neither RC or the companion computer supply commands, then a pre-determined failsafe command is executed until RC communication is recovered. This logic is repeated across the four input channels, which correspond to roll, pitch, yaw and throttle, and are treated independently during merging. This means that a safety pilot can override just one of these channels, while keeping the other channels under control of the companion computer. This is useful for artificially causing disturbances to some higher-level controller, making minor corrections during flight, or very quickly recovering control in the case of dangerous commands from the companion computer. The override switch affects all channels, and prevents execution of any companion computer commands. This is also helpful in the case that a prototype controller running on the companion computer does not perform as expected and needs to be disabled immediately.

Because poor controller response in throttle can be especially dangerous, the command manager incorporates extra functionality on the throttle channel. Beyond the normal mixing that also occurs in the roll, pitch and yaw channels, the throttle channel can be put into a mode that compares the desired throttle value from the RC and companion computer command, and takes the lower of these two values. This throttle saturation override provides to practical uses. First, it allows a safety pilot to limit the amount of throttle available to the companion computer and ease this limit up as they become more confident in the performance of a prototype control algorithm. Secondly, it also makes recovery much more graceful, because otherwise, the safety pilot much either try to match the throttle output of the companion computer before switching, or switch to some arbitrary throttle value without knowing how the MAV will respond.

Estimator The state estimator is responsible for using the accelerometer and rate gyroscope to estimate the current attitude, angular rate, and rate gyroscope biases of the MAV. ROSflight implements the passive unit quaternion filter from [30] with some modifications from [31].

Without external position or velocity measurements, attitude is only observable under unaccelerated conditions and the yaw-rate gyroscope bias is completely unobservable. Therefore, we employ an external attitude update mechanism that allows the companion computer to supply another estimate of attitude to the estimator, which gets fused with the accelerometer and gyroscope. If this measurement is accurate, then ROSflight performs much better in accelerated conditions, and the yaw-rate gyroscope bias can converge. A complete description of the algorithm used to perform state estimation is detailed in Appendix 3.6

Controller The controller is responsible for driving the MAV to its desired state. It consumes both the state estimate and the combined command and produces normalized desired torques and forces. The controller can be configured to operate in one of three modes: angle mode, rate mode, or passthrough mode. All relevant control loops use a PID-like structure as shown in Figure 3.6.

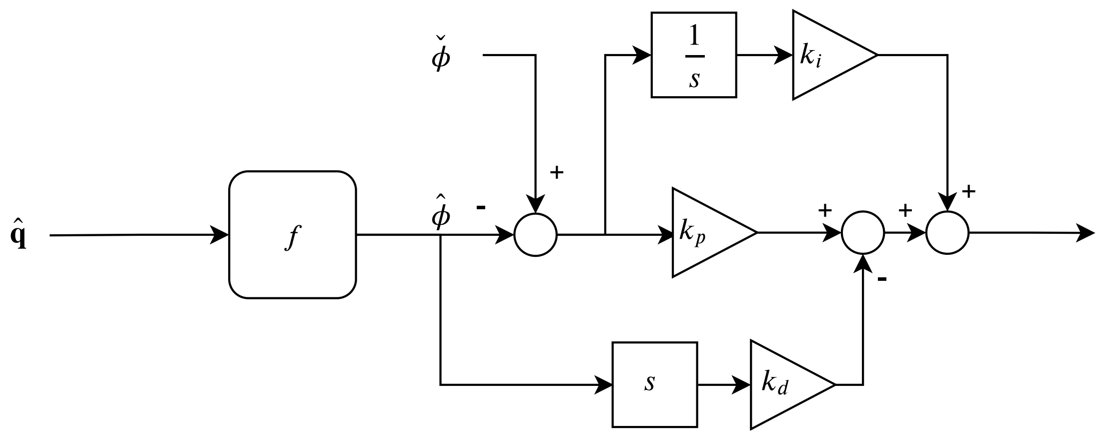


Figure 3.6: Block Diagram for PID controller design.

In angle mode, the user supplies the desired roll and pitch angle, the desired yaw rate and desired throttle values. Because the controller operates directly on the Euler angles, the current roll and pitch angles are first extracted from the current estimated attitude before being consumed by the PID control loops. Due to the Euler decomposition step, commanding extreme pitch angles in angle mode is not advised, because the Euler decomposition will encounter a singularity at 90 degrees pitch. Rate mode control does not have this problem, but it is considerably more difficult to operate from a safety-pilot perspective. The block diagram for the attitude-mode controller is shown in Figure 3.6.

In rate mode, the user supplies values of desired angular rate in all three axis and throttle. The angular rate control loops have the same structure as the attitude controllers besides for the Euler decomposition step at the beginning, however they use a separate set of gains with the k_i and the k_d gains typically set to zero.

In passthrough mode, the controller is completely bypassed, allowing direct access to the motor mixer. This allows for direct actuator control from the companion computer or RC if required, and is a common use case when using fixed-wing aircraft that have slower dynamics than multirotors.

It should be noted that the RC and the companion computer can supply angle, rate or passthrough commands, and that they do not have to match. That is, the safety pilot can be commanding angle-mode commands while the companion computer is operating in rate or passthrough mode.

Mixer The mixer takes the normalized torque and throttle outputs from the controller and maps them into individual actuator commands, depending on the location and type of actuator. For example, a quadcopter "+" configuration requires different motor actuation from a quadcopter "×" configuration to achieve the same torques. The motor mixing occurs by multiplying a static matrix $A \in \mathbb{R}^{n\times 4}$ where n is the number of actuators by the normalized force and torques such that

$$\mathbf{u} = A\boldsymbol{\tau},$$

where $\boldsymbol{\tau}$ is the column vector of normalized forces and throttle, and $\mathbf{u}$ is the actuator command to each actuator, between -1 and 1 for servos, and between 0 and 1 for motors.

Because both $\boldsymbol{\tau}$ and $\mathbf{u}$ are normalized, each row of A is also typically normalized, and can be calculated for a multirotor MAV as follows: If $\mathbf{r}_i$ is the vector from the center of mass of the MAV to center of the propeller plane, $\mathbf{e}_i$ is the unit vector in the direction of thrust, and s_i is the direction of propeller rotation along $\mathbf{e}_i$ (following the right-hand rule), then

$$\begin{aligned} M &= \begin{bmatrix} \mathbf{k}^\top \mathbf{e}_0 & \dots & \mathbf{k}^\top \mathbf{e}_n \\ \mathbf{r}_0 \times \mathbf{e}_0 + \mathbf{k}s_i & \dots & \mathbf{r}_0 \times \mathbf{e}_0 + \mathbf{k}s_i \end{bmatrix} \\ A &= \|M^\dagger\|_{\text{row}} \end{aligned}$$

where $\mathbf{k}$ is the unit vector in the z direction, $(\cdot)^\dagger$ is the Moore-Penrose pseudo-inverse, and $\|\cdot\|_{\text{row}}$ is the matrix resulting from normalizing every row of the supplied matrix.

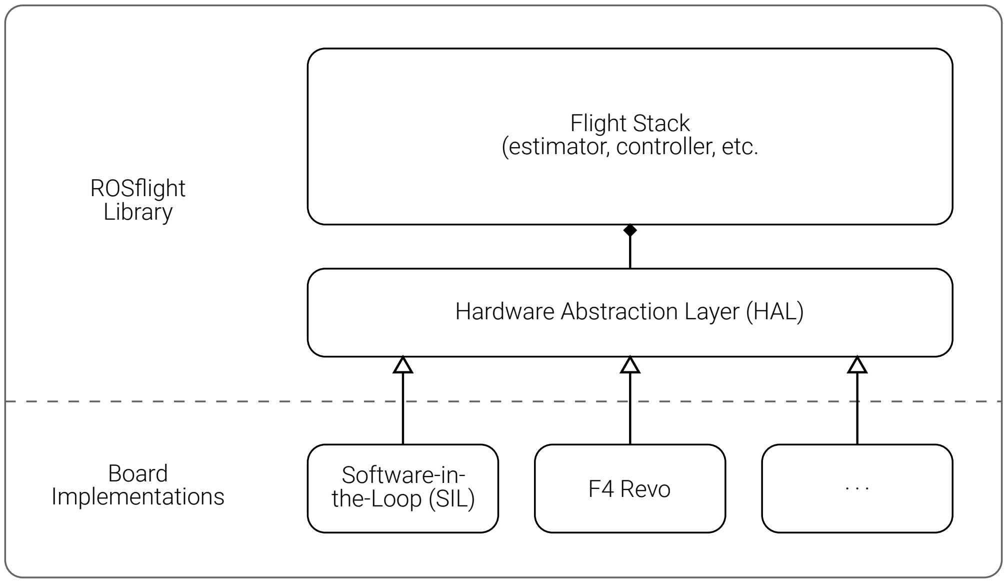

Flight Stack
(estimator, controller, etc.
ROSflight Library
Hardware Abstraction Layer (HAL)
Board Implementations
Software-in-the-Loop (SIL)
F4 Revo
. . .

Figure 3.7: Hardware abstraction layer and SIL

The mixer is also responsible for incorporating general-purpose inputs from the companion computer to additional actuators not essential to flight. This allows the user to perform tasks such as raising or lowering landing gear or dropping payloads without additional hardware.

3.4 Integration

Beyond a high-quality implementation of the flight control algorithms and associated software, ROSflight is designed to be easily integrated into an existing hardware and software system. As shown in Figure 3.7, the flight stack described in Section 3.3 makes all hardware-level calls through a hardware abstraction layer (HAL). This architecture makes integrating the flight stack into new hardware quite simple and makes true software-in-the-loop (SIL) simulations possible.

ROSflight is currently distributed with a hardware board interface to STM32F4-based flight control boards derived from the OpenPilot F4 Revolution flight controller [32] This hardware has become popular in first-persion view (FPV) drone racing and is therefore widely available, inexpensive and made to a high quality standard. The F4 board layer implements functions for reading all the associated sensors, writing to motors, serial, and USB connectivity and clock-related functions, such as getting the current time in microseconds from system boot. Although this is the only board family currently officially supported by the ROSflight organization, other flight control boards, such as STM32F1 and STM32F3-based FCUs have been made compatible through implementing the required board-level functions. Figure 3.1 shows a multirotor MAV and a fixedwing MAV, which both used ROSflight in visual-inertial state estimation research efforts.

Software-in-the-loop simulation is a valuable tool for researchers who wish simulations to be as close to hardware as possible. While other flight controllers support software-in-the-loop or even hardware-in-the-loop simulation, most of them require configuring the flight control stack into a simulation mode. This mode changes some aspects of how the flight controller operates, and therefore is only a partial SIL simulation. In constrast, a ROSflight SIL implementation will simply instantiate the proper API for the HAL to operate, and (depending on processor architecture) the same exact flight stack library that is run on hardware could be potentially linked to the simulation for accurate simulation testing. The ROSflight flight stack does not operate any differently while in a SIL environment, and there is no configuration to enable SIL so long as the simulation HAL properly emulates the hardware. One notable feature of this architecture is that the HAL also defines the means that the flight stack accesses the current time. This makes faster-than-realtime simulations possible even in SIL, which is useful for applications such as reinforcement learning.

ROSflight is currently distributed with a HAL implementation for the Gazebo robot simulator [11], however other, more visually realistic simulations have been developed which support ROSflight SIL using the Unreal 4 and Unity video game engines such as the Holodeck simulation [12] based on the Unreal 4 video game engine. Images of a ROSflight simulation in Gazebo are shown in Figures 3.8 and 3.9, and the Unreal 4 Holodeck ROSflight SIL simulation is shown in Figure 3.10.

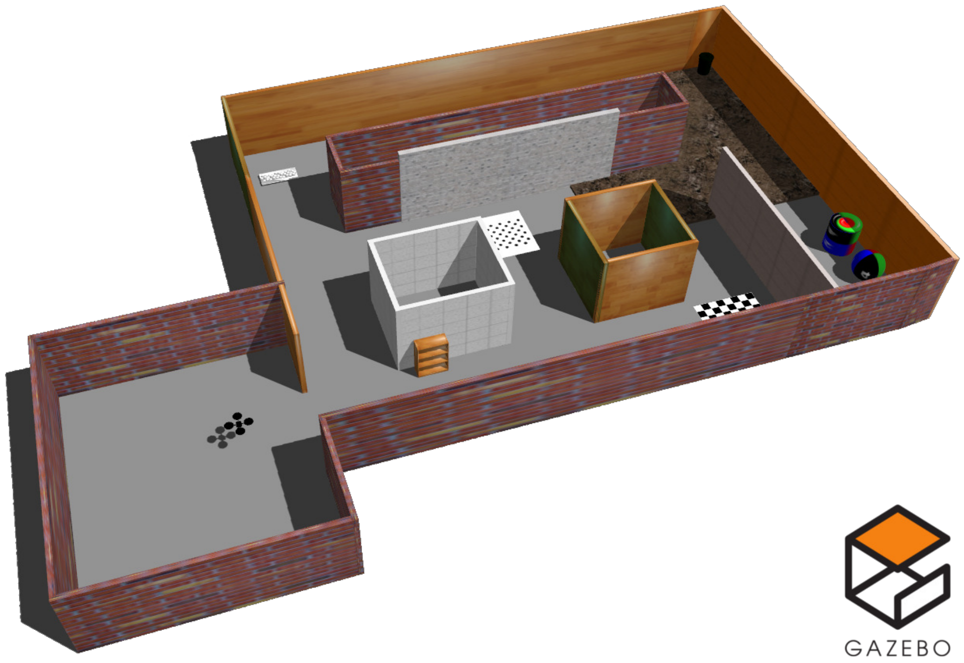


Figure 3.8: Simulated multirotor aircraft in Gazebo simulation environment running ROSflight SIL

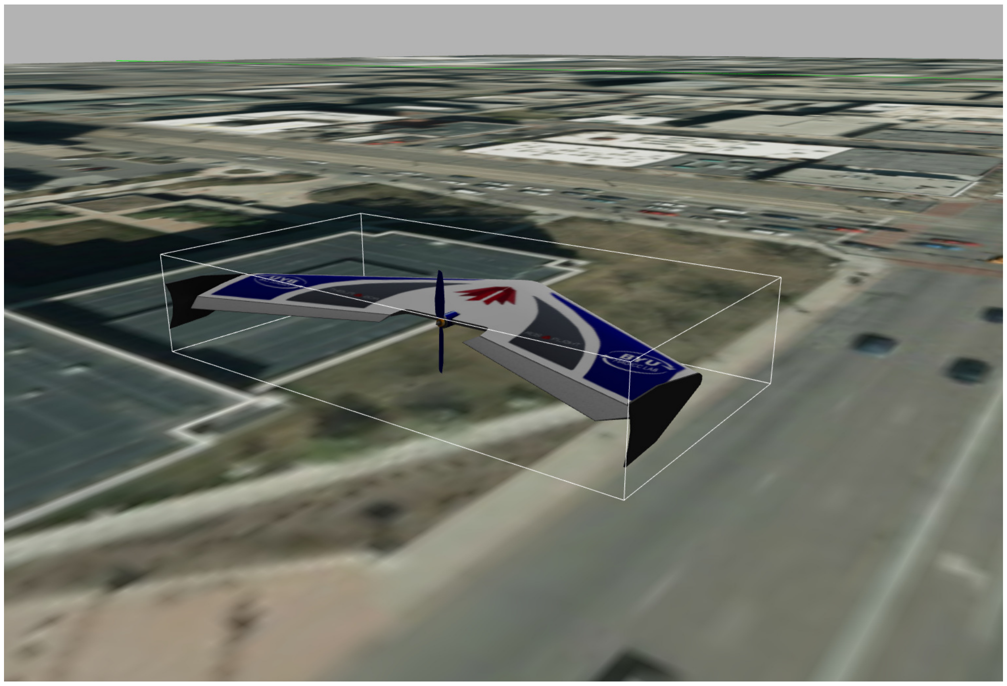


Figure 3.9: Simulated fixedwing aircraft in Gazebo simulation environment running ROSflight SIL

Figure 3.10: Simulated multirotor aircraft in Unreal 4 Holodeck simulation environment running ROSflight SIL

3.5 Conclusion

ROSflight is a lightweight, flexible, easy-to-understand research autopilot designed to meet the needs of researchers in autonomous operation of MAVs. The built-in functionality has by design been limited to a very narrow scope to enable easy understanding and integration into other projects, and serves as a base upon which higher-level functionality can be developed. To date, ROSflight has been used in research efforts focused on optimal control of quadrotors (both MPC [10] and LQR [13]), and visual-inertial Kalman Filtering [28]. However several unpublished works are currently in progress at several organizations in areas such as distributed SLAM, cooperative control and mapping of MAVs and GPS degraded autonomy. All of these works have required significantly more access to sensors and actuators than are possible on other flight control systems, and demonstrate ROSflight as a flexible research platform for control and state estimation.

3.6 Appendix: Estimator Algorithm Description

3.6.1 Introduction

The estimator is used to calculate an estimate of the attitude and angular velocity of the multirotor. It is assumed that the flight controller is mounted rigidly to the body of the aircraft (perhaps with dampening material to remove vibrations from the motors), such that measurements of the onboard IMU are consistent with the motion of the aircraft.

Due to the limited computational power on the embedded processor, and to calculate attitude estimates at speeds up to 1000 Hz, a simple complementary filter is used, rather than an extended Kalman filter. This method is used widely throughout commercially available autopilots. There are a variety of complementary filters, but the general theory is the same. A complementary filter is a method to fuse the measurements from a gyroscope, accelerometer and sometimes magnetometer to produce an estimate of the attitude of the MAV.

3.6.2 Sensors

There are a variety of sensors available on most flight control units, but we will limit discussion to accelerometers and rate gyroscopes. Both of these sensors are important in estimating the attitude of an MAV.

Accelerometers Accelerometers measure all the forces acting on the aircraft. We can assume that these forces consist primarily of forces from the propellers, but it often also consists of vibrations from unbalanced motors and propellers, wind, ground effect, and a variety of other sources. One thing accelerometers do *not* measure is acceleration due to gravity. As a result, if one holds an accelerometer still and look at the acceleration measurement values, one will notice it measures the forces required to hold it in place. For example, if an accelerometer sitting on a table will measure the normal force the table exerts on the accelerometer. If we assume that the accelerometer is at rest (albeit a rather significant assumption for a MAV), then we can assume that the forces acting on the accelerometer must be directly opposite gravity. Knowing then the direction of gravity with respect to the body-fixed axes of the MAV allows us to infer the MAV's roll and pitch angles.

As one might expect, a MAV is not always at rest, so accelerometer measurements tend to vary as the MAV accelerates. In addition, there are often also significant high-frequency external forces (such as vibrations induced by imbalanced propellers and motors) on a MAV in flight which adds noise to the accelerometer measurement. We will assume, however, that these accelerations are short-lived, and the accelerometer measurement remains stable over time.

Modern MEMS accelerometers suffer from a significant degree of temperature-dependent bias. This can be compensated for by recording the value of the stationary bias for all axes of the accelerometer during a temperature sweep, and performing a least-squares approximation. In ROSflight, this approximation is first-order, and takes place on the companion computer to take advantage of the superior computing power and ability to handle large amounts of data. Biases and compensation coefficients are then sent and applied on the flight controller.

Rate Gyrocopes Rate gyroscopes, or gyros, on the other hand, directly measure the angular velocity of the IMU. Gyros are not susceptible to external forces like accelerometers, and instead measure angular velocity directly. Unfortunately, integrating gyroscopes over time to calculate attitude would lead to unbounded drift. Common MEMS rate gyros also often have some non-zero bias which must be corrected for their use on MAVs. This bias wanders in a stochastic manner, and cannot be corrected entirelly through temperature compensation. It can, however, be estimated using certain filters.

3.6.3 Complementary Filtering

The idea behind complementary filtering is to try to get the best of both worlds of gyros and accelerometers. Gyros are very accurate over short timescales, but they are subject to drift. Accelerometers do not drift over long timescales, but they are noisy as the MAV moves about. Therefore, to solve these problems, the complementary filter primarily propagates states using gyroscope measurements, but then corrects drift with the accelerometer. In some ways, this process is similar to high-pass filtering gyroscope measurements and low-pass filtering the accelerometer measurements, then fusing the two together in a manner that results in an estimate that is stable over time, but also able to handle quick transient motions.

3.6.4 Attitude Representation

There are a number of ways to represent the attitude of a MAV. Most commonly, attitude is represented in terms of the Euler angles yaw, pitch and roll, but it can also be represented in other ways, such as rotation matrices and quaternions. In this paper, we will refer to $\boldsymbol{\theta}$ as the vector of Euler angle values, roll (ϕ) pitch (θ) and yaw (ψ)

$$\boldsymbol{\theta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

Euler Angles Euler angles represent rotations about sequential axes. This method is often the most easy for users to understand and interpret, but it is by far the least computationally efficient. To propagate Euler angles, the following kinematics are employed:

$$\dot{\boldsymbol{\theta}} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \boldsymbol{\omega}. \tag{3.1}$$

Note the large number of trigonometric functions associated with this propagation. In a complementary filter, this will be evaluated at every measurement, and the nonlinear coupling between ω and the attitude becomes expensive, particularly on embedded processors.

Another shortcoming of Euler angles is known as *gimbal lock*. Gimbal lock occurs at the singularity of the Euler angle representation, or when θ is equal to 90 or negative 90 degrees. The problem occurs because there is more than one way to represent this particular rotation, and numerical errors arise in finite-precision processors as one approaches the singularity. There are some steps one can take to handle these issues, but it is a fundamental problem associated with using Euler angles and it motivates the use of other attitude representations.

Rotation Matrices Rotation matrices (also known as direction cosine matrices or DCMs) are often used in attitude estimation because they do not suffer from gimbal lock, are quickly converted to and from Euler angles, and because of their simple kinematics, given as

$$\dot{R}_I^b = \lfloor \omega_{b/I}^b \rfloor R_I^b,$$

where $\lfloor \omega \rfloor$ is the skew-symmetric matrix of ω, and is related to calculating the cross product as

$$\lfloor \omega \rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix},$$

which is much simpler and faster to compute than Equation 3.1.

A rotation matrix from the inertial to body frame can be constructed from Euler angles via

$$R_I^b = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & s_\phi s_\theta \end{bmatrix},$$

and converting back to Euler angles is done with

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(R_{32}, R_{33}\right) \\ \text{atan2}\left(-R_{31}, \sqrt{R_{21}^2 + R_{33}^2}\right) \\ \text{atan2}\left(R_{21}, R_{11}\right) \end{bmatrix}.$$

Unit Quaternions Quaternions are a number system that extends complex numbers. They have four elements, which we will refer to as $\boldsymbol{q}_0$, $\boldsymbol{q}_x$, $\boldsymbol{q}_y$, and $\boldsymbol{q}_z$. The group of unit quaternions can be used to represent attitude, which is related to an axis-angle representation. The last three elements can be though of as describing a unit vector $\boldsymbol{\beta}$ about which a rotation occurred and the first element, $\boldsymbol{q}_0$ can be though of as describing the amount of rotation α about that axis as in

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_0 \\ \boldsymbol{q}_x \\ \boldsymbol{q}_y \\ \boldsymbol{q}_z \end{bmatrix} = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2)\cos(\boldsymbol{\beta}_x) \\ \sin(\alpha/2)\cos(\boldsymbol{\beta}_y) \\ \sin(\alpha/2)\cos(\boldsymbol{\beta}_y) \end{bmatrix} = \begin{bmatrix} s \\ \boldsymbol{v}_x \\ \boldsymbol{v}_y \\ \boldsymbol{v}_z \end{bmatrix}.$$

While this may seem straight-forward, unit quaternions used for attitude representations must be normalized so that they form a group. (That is, a unit quaternion multiplied by a unit quaternion is a unit quaternion), so they are sometimes difficult to interpret from inspecting the values alone. However, they provide some significant computational efficiencies, most of which comes from the following special mathematics associated with quaternions.

The unit quaternion representation of an attitude can be computed from the Euler-angle representation with

$$\boldsymbol{q} = \begin{bmatrix} \cos(\theta/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\theta/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\theta/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\theta/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix},$$

and is converted back to Euler angles with

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(2\left(\boldsymbol{q}_0\boldsymbol{q}_x + \boldsymbol{q}_y\boldsymbol{q}_z\right), 1 - 2\left(\boldsymbol{q}_x^2 + \boldsymbol{q}_y^2\right)\right) \\ \arcsin\left(2\left(\boldsymbol{q}_0\boldsymbol{q}_y - \boldsymbol{q}_z\boldsymbol{q}_x\right)\right) \\ \operatorname{atan2}\left(2\left(\boldsymbol{q}_0\boldsymbol{q}_z + \boldsymbol{q}_x\boldsymbol{q}_y\right), 1 - 2\left(\boldsymbol{q}_y^2\boldsymbol{q}_z^2\right)\right) \end{bmatrix}.$$

Unit quaternions are closed under the following operation, termed quaternion multiplication

$$\boldsymbol{q_1} \otimes \boldsymbol{q_2} = \begin{bmatrix} s_1 s_2 - \boldsymbol{v_1}^\top \boldsymbol{v_2} \\ s_1\boldsymbol{v_2} + s_2\boldsymbol{v_1} + \boldsymbol{v_1} \times \boldsymbol{v_2} \end{bmatrix},$$

and the inverse of a unit quaternion can be computed by negating the vector portion $\boldsymbol{v}$ as

$$\boldsymbol{q}^{-1} = \begin{bmatrix} \boldsymbol{q}_0 \\ -\boldsymbol{q}_x \\ -\boldsymbol{q}_y \\ -\boldsymbol{q}_z \end{bmatrix}.$$

Unit quaternions can be differenced by multiplying the inverse of one quaternion with the other, as in

$$\tilde{\boldsymbol{q}} = \hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}.$$

Unit quaternion dynamics are given below, as

$$\dot{\boldsymbol{q}}_I^b = \frac{1}{2}\boldsymbol{q}_I^b \otimes \begin{bmatrix} 0 \\ \omega_{b/I}^b \end{bmatrix}.$$

What this means is that, like rotation matrices, unit quaternion dynamics are much simpler than Euler angle dynamics and they also take less computation than rotation matrices because they have fewer terms. In one study, complementary filters using an Euler angle representation was twelve times as costly on average than a unit quaternion-based filter [31]. The unit quaternion representation was also about 20 percent more efficient than a rotation matrix representation. For these reasons, ROSflight uses unit quaternions to represent attitude in its filter.

3.6.5 Derivation

ROSflight implements the unit quaternion-based passive nonlinear observer filter as described in [30]. In particular, we implement Equation 47 from that paper, which also estimates gyroscope biases. A Lyuapanov stability analysis is given in [30] which shows that all states and biases, except heading, are globally asymptotically stable given an accelerometer measurement and gyroscope. The above reference also describes how a magnetometer can be integrated in a similar method to the accelerometer, However, that portion of the filter is ommitted here due to the unreliable nature of magnetometers onboard modern small UAS.

Passive Complementary Filter The original filter propagates per the following dynamics:

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes \begin{bmatrix} 0 \\ \omega_{comp} \end{bmatrix}$$

$$\dot{\hat{\boldsymbol{b}}} = -2k_i\omega_{acc},$$

where $\boldsymbol{\omega}_{comp}$ consists of a combination of the angular rate measurement, $\bar{\boldsymbol{\omega}}$, the estimated gyroscope biases, $\hat{\boldsymbol{b}}$, and an adjustment term, $\boldsymbol{\omega}_{acc}$, as

$$\boldsymbol{\omega}_{comp} = \bar{\boldsymbol{\omega}} - \hat{\boldsymbol{b}} + k_P \boldsymbol{\omega}_{acc} \quad , \tag{3.2}$$

and the constant gains k_p and k_i are used in determining the dynamics of the filter.

The accelerometer adjustment term $\boldsymbol{\omega}_{acc}$ can be described as the error in the attitude as predicted by the accelerometer. To calculate $\boldsymbol{\omega}_{acc}$ the quaternion describing the rotation between the accelerometer estimate and the inertial frame, $\boldsymbol{q}_{acc}$, is first calculated as

$$\boldsymbol{a} = \frac{\boldsymbol{y}_{acc}}{\|\boldsymbol{y}_{acc}\|}, \qquad \boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \boldsymbol{\gamma} = \frac{\boldsymbol{a} + \boldsymbol{g}}{\|\boldsymbol{a} + \boldsymbol{g}\|}, \qquad \boldsymbol{q}_{acc} = \begin{bmatrix} \boldsymbol{a}^\top \boldsymbol{\gamma} \\ \boldsymbol{a} \times \boldsymbol{\gamma} \end{bmatrix},$$

where $\boldsymbol{y}_{acc}$ is the accelerometer measurement. Next, the quaternion error between the estimate $\hat{\boldsymbol{q}}$ and the accelerometer measure $\boldsymbol{q}_{acc}$ is calculated as,

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}_{acc}^{-1} \otimes \hat{\boldsymbol{q}} = \begin{bmatrix} \tilde{s} \\ \tilde{\boldsymbol{v}} \end{bmatrix}.$$

and finally, $\boldsymbol{q}_{acc}$ is converted back into a vector $\in \mathbb{R}^3$ with

$$\boldsymbol{\omega}_{acc} = 2\tilde{s}\tilde{\boldsymbol{v}},$$

as described in [30].

Modifications to Original Passive Filter There have been a few modifications to the passive filter described in [30], consisting primarily of contributions from [31]. Firstly, rather than simply taking gyroscope measurements directly as an estimate of ω, a quadratic polynomial is used to approximate the true angular rate from gyroscope measurements to reduce error. In [31], this process was shown to reduce RMS error by more than 1,000 times. The equation to perform this calculation is given as

$$\bar{\omega} = \frac{1}{12}\left(-\omega\left(t_{n-2}\right) + 8\omega\left(t_{n-1}\right) + 5\omega\left(t_n\right)\right),$$

where $\omega(t_{n-i})$ is the gyroscope measurement of the angular rate i measurements previous.

The second modification is in the way that the attitude is propagated after finding $\dot{\hat{\boldsymbol{q}}}$. Instead of performing first-order Euler integration as in

$$\hat{\boldsymbol{q}}_n = \hat{\boldsymbol{q}}_{n-1} + \dot{\hat{\boldsymbol{q}}}_n \delta t,$$

we use an approximation of the matrix exponential. The matrix exponential arises out of the solution to the differential equation $\dot{\mathbf{x}} = A\mathbf{x}$, namely

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0)$$

and the discrete-time equivalent

$$\mathbf{x}\left(t_{n+1}\right) = e^{A\delta t}\left(t_n\right).$$

This discrete-time matrix exponential can be approximated by first expanding the matrix exponential into the infinite series

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

and then grouping odd and even terms from the infinite series into two sinusoids. This results in the following equation for the propagation of the filter dynamics

$$\hat{\boldsymbol{q}}(t_n) = \left[\cos\left(\frac{\left\|\omega_{comp}\right\|\delta t}{2}\right) I_4 + \frac{1}{\left\|\omega_{comp}\right\|}\sin\left(\frac{\left\|\omega_{comp}\right\|\delta t}{2}\right)\lfloor\omega_{comp}\rfloor_4\right]\hat{\boldsymbol{q}}(t_{n-1}), \qquad (3.3)$$

where $\lfloor\boldsymbol{\omega}\rfloor_4$ is the 4×4 skew-symmetric matrix formed from $\boldsymbol{\omega}$ given as

$$\lfloor\boldsymbol{\omega}\rfloor_4 = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}.$$

External Attitude Measurements Using the ROSflight estimator with gyro measurements only will quickly drift due to gyro biases. The accelerometer makes the biases in ω_x and ω_y observable and provides a measurement of pitch and roll under unaccelerated conditions. To make yaw observable, and to make attitude observable during accelerated motions, an external attitude measurement can be provided to the estimator. This is fused in much the same way as the accelerometer, with an additional correction term $\boldsymbol{\omega}_{err}$ that is calculated as

$$\boldsymbol{\omega}_{err} = k_{ext} \sum_{i=1}^{3} R\left(\hat{\boldsymbol{q}}\right)^{\top} \boldsymbol{e}_i \times R\left(\bar{\boldsymbol{q}}\right)^{\top} \boldsymbol{e}_i,$$

where $\boldsymbol{e}_i$ is the unit vector in the pointing along the i-th basis vector of the standard coordinate frame, $\bar{\boldsymbol{q}}$ is the externally-supplied attitude measurement and where $k_{ext} = {}^{F_s^{IMU}}/_{F_s^{ext}}$ is the ratio of the IMU sample rate to the external attitude sample rate.

When using the external attitude measurement, the composite angular rate (Equation 3.2) is expanded to become

$$\boldsymbol{\omega}_{comp} = \bar{\boldsymbol{\omega}} - \hat{\boldsymbol{b}} + k_P\left(\boldsymbol{\omega}_{acc} + \boldsymbol{\omega}_{err}\right)$$

before propagating the filter dynamics (Equation 3.3).

Tuning The filter can be tuned with the two gains k_P and k_I. Upon initialization, k_P and k_I are deliberately set high to quickly cause the filter to converge upon approprate values. After a few seconds, however they are both reduced by a factor of ten, to a value chosen through manual tuning. A high k_P will cause sensitivity to transient accelerometer errors, while a small k_P will cause sensitivity to gyroscope drift. A high k_I will cause biases to wander unnecessarily, while a low k_I will result in slow convergence to accurate gyroscope bias estimates.

CHAPTER 4: OPTIMAL CONTROL OF A MULTIROTOR USING ERROR-STATE FORMULATION

4.1 Introduction

In the last decade, we have seen the emergence of miniature aerial vehicles (MAVs) in everyday life for many industries. The price and weight of processing power, sensors, and motors have decreased dramatically in recent years and has resulted in modern MAVs being incredibly agile and acrobatic. To highlight this point, the sport of quadrotor racing among MAV enthusiasts has seen top speeds regularly exceeding 100 kph.

The most skilled human operators provide angular rate and throttle commands to an embedded flight controller at a rate of 50 Hz using a radio transmitter, and often only receive feedback from a radio video link transmitted from the quadrotor. From a state estimation and control perspective, the abilities of human operators of MAVs are truly incredible and we have yet to see equivalent performance by an autonomous MAV. This is due to a variety of factors including limitations in state estimation, path planning, and control. In this work, we hope to apply the optimal control methods infinite-horizon linear quadratic regulation (LQR) and finite-horizon linear model predictive control (MPC) to a quadrotor to perform waypoint following, as one step towards reaching human performance in MAV control.

There are many existing approaches to performing optimal control on quadrotors, but most of these rely on a flight controller to perform higher-level attitude commands as opposed to the lower-level angular rates given by skilled human operators [33–38]. Providing angular rate commands, as opposed to full attitude control, allows for acrobatic maneuvers, outside the limitations of the flight controller attitude control scheme. Many of these approaches developed previously rely on

Figure 4.1: Quadrotor used for modeling

,

an Euler-angle decomposition [33–38] of the attitude or ignored attitude dynamics [39,40] to avoid dealing with the non-vector nature of rotations. In one notable approach, [41] changes the Euler axis representation such that the singularity occurred at ninety degrees of heading, instead of pitch and could therefore perform acrobatics, however, even this implementation was limited in the types of trajectories available.

While Euler angles are not a vector space [42], they approximate a vector space given a linearization [43]. However, they suffer from a singularity when pitched to 90 degrees, suffer from linearization error when deviated from level, and they require some book keeping to deal with wrapped angles. Therefore, most existing optimal control problems do not actually perform acrobatic flight where they would have to leave the linearization point used to cast Euler angles into a vector space.

Recently there has been a movement in the robotics community to appropriately deal with the evolution of a robot's state along a manifold using Lie theory [44]. Though these methods have been widely used in the field of state estimation [45] [46], a few methods have emerged that also apply Lie theory to control [47] [48]. We propose a formulation of the optimal control problem that properly deals with the manifold nature of the state, specifically the attitude component. While some methods use unit quaternions or rotation matrices to properly represent attitude, these are also not inherently a vector space and extra steps are required to orthonormalize or otherwise force the attitude to stay on the manifold [49] [14]. In contrast, our proposed solution is derived from Lie theory and care is taken to ensure that all vector manipulations are done with appropriate vector quantities so that the state remains on the manifold.

Another important consideration to good quadrotor controller design is achieving real-time performance. Solutions to optimal control methods such as LQR and MPC are significantly more complicated than nonlinear or PID control methods, so a lot of research effort has been put into efficient solutions to optimal control problems. Some methods to achieve real-time performance have included performing LQR with a fixed gain [42], or gain scheduling with a library of LQR gains for different deviations from the desired state [49]. Some recent notable examples include [14], which proposes re-linearizing the system at a fixed rate, slower than the control loop and recomputing the LQR gain at this interval and [10] which achieved real-time performance of an MPC controller using a novel matrix-free solver.

In our case, a side benefit of performing control directly on the manifold is that it not only improves accuracy but it also vastly improves the computational efficiency. In contrast to other methods, we are able to linearize and solve the optimal control problem at every control step. Farrell et al. [13] used this formulation to achieve 400μs control loop rates onboard a quadrotor during tracking.

This chapter is organized as follows: First, we will discuss the dynamics of quadrotor agents and the specific modeling considerations used in the controller design. Next, we will derive the error-state model used to perform optimal control over the manifold, and describe our implementation.

Third, we will describe results from a simulation study and finally, we will discuss the potential for future work, most of which was undertaken by Farrell et al. [13].

4.1.1 Nomenclature

$\mathbf{p}_{b/I}^{I}$:	Position of the MAV body frame with respect to the inertial frame, expressed in the inertial frame.
$\mathbf{v}_{b/I}^{b}$:	Velocity of the MAV body frame with respect to the inertial frame, expressed in the body frame.
$\mathbf{q}_{I}^{b}$:	Rotation from the inertial to the body frame, expressed in the inertial frame as a Hamilton unit quaternion.
$\omega_{b/I}^{b}$:	Angular velocity of the MAV in body frame.
T^b :	Total thrust of quadrotor (in body frame).
s :	Throttle signal between 0 and 1 which is used by a flight controller to control T^b.
$\lfloor\cdot\rfloor_\times$:	The skew-symmetric matrix operator.
$\mathbf{i}, \mathbf{j}, \mathbf{k}$:	The orthonormal unit vectors which describe the x, y, and z axes of a standard coordinate frame.

4.1.2 Quaternions

A quaternion $\mathbf{q}$ is a hyper-complex number of rank four. It is well known that a unit quaternion $\in S^3$ can be used to efficiently represent attitude, as S^3 is a double cover of $SO(3)$. Quaternions have the advantage over $SO(3)$ of being more efficient to implement on modern hardware [31], therefore in the software implementation of the described algorithm, we use quaternions, rather than rotation matrices.

We use Hamiltonian notation for unit quaternions $\in S^3$

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_0 & \mathbf{q}_x i & \mathbf{q}_y j & \mathbf{q}_z k \end{pmatrix}$$

and define the complex numbers i, j, and k, such that

$$\begin{aligned} ij &= -ji &= k, \\ jk &= -kj &= i, \\ ki &= -ik &= j, \\ i^2 = j^2 = k^2 &= ijk &= -1. \end{aligned}$$

For convenience, we sometimes refer to the complex portion of the quaternion as

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top$$

and write quaternions as the tuple of the real and complex portions

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{\mathbf{q}} \end{pmatrix}.$$

Given our use of the Hamiltonian notation, the quaternion group operator $\otimes$ can be written as the following matrix-like product

$$\mathbf{q}^a \otimes \mathbf{q}^b = \begin{pmatrix} -q_0^a & (-\bar{\mathbf{q}}^a)^\top \\ \bar{\mathbf{q}}^a & q_0^a I + \lfloor \bar{\mathbf{q}}^a \rfloor_\times \end{pmatrix} \begin{pmatrix} q^b \\ \bar{\mathbf{q}}^b \end{pmatrix}.$$

We note that rotations may be written equivalently as $R\left(\mathbf{q}_a^b\right) = R_a^b$, where the choice of these is dictated by convenience. We use passive rotation matrices, meaning that the rotation matrix R_a^b acts on a vector $\mathbf{r}^a$, expressed in frame a, and results in the same vector, now expressed in frame b as

$$\mathbf{r}^b = R_a^b \mathbf{r}^a.$$

It is often convenient to convert a quaternion $\mathbf{q}$ to its associated passive rotation matrix. This is done with

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0 \lfloor \bar{\mathbf{q}} \rfloor_\times + 2\bar{\mathbf{q}}\bar{\mathbf{q}}^\top \in SO(3).$$

We also need to frequently convert between the Lie group of the unit quaternion, S^3, and the Lie algebra, $\mathbb{R}^3$, which enables us to operate in a vector space. This is done with the exponential and logarithmic mappings. The exponential mapping for a unit quaternion is defined as

$$\begin{aligned} \exp_{\mathbf{q}} &: \mathbb{R}^3 \to S^3 \\ \exp_{\mathbf{q}}(\boldsymbol{\delta}) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{\delta}\|}{2}\right) \\ \sin\left(\frac{\|\boldsymbol{\delta}\|}{2}\right)\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|} \end{bmatrix}, \end{aligned} \tag{4.1}$$

with the corresponding logarithmic map defined as

$$\begin{aligned} \log_{\mathbf{q}} &: S^3 \to \mathbb{R}^3 \\ \log_{\mathbf{q}}(\mathbf{q}) &\triangleq 2\,\mathrm{atan2}\left(\|\bar{\mathbf{q}}\|, q_0\right)\frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}. \end{aligned} \tag{4.2}$$

To avoid numerical issues when $\|\bar{q}\| \approx 0$, we also employ the small-angle approximations of the quaternion exponential and logarithm

$$\begin{aligned} \exp_{\mathbf{q}}(\boldsymbol{\delta}) &\approx \begin{bmatrix} 1 \\ \frac{\boldsymbol{\delta}}{2} \end{bmatrix} \\ \log_{\mathbf{q}}(\mathbf{q}) &\approx 2\,\mathrm{sign}(q_0)\,\bar{\mathbf{q}}. \end{aligned}$$

In this work, we will be modeling the quadrotor MAV shown in Figure 4.1. This quadrotor weighs approximately 14 oz with a battery and measures 220 mm diagonally from rotor center to rotor center. It also uses five-inch three-blade propellers and appropriately sized motors, battery and speed controllers.

4.2 Derivation

4.2.1 Quadrotor Dynamics

The free-body diagram in Figure 4.2 can be used to derive the dynamic model of a quadrotor with a linearized drag constant term μ, mass M and gravity vector $\mathbf{g}^I$, given in Equation 4.3, and shown in [50, 51]

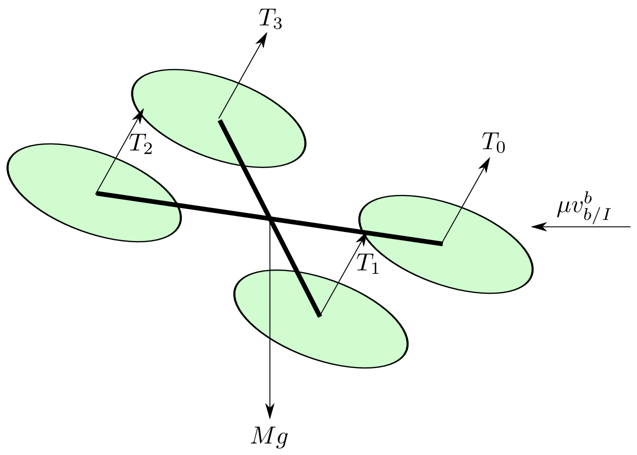


Figure 4.2: Forces acting on quadrotor MAV

$$\begin{aligned}
\dot{\mathbf{p}}^I_{b/I} &= R\left(\mathbf{q}^b_I\right)^\top \mathbf{v}^b_{b/I} \\
\dot{\mathbf{v}}^b_{b/I} &= R\left(\mathbf{q}^b_I\right)\mathbf{g}^I - \frac{T^b}{M} - \mu\mathbf{v}^b_{b/I} \\
\dot{\mathbf{q}}^b_I &= \omega^b_{b/I}.
\end{aligned} \tag{4.3}$$

Both because of the fast dynamics typical of a quadrotor of this size and to mimic the inputs available to a human operator, we will command a throttle signal and angular rates, rather than actual forces and torques. However, we cannot command thrust directly, as the flight controller does not have access to the motor model. Instead, we are able to command a *throttle* value between 0 and 1, where 1 maps to maximum thrust, 0 maps to no thrust, and a non-linear mapping in between.

To obtain this mapping, we collected thrust and torque data on a test stand. This data is shown in Figure 4.3 and indicates a strong quadratic relationship between throttle signal s and thrust T_i and almost no transients.

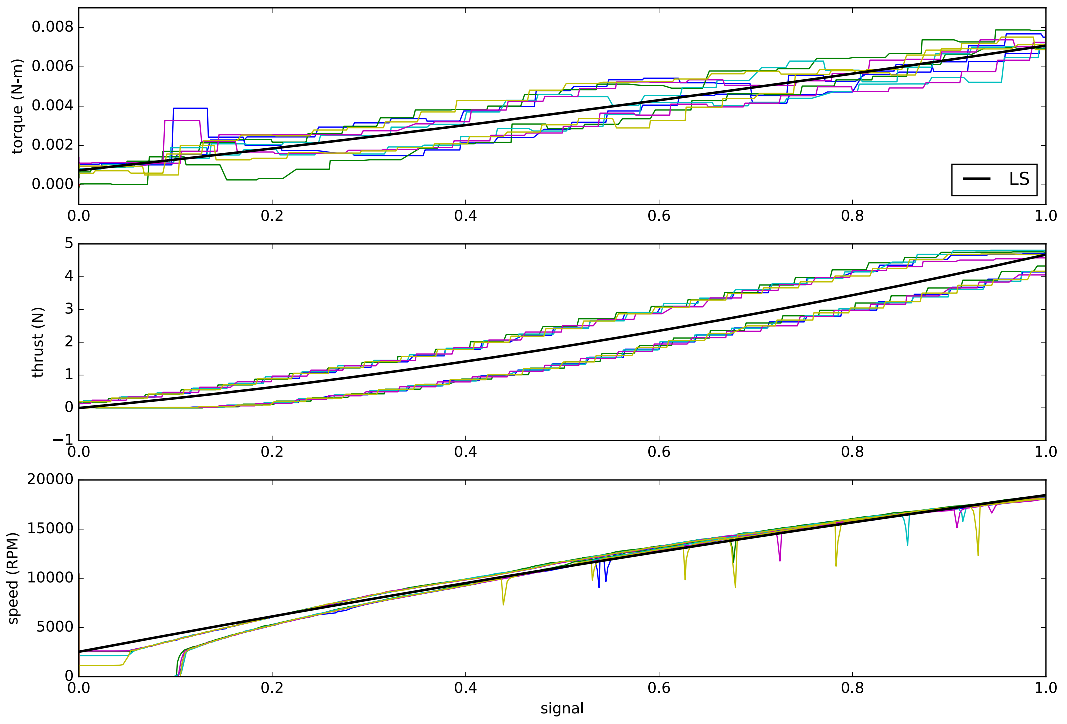


Figure 4.3: Experimental data from thrust and torque stand. It is unclear what causes what appears to be hysteresis in the torque and thrust plots. The two main paths are the sweep up (higher) and sweep down (lower). The reason for this difference is unclear because the speed plot does not have the same hysteresis, and therefore is unlikely to be due to transients in motor speed response or actual hysteresis in the motor. One hypothesis is that the mass of air in the test stand environment introduces a lag in torque readings, as thrust and torque are higher in static air environments. On a downward sweep, the torque and thrust are less, because the air is already moving. Blips in the speed plot are due to sampling errors.

We performed a least-squares quadratic regression on the data shown in Figure 4.3 to solve for the coefficients of a quadratic model for thrust versus signal, given below:

$$T_i = \left(a_T s_i^2 + b_T s + c_T\right)\mathbf{k}. \tag{4.4}$$

The coefficients for this model for our data set are given in Table 4.1. The total thrust simply the sum of each of the four motors, given as

$$T^b = \sum_{i=0}^{4} T_i.$$

Modeling the applied torques for our quadrotor is slightly more complicated, as we must make use of the geometry of each arm. Let $\mathbf{r}^b_{i/b}$ be the vector that describes the location of motor i with respect to the center of mass in the body frame. There are two components of torque from each motor, the first is due primarily to drag of the propeller, and is fitted with a quadratic fit from the data shown in Figure 4.3. The second is due to the thrust of the motor acting on the lever arm

$$\tau_i = \left\lfloor \mathbf{r}^b_{i/b} \right\rfloor_\times T_i + \left(a_\tau s_i^2 + b_\tau s + c_\tau\right)\mathbf{k}.$$

The total torque is simply calculated as the sum of torques from the four motors as

$$\tau^b = \sum_{i=0}^{4} \tau_i.$$

As mentioned before, we could, in theory, use this model to control torque directly, however, like a human operator, we are going to rely on the low-level flight controller to perform angular rate control at much higher rates. It is unreasonable to directly control the angular velocity of a quadrotor of this size with MPC due to computational and communication limitations. Instead, we will convert T^b to a desired throttle signal s, assuming that all motors are commanded the same signal (the case that the aircraft is in hover).

To get s, we simply take Equation 4.4 and solve for s as follows:

$$\begin{aligned} T^b &= 4\left(a_T s^2 + b_T s + c\right) \\ 0 &= a_T s^2 + b_T s + c - \frac{T^b}{4} \end{aligned}$$

$$s = \frac{-b_T + \sqrt{b_T^2 - 4a_T\left(c_T - \frac{T^b}{4}\right)}}{2a_T}.$$

This calculation will return a real signal, so long as

$$\begin{aligned} b_T^2 - 4a_T\left(c_T - \frac{T^b}{4}\right) &> 0 \\ \frac{-b_T^2}{a_T} + 4c_T &< T^b, \end{aligned}$$

which in this case, using the values in Table 4.1 leads us to a lower-bound on T^b for a valid signal of

$$-4.1867 < T^b.$$

This is not a concern for us, as this equates a value of s outside of valid bounds, which are $[0, 1]$.

Table 4.1: Coefficients to quadratic approximation to motor response for torque and thrust found using a least-squares regression of test data.

	τ	T
a	0.0010008	1.87953563
b	0.00533861	2.79829004
c	0.00074126	-0.00514948

If we substitute this expression for throttle control into the thrust term of our dynamics (Equation 4.3), we get the following expression for our quadrotor dynamics:

$$\begin{aligned} \dot{\mathbf{p}}_{b/I}^I &= R\left(\mathbf{q}_I^b\right)^\top \mathbf{v}_{b/I}^b \\ \dot{\mathbf{v}}_{b/I}^b &= R\left(\mathbf{q}_I^b\right) g^I - \frac{4}{M}\left(a_T s^2 + b_T s + c\right)\mathbf{k} - \mu \mathbf{v}_{b/I}^b \\ \dot{\mathbf{q}}_I^b &= \omega_{b/I}^b, \end{aligned}$$

and we define our state and inputs as

$$\mathbf{x} = \left[\mathbf{p}^{I}_{b/I}, \mathbf{v}^{b}_{b/I}, \mathbf{q}^{b}_{I}\right]^{\top} \quad \mathbf{u} = \left[s, \omega^{b}_{b/I}\right].$$

4.2.2 Error State Tracking

At this point we must face the difficulty of dealing with the quaternion attitude object. Quaternions are not vectors, and instead form a group. (It should also be mentioned that there is no sensible vector representation of attitude, due to the nature of $SO(3)$). The large majority of non-linear optimization solvers make the assumption that the optimization variables are vectors and leverage efficient linear algebra routines to perform the optimization. This is not possible if we use the group objects directly.

However, if we use the Lie algebra associated with $SO(3)$, we can perform control in the vector space of the difference between quaternions. To do this, we have to re-cast our problem into an error state formulation and perform control over the error between a reference trajectory and our current state.

In this work, our desired trajectory, $\check{\mathbf{x}}$, is a series of piecewise-constant waypoints in the form of a 2 meter by 2 meter square, located 5 meters above the starting position, with waypoints changed every twenty seconds. This trajectory is not dynamically feasible (as it would require instantaneous movement between waypoints) but is easy to calculate, and trying to achieve infeasible dynamic trajectories actually better demonstrates the effectiveness of optimal control. Although it is outside the scope of this work, it would be much better to generate dynamically feasible trajectories using methods such as trajectory optimization [52–54], or approaches that leverage the differential flatness of the quadrotor [40, 55]. The reference trajectory dynamics are the same as our normal quadrotor dynamics, given below for reference

$$\begin{aligned}
\dot{\check{\mathbf{p}}}^I_{b/I} &= R\left(\check{\mathbf{q}}^b_I\right)^\top \check{\mathbf{v}}^b_{b/I} \\
\dot{\check{\mathbf{v}}}^b_{b/I} &= R\left(\check{\mathbf{q}}^b_I\right) g^I - \frac{4}{M}\left(a_T \check{s}^2 + b_T \check{s} + c_T\right)\mathbf{k} - \mu \check{\mathbf{v}}^b_{b/I} \\
\dot{\check{\mathbf{q}}}^b_I &= \check{\omega}^b_{b/I},
\end{aligned}$$

and with our piecewise-constant waypoints, the reference inputs are the just the equilibrium throttle to remain at hover.

$$\check{\mathbf{u}} = \begin{bmatrix} \frac{-b_T + \sqrt{b_T^2 - 4a_T\left(c_T - \frac{Mg}{4}\right)}}{2a_T} \\ 0 \end{bmatrix}.$$

We are interested in driving the error between our desired trajectory and our state to zero. We define the error state of some quantity $\mathbf{y}$ as

$$\tilde{\mathbf{y}} = \mathbf{y} \boxminus \check{\mathbf{y}}, \tag{4.5}$$

where $\boxminus$ is an appropriate difference operator, as described by [56]. For instance, if $\mathbf{y}, \check{\mathbf{y}} \in \mathbb{R}^n$, the $\boxminus$ operator may be defined as the vector subtraction operator. However, due to the attitude component of our state, the vector subtraction operator is not defined between $\mathbf{x}$ and $\check{\mathbf{x}}$. We instead define the error state piecewise for each component of the state and combine these into an error state vector

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{p}} & \tilde{\mathbf{v}} & \tilde{\mathbf{r}} \end{bmatrix}^\top \in \mathbb{R}^{9\times 1},$$

where $\tilde{\mathbf{p}}$ is the error state associated with position, $\tilde{\mathbf{v}}$ is the error state associated with velocity, and $\tilde{\mathbf{r}}$ is the error state associated with attitude.

In our case, the error states associated with position and velocity are simply defined using vector subtraction

$$\begin{aligned}\tilde{\mathbf{p}} &= \mathbf{p}^I_{b/I} - \check{\mathbf{p}}^I_{b/I} \\ \tilde{\mathbf{v}} &= \mathbf{v}^b_{b/I} - \check{\mathbf{v}}^b_{b/I},\end{aligned}$$

however, the error state associated with attitude is more complicated.

It is commonly understood that any representation of attitude has three underlying degrees of freedom. A unit quaternion has four parameters, but its error can be described in terms of three degrees of freedom that we wish to represent as a vector quantity. In a neighborhood sufficiently close to the identity, these behave similarly to the Euler angle representation of roll, pitch, and yaw. However, Euler angles are not a vector because the sequential rotation method used to define Euler angles nonlinearly couples the three degrees of freedom. Therefore, we define the vector

$$\mathbf{r}^b_I(t) = \mathbf{r}^b_I(t_0) + \int_{t_0}^{t} \omega^b_{b/I}(\tau)\, d\tau,$$

such that $\mathbf{r}^b_I(t_0) = \mathbf{0}$ and $\dot{\mathbf{r}}^b_I = \boldsymbol{\omega}^b_{b/I}$. With this definition, we can use (4.1) and (4.2) to express

$$\begin{aligned}\mathbf{q}^b_I &= \check{\mathbf{q}}^b_I \otimes \exp_{\mathbf{q}}(\tilde{\mathbf{r}}) \\ \tilde{\mathbf{r}} &= \log_{\mathbf{q}}\left(\left(\check{\mathbf{q}}^b_I\right)^{-1} \otimes \mathbf{q}^b_I\right).\end{aligned}$$

Even though $\mathbf{r}$ is a vector, we cannot simply compute the attitude error state as $\tilde{\mathbf{r}} = \mathbf{r}^b_I - \check{\mathbf{r}}^b_I$ because $\mathbf{r}^b_I$ is a minimal representation of $\mathbf{q}^b_I$, which is a double cover of the Lie group $SO(3)$. Vector subtraction of members in this group is not valid. However, the derivative of $\mathbf{r}^b_I$ exists in the tangent space of $SO(3)$, so we can perform

$$\dot{\tilde{\mathbf{r}}} = \dot{\mathbf{r}}^b_I - R^b_I\left(\check{R}^b_I\right)^{\top} \dot{\check{\mathbf{r}}}^b_I, \tag{4.6}$$

where $R_I^b \left(\check{R}_I^b\right)^\top$ transforms the desired vector derivative, $\dot{\check{\mathbf{r}}}_I^b$, from its own tangent space to the tangent space of $\dot{\mathbf{r}}_I^b$. With both vectors in the same tangent space, the vector subtraction in (4.6) is valid.

For use in control, we similarly define an error state for the control input with the error state being the difference between the current control input and some reference input. Using the same definition as in (4.5), we can see that

$$\tilde{s} = s - \check{s}$$
$$\tilde{\omega} = \omega_{b/I}^b - \check{\omega}_{b/I}^b$$

where $\check{s}$ and $\check{\omega}_{b/I}^b$ are respectively the reference throttle signal and reference angular velocity. Note that we do not model the dynamic response to these inputs. Instead, our model assumes that the quadrotor instantaneously reaches any commanded throttle and angular velocity.

Using the error state definitions above, we can derive the error state dynamics of the quadrotor as

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}} &= \left(R_I^b\right)^\top \tilde{\mathbf{v}}_{b/I}^b - \left(R_I^b\right)^\top \left\lfloor \mathbf{v}_{b/I}^b \right\rfloor_\times \tilde{\mathbf{r}}_I^b \\
\dot{\tilde{\mathbf{v}}} &= \lfloor R(\mathbf{q})\, \mathbf{g} \rfloor_\times \tilde{\mathbf{q}} - \frac{4}{M}\left(a_T \tilde{s}^2 + (2a_T s + b_T)\, \tilde{s}\right) \mathbf{k} - \mu \left(I - \mathbf{k}\mathbf{k}^\top\right) \tilde{\mathbf{v}} \\
\dot{\tilde{\mathbf{r}}} &= \tilde{\omega}_{b/I}^b - \left\lfloor \omega_{b/I}^b \right\rfloor_\times \tilde{\mathbf{r}}_I^b.
\end{aligned} \tag{4.7}$$

We now can define the input to our error state system as

$$\tilde{\mathbf{u}} = [\tilde{s}, \tilde{\omega}]^\top$$

where, as before

$$\tilde{\mathbf{u}} = \check{\mathbf{u}} - \mathbf{u}.$$

In performing optimal control, we will make frequent use of the Jacobians of Equation 4.7 $A = \frac{\partial f}{\partial \tilde{\mathbf{x}}}$ and $B = \frac{\partial f}{\partial \tilde{\mathbf{u}}}$:

$$A = \begin{bmatrix} 0 & R(\mathbf{q})^\top & -R(\mathbf{q})^\top \lfloor \mathbf{v} \rfloor_\times \\ 0 & -\mu\left(I - \mathbf{k}\mathbf{k}^\top\right) & \lfloor R(\mathbf{q})\,\mathbf{g} \rfloor_\times \\ 0 & 0 & -\lfloor \omega \rfloor_\times \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ -\frac{4}{M}\left(2a_T\tilde{s} + 2a_T s + b_T\right)\mathbf{k} & 0 \\ 0 & I \end{bmatrix}.$$

4.2.3 Linear Quadratic Regulator

A linear-quadratic regulator provides the optimal state space controller gains for an LTI system given by

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

assuming full-state feedback. We define the cost-to-go for the infinite-time solution as

$$J(\mathbf{x}, \mathbf{u}) = \int_0^\infty \left(\mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u}\right) dt \tag{4.8}$$

with Q and R being positive definite matrices that define the costs associated with the state and the input. The cost function given in (4.8) is minimized by the control input

$$\mathbf{u} = -K\mathbf{x},$$

where K is given by

$$K = R^{-1}B^\top P,$$

and P is the solution to the continuous-time algebraic Riccati equation (CARE),

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0.$$

It should be noted that in its basic form, the LQR problem attempts to drive the state to zero in an optimal way. If the desire is for the system to reach a desired state, $\check{\mathbf{x}}$, then we use our error state definition

$$\tilde{\mathbf{x}} = \mathbf{x} \boxminus \check{\mathbf{x}}$$

and compute the error state control input

$$\tilde{\mathbf{u}} = -K\tilde{\mathbf{x}},$$

which we either augment with an integrator or by applying a model-based feed-forward control input,

$$\begin{aligned}\mathbf{u} &= \tilde{\mathbf{u}} + \check{\mathbf{u}} \\ &= -K\tilde{\mathbf{x}} + \check{\mathbf{u}}.\end{aligned}$$

4.2.4 Model Predictive Control

Another, more general, form of optimal control is model predictive control, or MPC. In this work we will also implement linear model predictive control (L-MPC) with a finite horizon of N time steps. In this problem we wish to solve

$$\begin{aligned}\min \quad & J = \sum_{i=t}^{N} \tilde{\mathbf{x}}[i]^\top Q\tilde{\mathbf{x}}[i] + \tilde{\mathbf{u}}[i]^\top R[i]\,\tilde{\mathbf{u}}[i] \\ \text{s.t.} \quad & \tilde{\mathbf{x}}[i+1] = \tilde{\mathbf{x}}[i] + dt\left(A|_{\mathbf{x}[t]}\tilde{\mathbf{x}}[i] + B|_{\mathbf{u}[t],\tilde{\mathbf{u}}[t]}\tilde{\mathbf{u}}[i]\right),\end{aligned}$$

$$\begin{bmatrix} 0 \\ -\omega_{\max} \end{bmatrix} < \mathbf{u} < \begin{bmatrix} 1 \\ \omega_{\max} \end{bmatrix}. \tag{4.9}$$

While MPC has greater flexibility than LQR in terms of its constraints, it comes at significant computational cost. Instead of solving the CARE, we must resort to sequential quadratic programming (SQP). This has made achieving real-time performance of MPC a significant challenge in previous work.

Similary to the LQR implementation, solving the MPC problem gives us our error state input $\tilde{\mathbf{u}}$. At each control step, we solve the full MPC problem, but take the error state input for the first time step in the horizon and add our feed-forward input from the trajectory to get our control input that we provide to the flight controller as:

$$\mathbf{u} = \tilde{\mathbf{u}}[t] + \check{\mathbf{u}}.$$

4.3 Implementation and Results

In this section, we report on simulation results of both the LQR and L-MPC formulations. The simulation environment was written in C++, utilized a nonlinear drag model, and the same thrust model used in our dynamics. We used a time step of 0.002 s for the simulation and performed on-manifold RK4 integration between time steps. Additive acceleration noise was applied to the simulated quadrotor to simulate small wind disturbances similar to those observed in real hardware. The simulation study was done on a laptop with an i7-8750H CPU and 16 GB of RAM. The MAV was given 20 seconds to reach each waypoint before the reference trajectory was changed to a new waypoint.

4.3.1 LQR Implementation

The LQR problem is solved at every time step using the closed form, sorted Schur decomposition method [57], and the updated gains are then applied to the current error-state estimate to produce the desired output.

Although we relinearize and solve the CARE at each time step, the Q and R gain matrices are fixed. We choose these gains based on Bryson's rule [58]. As an important note, we achieved the best results by saturating the error-state in accordance with the maximum error terms used to choose the gains with Bryson's rule prior to computing the control as

$$\tilde{\mathbf{u}} = -K\mathrm{sat}\left(\tilde{\mathbf{x}}\right).$$

4.3.2 MPC Implementation

The convex optimization library CVXGEN [59] was used to solve the MPC problem. To strike a balance between computational load and robustness, the optimization time horizon was set to 10 timesteps, and the MPC problem (Equation 4.9) was discretized at 0.05 s intervals, which means that the horizon covered 0.5 s.

4.3.3 Results

Both the LQR and MPC implementations were run in identical simulations and the results of these runs are shown shown in Figures 4.4-4.7. These results show clearly the differences between LQR and MPC. In this simulation, the trajectory is quite aggressive, so despite both controllers being tuned similarly, the LQR solution is not as good at tracking the trajectory as the MPC controller. This is because MPC is better able to handle the saturated inputs and compensate for these saturation limits with other actuators. LQR is not aware of the actuator saturation, so it performs worse when the trajectory has discontinuities. Despite this difference, LQR remains

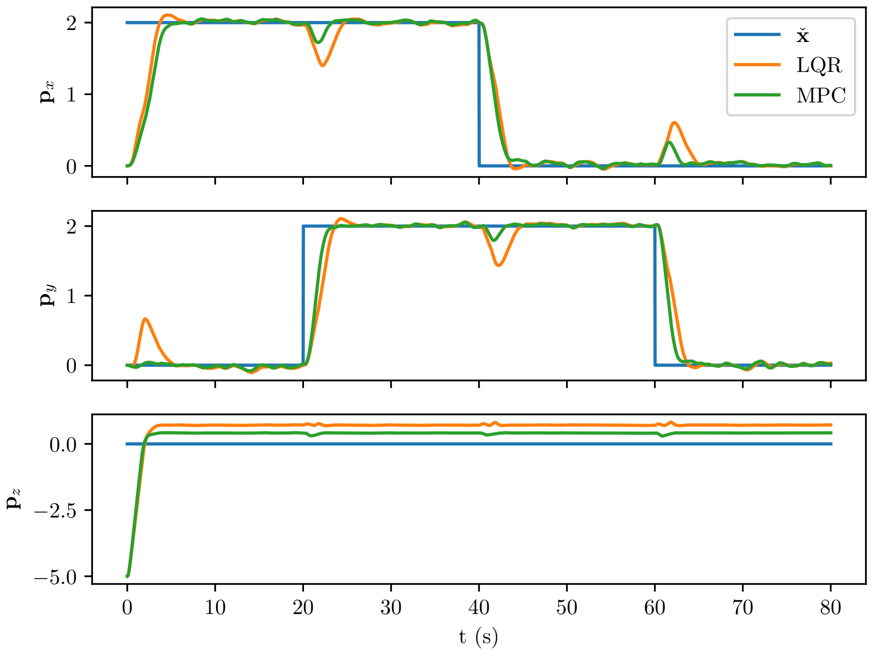


Figure 4.4: Simulated position response for MPC and LQR controllers

stable, and reaches the desired waypoints at about the same time as MPC, just with considerably more overshoot. As one might expect, LQR performs virtually the same as MPC during the steady-state condition after the waypoint has been reached several seconds after a waypoint transition. This happens because LQR and MPC have almost identical cost functions and therefore nearly identical solutions when the inputs are not near saturation.

Another interesting comparison is the time required to compute each algorithm, shown in Figure 4.8. The LQR solution is about twice as fast as MPC to solve, and the maximum time for LQR is also about half the maximum time of MPC. While this may be important for some computationally-constrained platforms, this MPC implementation is still extremely fast, even exceeding some state-of-the-art LQR methods [14]. Based on this performance, we would likely recommend using the MPC configuration because it comes with the better trajectory tracking performance shown in these results and the computational cost, while still double LQR, is almost inconsequential for most modern processors.

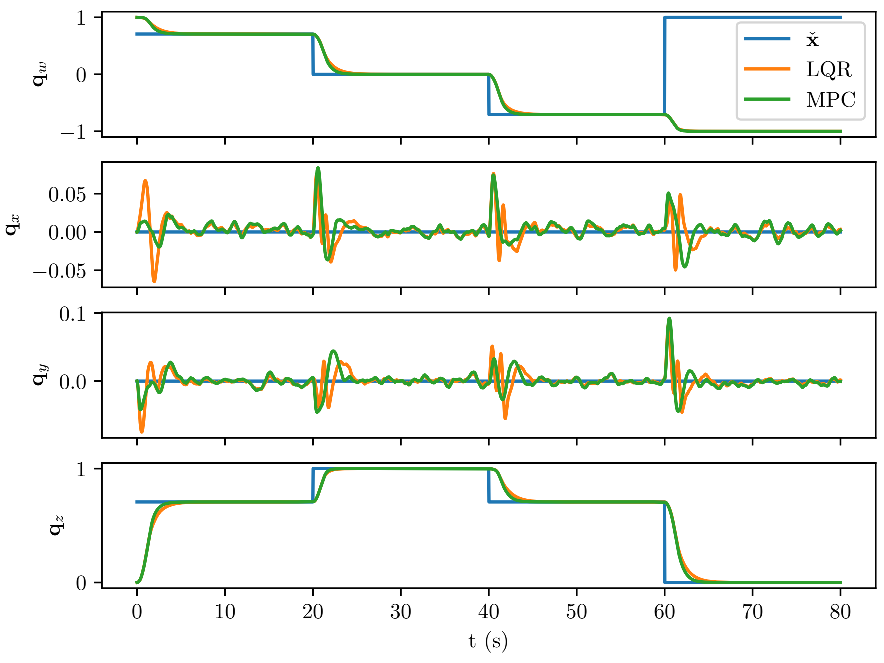


Figure 4.5: Simulated attitude response for MPC and LQR controllers. Note that quaternions provide double coverage of $SO(3)$, therefore the resultant state is actually the same, despite the response converging to $-\check{\mathbf{q}}_w$ at $t > 60$

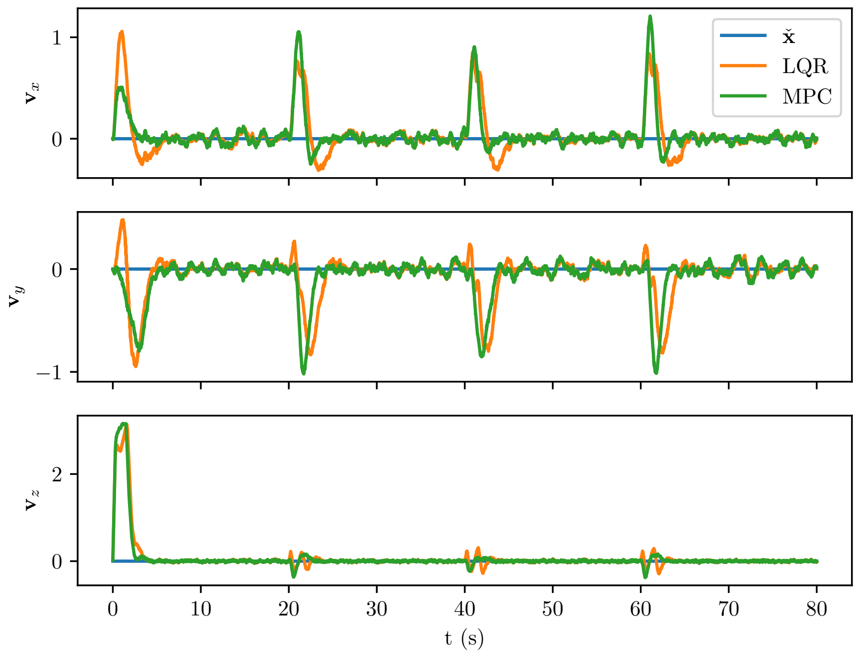


Figure 4.6: Simulated velocity response for MPC and LQR controllers

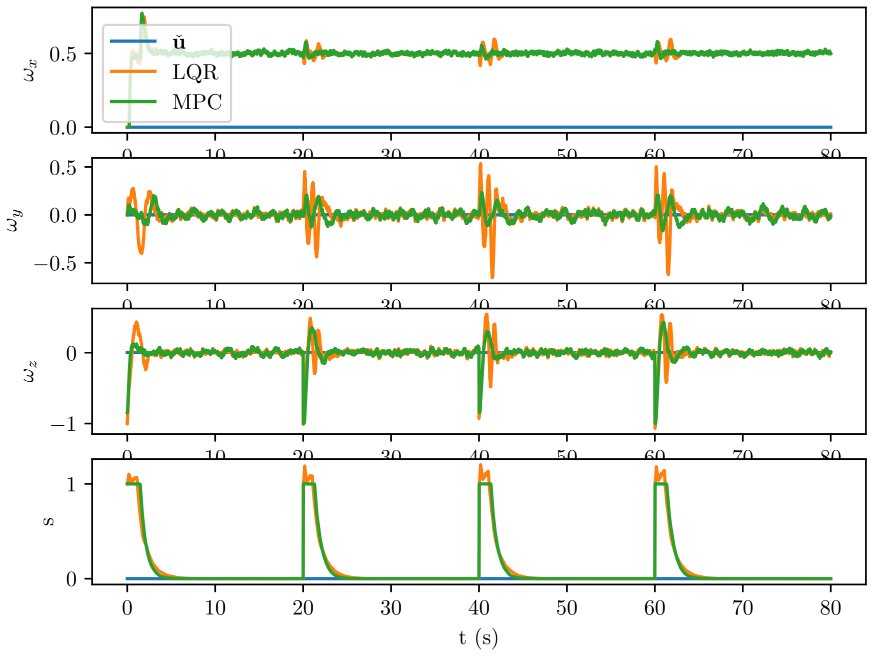


Figure 4.7: Control inputs calculated by LQR and MPC methods

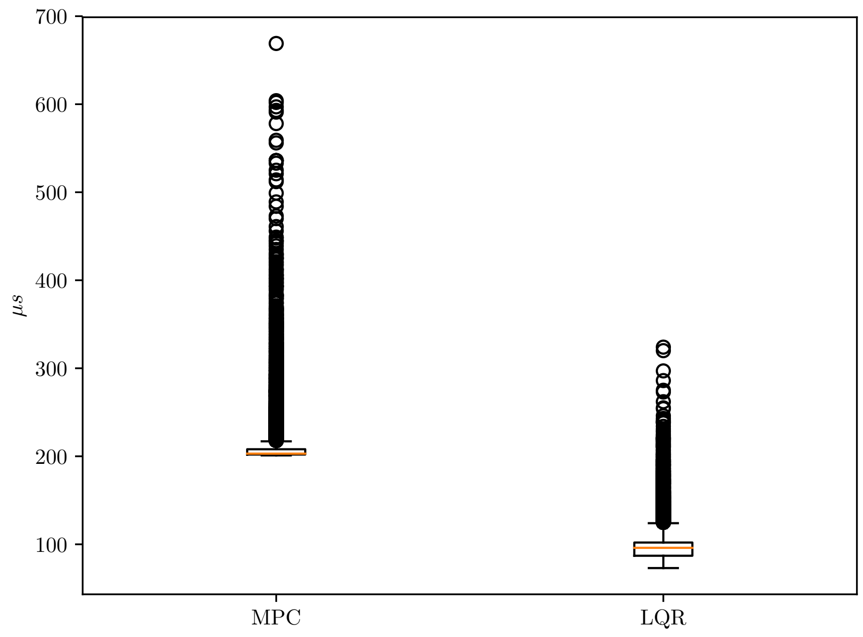


Figure 4.8: Time to complete LQR and MPC solution in simulated trajectory

4.4 Conclusions

This paper derives the error state dynamics for a quadrotor so that we could perform optimial control on the $SO(3)$ manifold. This method is extensible to any reference trajectory and operates at real time on even computationally constrained platforms.

Future work would include generating smooth feasible trajectories (instead of static waypoints) which could include potentially inverted flight sections. This would exercise the strength of the manifold parameterization because there are no singularities, as opposed to the more common Euler-angle based approaches. Much of this work was extended by [13] where the LQR controller developed here was demonstrated in hardware experimentation and shown to meet expectations. The MPC controller has yet to be implemented in hardware and doing this is a clear area of future work.

CHAPTER 5: CUSHIONED EXTENDED-PERIPHERY AVOIDANCE: A REACTIVE OBSTACLE AVOIDANCE PLUGIN[1]

5.1 Introduction

As technological advancements push to meet the size, weight, and power (SWAP) constraints imposed by micro air vehicles (MAVs), exciting applications become possible. Unfortunately the sophistication of estimation and control laws do not yet meet the safety, reliability, and robustness required for full integration into society. One open field of research is autonomous multirotor flight in unknown, dynamic, tightly confined, and cluttered environments.

As illustrated in Figure 5.1, path planning and obstacle avoidance algorithms generally address three objectives: avoiding collisions, facilitating stable flight, and accomplishing a mission or goal. This field of research is well developed, particularly in the context of ground robots. Because a ground robot can generally pause as needed, often the literature assumes a static, known environment. Further, due to the slow, stable dynamics of ground vehicles, disturbances, like wind will rarely induce collisions. These factors, in conjunction with less restrictive weight and computational power constraints, motivate the literature's primary emphasis on the optimal, or at times suboptimal, accomplishment of goals with respect to some specific cost function (item 3 in Figure 5.1).

Generally a global map, represented in a Cartesian coordinate frame, is provided to the path planner. This map comes from a priori data or from fusing sensor information using Simultaneous Localization and Mapping (SLAM) techniques. For example a 2-D obstacle map can be created as

[1]This paper was written by James S. Jackson, David O. Wheeler, and Timothy W. McLain, and published in the International Conference on Unmanned Aircraft Systems in 2016 [60]

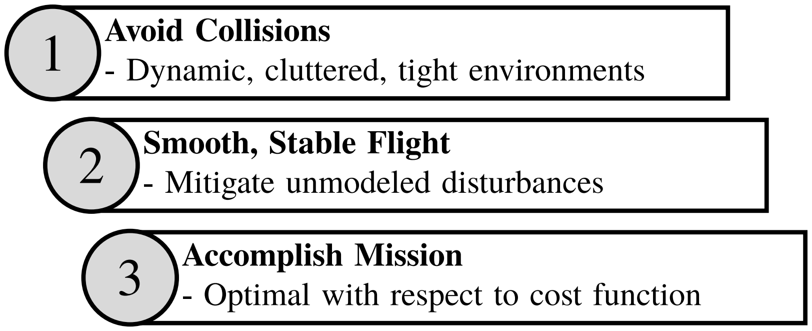

1
Avoid Collisions
- Dynamic, cluttered, tight environments
2
Smooth, Stable Flight
- Mitigate unmodeled disturbances
3
Accomplish Mission
- Optimal with respect to cost function

Figure 5.1: MAV priorities in general. Avoiding collisions, even when they violate environment assumptions, is of paramount importance. Of secondary importance is smooth, stable flight, mitigating destabilizing disturbances. Accomplishing the desired mission should generally not come at the expense of items 1 and 2.

a series of body-fixed, polar laser scans are transformed into a global, Cartesian coordinate frame and fused based on sensor and state uncertainty estimates [61].

Given a map, obstacle-free paths are found through the environment using one of several methods. Potential field methods create artificial forces away from obstacles and towards goals [62]. These methods are generally simple and quick to calculate, but suffer from local minima and cannot guarantee obstacle avoidance. The probability road map (PRM) can be used to randomly generate waypoints connecting the agent with the goal in a manner to avoid obstacles [63], but is designed for use by holonomic agents. Rapidly-exploring random trees (RRT), a modification of PRM uses a similar obstacle-free waypoint path planning technique, while taking into account kinematic constraints of the vehicle. More robust algorithms such as D* Lite [64], can be used to heuristically find the shortest path to the goal through the environment.

While derivatives of these approaches have proven to be effective at fusing sensor measurements and calculating safe paths through the environment, they can incur significant computational, memory, and sensing requirements, and often assume the agent is unaffected by disturbances while safe paths are calculated. While these assumptions may be valid for ground robots and MAVs flying in spacious environments, this problem can become difficult to solve quickly enough to effectively react to large disturbances and errors in environment estimation during autonomous flight in tight quarters.

As an alternative to map-based planning, some simple and efficient algorithms use the concept of optical flow to demonstrate effective corridor-centering [65] and obstacle avoidance [66]. Other, more sophisticated methods use this type of data combined with other monocular features to train agents to avoid obstacles based on input data generated by an expert pilot [67]. These methods have also been demonstrated to be effective in avoiding obstacles during MAV operation but require consistent forward motion to generate meaningful features required by the controller.

In response, we outline the reactive obstacle avoidance plugin (ROAP) framework in Section 5.2 and propose a new reactive algorithm, cushioned extended-periphery avoidance (CEPA) in Section 5.3 as a specific implementation of this framework. We present simulation and hardware results of CEPA and the ROAP architecture in Section 5.4 and conclude in Section 5.5.

5.2 ROAP Motivation

In the ROAP framework, a high-level planner uses any map-based approach to plan smooth paths through a known environment while a reactive obstacle avoidance algorithm is implemented underneath to recover from disturbances or estimation errors, as illustrated in Figure 5.2. In this way, an efficient reactive obstacle avoidance algorithm can match the rate of the sensor with minimal latency, improving robustness in dynamic, cluttered, and tight environments with non-negligible disturbances. This provides the high-level path planner the time to account for changes in the environment, such as a recently closed door or moved obstacle, and plan an alternative feasible path. While a reactive obstacle avoidance plugin may cause the path to become suboptimal in a precarious environment, it requires much less in terms of computational and sensor capabilities, and is effective in real-life testing [68–71].

Clearly, in this configuration, a reactive obstacle avoidance may take action that prevents the completion of a global mission but ensures that the MAV does not damage itself or the environment. This concept parallels the MAVs priorities, illustrated in Figure 5.1, where in general, avoiding collisions and maintaining stable flight is of paramount importance. This is particularly relevant

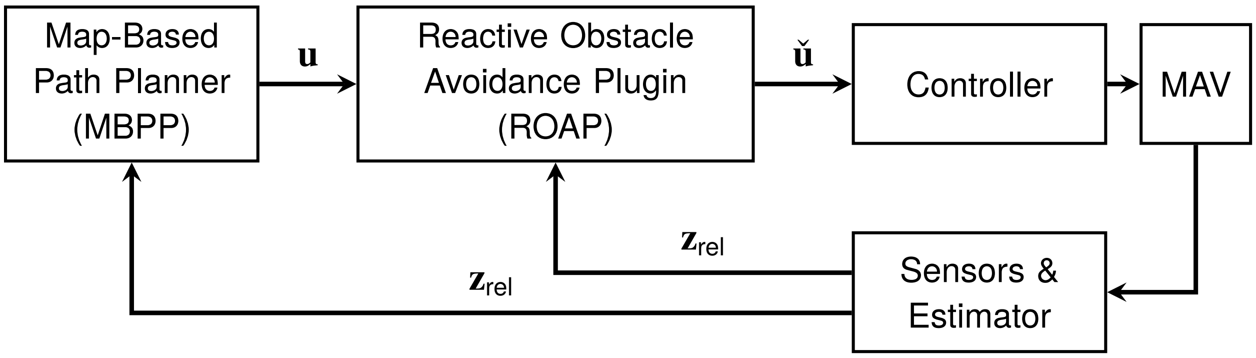

Map-Based Path Planner (MBPP)
$\mathbf{u}$
Reactive Obstacle Avoidance Plugin (ROAP)
$\check{\mathbf{u}}$
Controller
MAV
Sensors & Estimator
$\mathbf{z}_{\text{rel}}$
$\mathbf{z}_{\text{rel}}$

Figure 5.2: Block diagram illustrating how ROAP supplements an existing path planner by modifying commands. The inner control loop rate matches the sensor rate with minimal latency, thereby improving robustness in dynamic, cluttered, and tight environments with non-negligible unmodeled disturbances.

in environments when sensors perform poorly, such as during GPS-degradation or in featureless scenes, and in the presence of disturbances, such as wind or ground and wall effect.

For a ROAP implementation to be robust, the algorithm must exhibit the following properties:

1. Fast response, i.e. low latency, high bandwidth.
2. Independent of a priori or outdated information.
3. Limited memory/computation requirements.
4. No motion assumptions (e.g., constant motion, only forward motion).
5. Safe commands despite erroneous, outdated, or absent high-level goals.

Scherer et al. were first to propose a ROAP algorithm in their paper *flying fast and low among obstacles* (FFLAO) [68] and demonstrated impressive hardware results using a laser scanner. While accounting for the first three properties by responding quickly to the most recent obstacle information, FFLAO constrains the MAV to move only in the direction of the sensor, limiting the MAV to forward and yawing motion alone. While this assumption works under ideal conditions, we have found that this assumption makes safe navigation difficult in tight environments or in the presence of infeasible goals where hovering, reversing and lateral motion are often necessary.

Since FFLAO, Oleynikova et al. has presented a compelling ROAP implementation using stereo vision [69], stressing the importance of low computation requirements. Schopferer et al. has presented a novel decoupled iterative planning method [72] that achieves near-optimal reactive avoidance under computational limitations by considering the kinematic feasibility of planned trajectories. Hrabar presented a method that blurs the line between reactive and map-based obstacle avoidance [71] by keeping a local memory of the environment in the form of a 3D voxel grid and searching for a feasible path using PRM. While the ability to hover is added in this method, it focuses primarily on extending the field-of-view of the sensor, rather than extending the possible maneuvers of the MAV to include lateral and reverse motion. While these methods are all accompanied by impressive results, they are subject to most or all of the same motion constraints found in FFLAO. To address this concern, we present the cushioned extended-periphery avoidance (CEPA) algorithm, which extends these previous methods to allow for safe operation of MAVs in tightly constrained environments in the presence of infeasible goals and non-negligible disturbances.

5.3 CEPA Algorithm Description

The algorithm addresses two main issues related to safe autonomous MAV operation:

1. Guide the MAV around obstacles towards waypoints chosen by the high-level planner.
2. Apply additional control in emergency situations if the MAV comes too close to an obstacle.

Typical path planning approaches use a Cartesian coordinate or graph-based system, either iterating through each coordinate or node to form a cost map [73,74]. CEPA, like FFLAO, performs planning in the polar, body-fixed, sensor frame of the laser scanner. Further, CEPA analytically inflates the proposed path in polar coordinates. As a result, the path can be verified for obstacles by a simple differencing in the polar domain. These two features reduce computational load and algorithm latency.

To remove limiting motion assumptions, CEPA efficiently fuses recent laser scans to create a lower-bound, 360° sensor view. Like [71], this approach blurs the line between a purely reactive avoidance method and a map-based method, which could potentially reduce the reactive nature of the algorithm. However, without a 360° sensor or some level of local memory, necessary lateral or reverse movement cannot be executed safely. A small amount of local memory provides some of the environmental awareness of a map-based planner while maintaining the responsiveness of a reactive planner. CEPA expects velocity commands from a high-level planner and then outputs modified velocity commands, as needed, given input from the most recent laser scans, as shown in Figure 5.2. With this architecture, CEPA can be paired with any high-level path planner which outputs body-fixed velocity commands without modification.

CEPA is derived in two dimensions primarily due to the sensing capabilities of traditional laser scanners. This assumes relatively planar motion in a structured environment, which is often the case for indoor operation of MAVs. To extend CEPA to 3D operations, CEPA could either be layered in cylindrical coordinates or performed entirely in spherical coordinates. Because CEPA leverages the computational benefit of operating directly in the sensor frame, the choice of 3D coordinates should likely mimic the coordinates of the 3D sensor.

5.3.1 Steering Algorithm

The steering algorithm is designed to choose commands that are most like the commands provided by the high-level path planner, but that also safely avoids obstacles. To accomplish this, CEPA computes a cost function which balances modification of an incoming command with proximity to observed obstacles.

First, a suitable path must be in approximately the same direction and approximately the same size as the incoming command when feasible. This can formulated by maximizing the weighted sum of the inner product and the relative size of the goal vector $\mathbf{v}$ and the outgoing command $\check{\mathbf{v}}$,

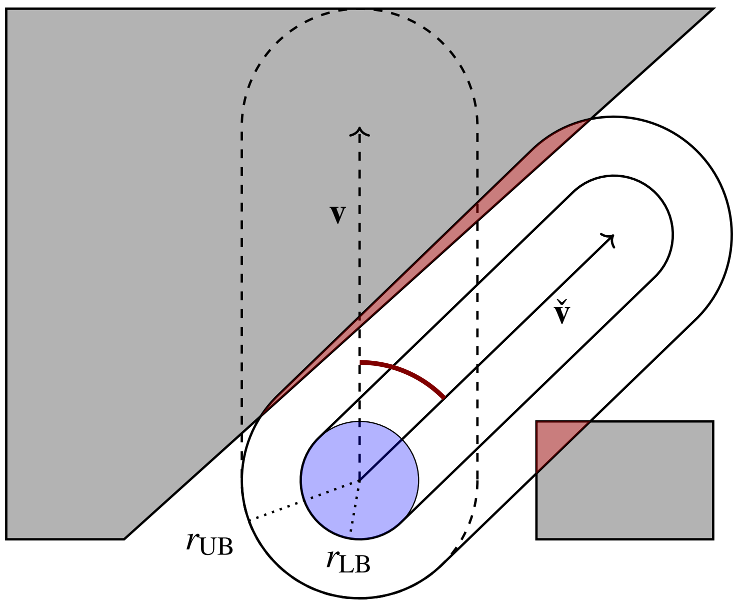


Figure 5.3: An example steering configuration. $\mathbf{v}$ is the obstacle-laden goal vector supplied by the path planner. CEPA identifies $\check{\mathbf{v}}$ as the minimum-cost, collision-free command and passes it to the controller. The heading discrepancy and the obstacle intrusion into the outer safety cushion induce costs shown in red. The proposed path is deemed feasible because the inner safety cushion is not penetrated. While the figure illustrates a Cartesian representation, CEPA works in the sensor's polar coordinate frame.

expressed by

$$k_1 \left(\mathbf{v}^\top \check{\mathbf{v}}\right) + k_2 \frac{||\check{\mathbf{v}}||}{||\mathbf{v}||}. \tag{5.1}$$

Secondly, the degree of interference for the proposed command is calculated by projecting two elongated safety cushions onto the polar map, with fixed look-ahead time T. As illustrated in Figure 5.3, a lower-bound safety cushion of radius r_{LB} defines the minimum required separation distance for a feasible path. An upper-bound safety cushion of radius r_{UB} defines where obstacles begin to influence commands. A safety cushion for a given radius r at specified bearing angle ϕ is defined analytically as

$$SC_r(\phi, \check{\mathbf{v}}) = \begin{cases} r \csc \phi & \phi \in [\gamma, \frac{\pi}{2}) \\ r & \phi \in [\frac{\pi}{2}, \dfrac{3\pi}{2}] \\ -r \csc \phi & \phi \in (\frac{3\pi}{2}, 2\pi - \gamma) \\ d \cos \phi + \sqrt{r^2 - d^2 \sin^2 \phi} & \phi \in [2\pi - \gamma, \gamma) \end{cases}, \tag{5.2}$$

where $d = \|\check{\mathbf{v}}\| \, T$ is the look-ahead distance and $\gamma = \text{atan2}(d, r)$. Note that Equation 5.2 assumes $\check{\mathbf{v}}$ is directed towards $\phi = 0$. Rotating the safety cushion is as simple as shifting the indices of the polar array containing the N returned range measurements.

The lower-bound safety cushion, SC_{LB}, is an estimate of the space the MAV will occupy during the execution of the command for the look-ahead time T. Any conflict with this inner cushion renders the proposed command invalid. The larger cushion, SC_{UB}, acts as a buffer region that may become occupied during the execution of a valid command, but during general operation should remain free. Like a deformable ball, the proposed path will respond to minimize intrusions, guiding the MAV away from obstacles. The extent of the intrusion is found by differencing the safety cushion and laser scan at each angle $LS(\phi_i)$, after masking the array to only regard potential conflicts. A discrete integral can then be used to model the amount of intrusion into the safety bubble for a potential command given a recent laser scan

$$\Omega\left(\check{\mathbf{v}} | LS\right) = \sum_{i}^{N} \kappa\left(\phi_i | \check{\mathbf{v}}, LS\right), \tag{5.3}$$

where

$$\kappa\left(\phi_i | \check{\mathbf{v}}, LS\right) = \begin{cases} \infty & LS(\phi_i) \in [0, SC_{\text{LB}}(\phi_i)] \\ f\left(SC_{\text{UB}}\left(\phi_i\right) - LS\left(\phi_i\right)\right) & LS\left(\phi_i\right) \in \left(SC_{\text{LB}}\left(\phi_i\right), SC_{\text{UB}}\left(\phi_i\right)\right) \\ 0 & LS\left(\phi_i\right) \in \left[SC_{\text{UB}}\left(\phi_i\right), \infty\right) \end{cases}$$

and $f(x)$ is any positive definite function for $x > 0$. In our implementation, $f(x) = x^2$.

A weighted sum of Equations 5.1 and 5.3 forms a cost function whose minimum is the command which is passed to the controller. Using a polar coordinate frame simplifies the cost function sufficiently that even a brute-force method is capable of solving the optimization as fast as the incoming laser scan measurements, typically 10 to 40 Hz:

$$\check{\mathbf{v}}^* = \arg\min_{\check{\mathbf{v}}} \left[k_3 \Omega\left(\check{\mathbf{v}}\right) - k_1 \left(\mathbf{v}^\top \check{\mathbf{v}}\right) - k_2 \frac{\|\check{\mathbf{v}}\|}{\|\mathbf{v}\|} \right].$$

The relative size of gains k_1, k_2, and k_3 can be adjusted for required performance. If k_3 is chosen to be larger than k_1 and k_2, CEPA will prefer to deviate from the planned path to ensure safety. A large k_3 makes the safety cushion inelastic, responding rigidly to approaching obstacles, while a smaller value will provide a softer response. The relative size of k_1 and k_2 will determine how CEPA responds to path deviations. If k_1 is larger than k_2, then CEPA will prefer changing direction to slowing down and vice-versa.

5.3.2 Map Memory

Applying a command in a direction that is not currently observed is inherently presumptuous. Previous ROAP algorithms [68–70] assume that it is always possible to find a viable path while maintaining forward motion. It is not uncommon, however, that a MAV needs to move in a direction in which it is not receiving measurements, such as overshooting a position goal or counteracting a disturbance propelling the vehicle forward. While it is possible to perform large yawing motions to always look in the direction of motion, the control delay makes rejecting disturbances in tight environments impossible.

As an alternative to colliding, some measure of memory must be integrated to ensure that the MAV does not move into objects that it has seen previously, but cannot currently observe with its sensor. This can be done by extending the vehicle's peripheral vision. The reactive planner should not, however, provide a full-resolution map of the explored environment due to computational constraints, but enough to ensure safe navigation.

To do this, some number of previous laser scans and the estimates of the relative transform between each, are saved as a queue in the reactive avoidance memory. In the event that backward motion is necessary, previous laser scans are transformed to be with respect to the current body frame, augmenting the current sensor measurement. If the MAV has moved forward recently, then the concatenation of even two 180 degree laser scans provide some 360° understanding of the environment, as illustrated in Figure 5.4. With this information, the MAV can more confidently execute commands which are not directly in the field of view.

This approach does not extend the field of view of the sensor, but rather assumes, (1) an object has not recently approached the MAV from the rear, and (2) accurate transform estimates are available. For a more conservative memory estimate, the covariance of the transforms can be used to provide the $n\sigma$ worst-case transform. Further, these covariances can be set to grow with time, shrinking the assumed distances to obstacles in the rear 180 degrees. This results in more conservative navigation, but also is more taxing on the processor during memory updates.

5.3.3 Emergency Avoidance

In some cases, a disturbance may cause an obstacle to penetrate the MAVs lower-bound safety threshold r_{LB}. In keeping with the proposed priorities presented in Figure 5.1, the command provided by the map-based path planner is temporarily ignored as emergency action is taken.

As illustrated in Figure 5.5, the periphery-enhanced 360-degree obstacle map is filtered such that

$$\frac{d\rho}{d\phi} \leq K \, ,$$

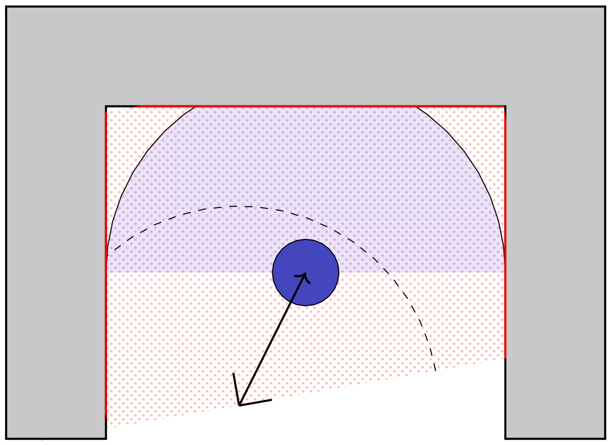


Figure 5.4: A visual description of the way memory is kept in the reactive planner. Although the MAV can only observe obstacles in the direction of the current 180° laser scan (blue-solid), appending previous laser scans gives the MAV a limited 360° understanding about the entire shaded area and allows the MAV to safely move backwards

where K represents the maximum-allowable slope in polar coordinates. For each obstacle detected within r_{LB} a small avoidance vector is formed pointing towards the MAV, proportional to the extent of the intrusion. The summation of these small vectors forms the final command $\check{\mathbf{v}}$. Filtering is critical to ensure that small obstacles are not overpowered by large obstacles in the map. Both small and large obstacles produce commands on similar orders of magnitude given they intrude the same amount into the cushion. In this way, the cushion models the physical response of a deformable ball. With a 360° understanding provided by the map memory, this command can be executed with some level of confidence in any direction.

5.4 Experimentation and Results

CEPA was implemented in ROS [75] and tested in a Gazebo simulator, adapted from [76], and on a hexacopter platform. The simulation parameters paralleled the hardware (3.81 kg, 1.0 m outer diameter). A 40 Hz Hokuyo UTM-30LX laser range finder with a 30 meter range and 180 degree field of view was used for obstacle detection and modeled in the simulator.

A PID velocity controller, using the multirotor model-inversion technique presented in [55] was used to control the system. Yaw was controlled with an under-damped proportional controller,

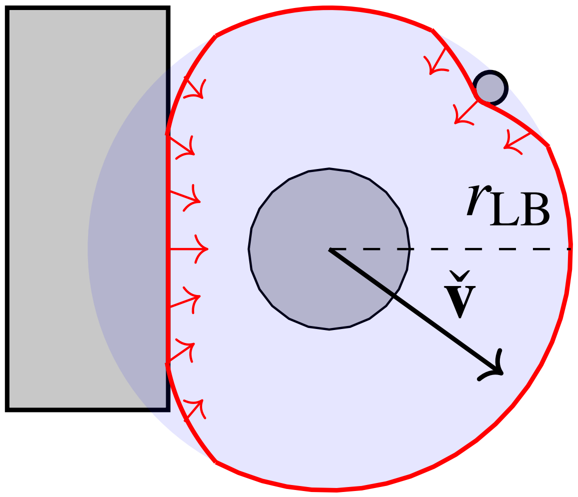


Figure 5.5: Illustration of emergency avoidance. The red line represents the 360° filtered obstacle map when $K = 0.01$. The summation of the individual red avoidance vectors forms the final command $\check{\mathbf{v}}$.

causing the laser scanner to generally be oriented in the direction of commanded motion. The following CEPA gains were used: $k_1 = 1$, $k_2 = 1$, $k_3 = 4$, $T = 4$ s, $K = 0.01$, $r_{\text{LB}} = 0.55$ m, $r_{\text{UB}} = 1.0$ m, and $f(x) = x^2$.

During each simulation experiment, wind was modeled as a succession of applied forces with a normally distributed magnitude, $\mathcal{N}(1N, 0.5N^2)$, and uniformly distributed direction. Wind magnitude and direction were recalculated according to a Poisson process with $\frac{1}{\lambda} = 10$ seconds. These wind model parameters were selected to mimic the significant wall effect that large multirotors experience in tight environments.

FFLAO, defined in [68] was also implemented in 2D for comparison. It was implemented with gains $k_g = 10.5$, $k_o = 0.8$, $c_1 = 1.0$, $c_2 = 0.25$, $c_3 = 1.0$ and $c_4 = 1.0$. It should be noted that this algorithm has demonstrated success in more than 700 flight tests and at speeds exceeding 10 m/s, but due to motion assumptions and constraints it is not designed for operation in tightly confined environments with non-negligible disturbances. It was implemented as a comparison to motivate the relaxation of motion constraints necessary in these types of environments.

Table 5.1: Simulation results for scenario 1.

	FFLAO	CEPA
Completion Rate	0.2188	0.9863
Average Duration (s)	71.61	63.54

5.4.1 Simulation Results

Two tightly-constrained environments were used to validate the algorithm. The first environment, shown in Figure 5.6 consists of a dense grid of cylinders requiring tight maneuvering. While the high-level path planner commands the MAV directly towards the goal, each respective ROAP algorithm modifies the commands to autonomously navigate through the environment. Each algorithm was tested 1500 times. The supplied high-level command had a magnitude between 1.0 m/s and 5.0 m/s and was directed towards the goal. However, regardless of the commanded magnitude, as the multirotor entered the cluttered environment, both CEPA and FFLOA reduced the outgoing command to close to 0.8 m/s to maintain safe flight throughout the course. The collision-free success rate and average flight duration of successful flights taken for the MAV to autonomously navigate safely through the several environments and reach its goal are recorded in Table 5.1.

As can be seen from Table 5.1, placing a constraint on lateral velocity causes performance to suffer in our tightly-confined environment with non-negligible disturbances. This is largely because when moving through such a tightly-confined environment, forward velocity, u, must be kept low. This gives opportunity for disturbances to induce non-negligible lateral velocity which must be corrected in order to avoid collisions. With a constraint on lateral velocity, the MAV is much slower at correcting these errors because it must induce large yawing motions, and therefore is unable to fly safely. CEPA, on the other hand, is able to handle these disturbances because of its ability to move the MAV in any direction to avoid collisions.

The second environment simulates the scenario where a high-level path planner commands an infeasible goal and the obstacle avoidance must prevent the MAV from crashing until a proper goal is received. Specifically, we explored the scenario when a goal is placed on the far side of

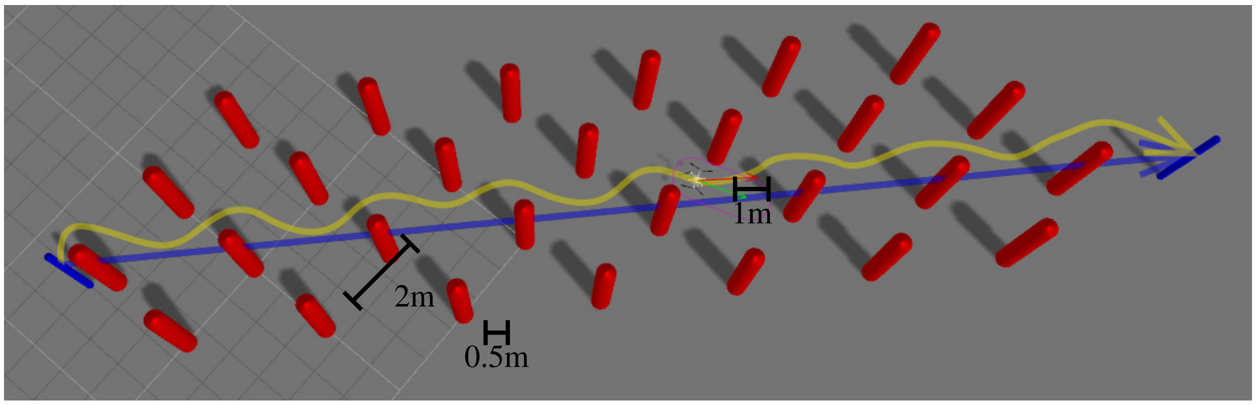


Figure 5.6: Scenario 1: A grid of densely positioned cylinders obstruct the MAV's path between the start and goal positions represented as blue pillars. The high-level path planner commanded a 1m/s velocity directly towards the goal at all times during the test. The blue line is the original infeasible path planned by the high level path planner, while the yellow line is the path ultimately taken by the MAV as a result of CEPA intervention. The red arrow is the current high-level command. The green arrow is the modified CEPA command with the magenta safety cushion shown.

recently closed door, as shown in Figure 5.7. After recognizing the obstruction, the avoidance algorithm was required to correct the commands for 30 seconds until an alternative route was provided. This second scenario was tested 50 times. In each trial, the CEPA algorithm enabled the MAV to successfully pause at the door, accounting for all disturbances while waiting for an updated plan. FFLAO, however, was never able to complete the task because its imposed motion constraints disallowed backward motion. As the MAV approached the closed door, it correctly stopped forward motion, but was unable to correct for any disturbance.

The average latency of CEPA was 2.9 ms with a standard deviation of 1.6 ms. Calculations were easily available at the laser scanner's bandwidth of 40 Hz even using a brute-force optimization method.

5.4.2 Hardware Results

To definitively understand its effectiveness, CEPA was exercised in hardware. Flight test computation was performed using an onboard Intel i7 computer with a 2.4 GHz quad core processor and 16 GB of RAM. To emphasize the light-weight nature of CEPA, avoidance was restricted to use

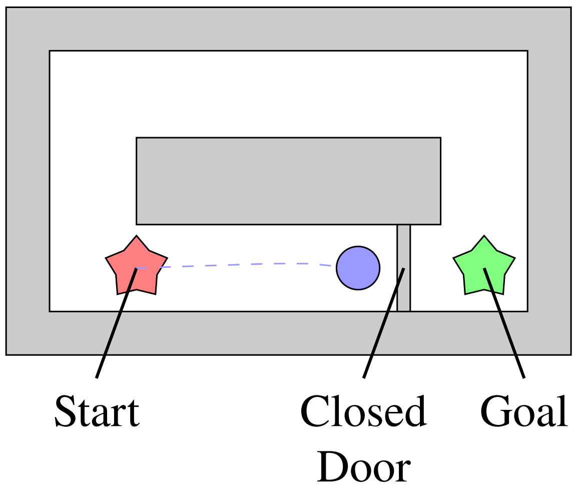


Figure 5.7: Scenario 2: The high-level path planner commands an infeasible path due to a recent environment change. The ROAP block must maintain safety while a new path is planned.

less than 1/16 of the available processing time. State estimation was performed using the relative multiplicative extended Kalman filter described in [77] provided with position measurements from an RGB-D visual odometry algorithm described in [78]. No external positioning system or off-board processing was required.

The MAV was placed in scenarios which isolated three particular challenges:

1. Selecting an appropriate path around several obstacles.
2. Taking action to avoid a previously observed obstacle when is no longer in the field of view.
3. Preventing collision when provided and infeasible goal.

Challenges 1 and 2 were addressed in the first scenario, where the MAV was placed in a wide hallway with two large obstacles in the middle, as shown in Figure 5.8. The high-level path planner continuously provided commands at 0.8 m/s directly towards to the goal, while CEPA correctly chose a safe path around the obstacles and arrived at the goal. During this flight, after navigating around the first obstacle, estimation errors and disturbances caused the MAV to be pushed backwards towards the first obstacle. Although the MAV was oriented towards the goal,

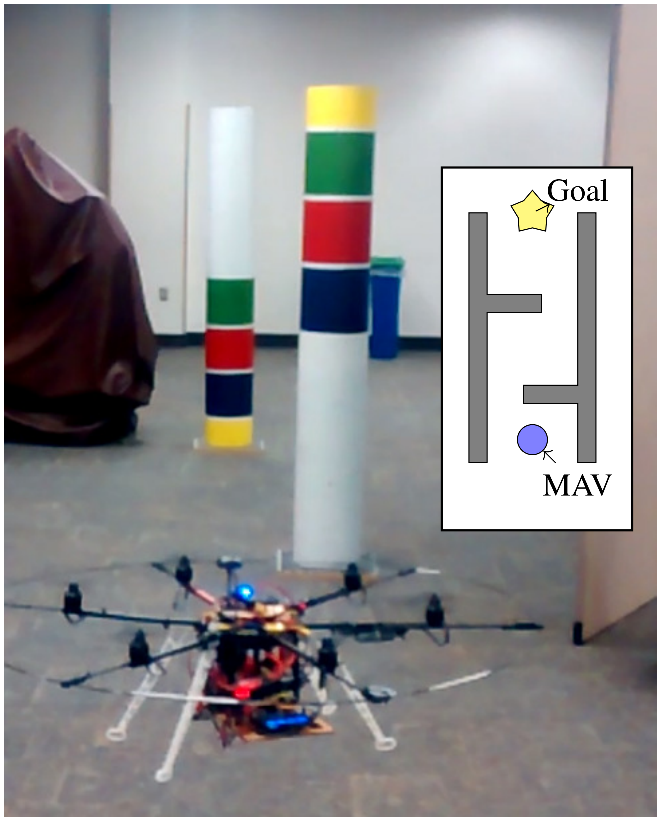


Figure 5.8: Hardware validation of CEPA in a GPS-denied environment using strictly onboard computation and sensing.

and could no longer directly see the first obstacle, it responded correctly by commanding additional control away from the unseen obstacle behind it. After avoiding the first obstacle, the MAV then navigated around the second obstacle and to the goal without further issues. During the test, the MAV maintained a distance of at least 0.1 m from any obstacle, successfully completing the task with no user input.

In the second scenario, the high-level path planner commanded the MAV directly through a flat wall for 5 seconds, very much like the closed-door simulations performed previously. In this demonstration, however, there was no feasible way to reach the goal. During this test, the MAV reached a minimum distance of 0.1 m from the wall, and after some damped oscillatory movement, hovered stably 0.5 m from the wall. Videos of the simulation and hardware demonstrations are available at `https://youtu.be/35Og9PYwXOI`.

5.5 Conclusions

We have outlined the reactive obstacle avoidance plugin framework, which allows for high-bandwidth, low-latency control corrections to improve MAV robustness. This method allows SWAP constrained MAVs to robustly leverage map-based path planners, generally designed for ground robots in static, known environments, while mitigating disturbances and avoiding collisions. To demonstrate the effectiveness of this framework, we have presented the cushioned extended-periphery avoidance algorithm. CEPA relaxes motion assumptions common in other reactive path planners, allowing for more confident control in tight environments with non-negligible disturbances. By working in the laser scanner's polar coordinate frame, and by incorporating previous laser scans, safe controls can be efficiently computed despite erroneous, outdated, or even absent high-level goals.

Future work includes improving the safety cushion lookahead window by incorporating the MAV's dynamics (e.g., momentum) and allowing trajectory based inputs as well as extending CEPA to three dimensions. Developing a fast, camera-based ROAP algorithm without limiting motion assumption remains an open problem. Current work also includes more extensive hardware testing, especially in the presence of moving obstacles.

CHAPTER 6: IMPROVING THE ROBUSTNESS OF VISUAL-INERTIAL EXTENDED KALMAN FILTERING[1]

6.1 Introduction

Visual-inertial (VI) navigation is becoming an increasingly important tool for autonomous operation of miniature aerial vehicles (MAVs) and other robotic agents. While many missions can be performed using GPS or other global measurements to constrain drift, there are numerous scenarios that do not have reliable access to these global measurements. For example, a camera and MEMS IMU can provide a low-cost way to autonomously navigate, and visual camera features provide a method to constrain IMU drift, while also making sensor biases observable for accurate integration.

Recent results in this area have demonstrated remarkable performance and capability [79–84]. While smoothing methods and nonlinear batch optimization-based methods [85–87] have demonstrated significant advantages in terms of accuracy and consistency, they can be too computationally intense for many low-cost platforms. Filtering approaches have the advantage of being computationally efficient but can struggle in certain situations, due to significant nonlinearities and unobservability [6, 7, 51]. This paper discusses filtering techniques for VI estimation that significantly increase robustness to these issues.

One major source of unobservability in VI filtering is the parameterization of feature locations. Feature locations parameterized in an inertial coordinate frame typically assume observability of the transform to that frame. In many situations, however, this transform is unobservable, and estimation becomes inconsistent [6, 7, 51]. Recent methods have shown how to estimate features

[1]This paper was written by James Jackson, Jerel Nielsen, Tim McLain, and Randal Beard. It was and presented at the International Conference of Robotics and Automation (ICRA) in 2019

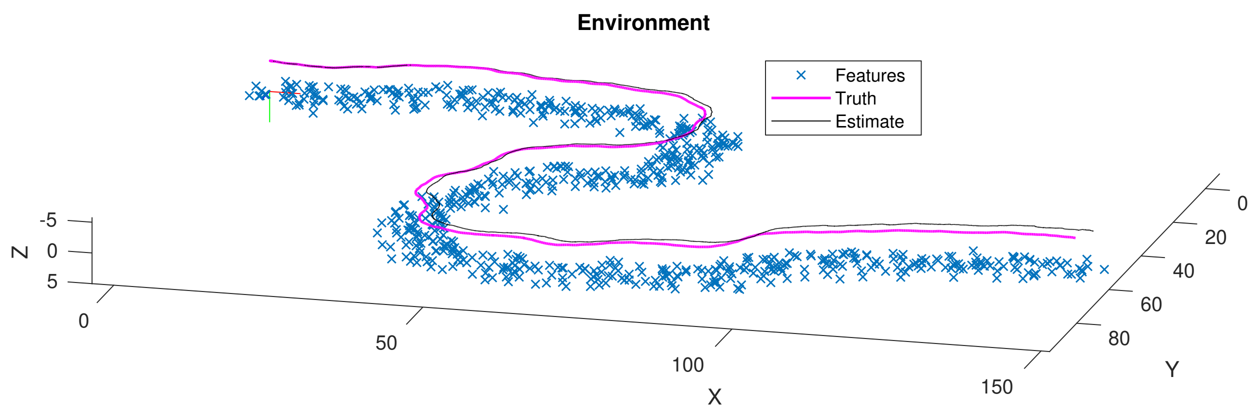


Figure 6.1: A sample trajectory from the Monte Carlo simulation experiment.

in the camera frame, rather than an inertial frame [80]. These parameterizations partition the states cleanly into observable and non-observable states, with global position and heading being completely unobservable. The unobservability of position and heading can be handled using the method proposed by [51], where the position and heading states are periodically reset, so that they remain observable and consistent. The global state and uncertainty are then calculated using other methods such as batch optimization, which are external to the Kalman filter.

Finally, many visual-inertial estimation approaches assume no knowledge about certain aspects of the system dynamics. In some applications, knowledge of specific parts of the system dynamics can help improve estimation accuracy and prevent divergence in certain modes at the expense of becoming less portable to other systems. [50, 88] For example, information regarding the speed capabilities of a multirotor aircraft can bound changes in estimates of depth to visual features. Leishman et al. [50] showed that including a linear model of drag on a multirotor significantly improves estimation accuracy. We will use this model to improve estimator robustness in this work.

Another source of nonlinearity and unobservability is the presence of filter states that are only partially observable or unobservable given specific vehicle motion. Examples of these states include IMU biases, depth to features and the above mentioned linear drag term. Brink [89] has shown that using a partial update can improve filter robustness to these so-called nuisance states, while maintaining consistency.

In this paper, we extend the robocentric visual-inertial Kalman filtering approach described in [80] with the principles of relative navigation described in [90]. We also show that improving the dynamic model can significantly improve estimation accuracy of VI estimation applied to a multirotor and use the partial update formulation to deal with the additional nuisance state used in modeling drag. The paper is organized as follows. In section 6.2, we describe several mathematical concepts and notation used throughout the paper. In section 6.3, we briefly discuss the derivation of our baseline filter [80] with the improved dynamic model. Sections 6.3.2 and 6.3.3 discuss the measurement models used and sections 6.3.4 and 6.3.5 detail the keyframe reset and partial update steps, respectively. Finally, section 6.4 details a Monte Carlo simulation experiment and compares the performance of the proposed improvements in terms of accuracy and consistency.

6.2 Notation

The following definitions are used throughout the paper.

$\mathbf{e}_i$	Unit vector with a one in the i^{th} element
$\mathbf{p}^I_{b/I}$	Position of the body, with respect to the world frame, expressed in the world frame
$\mathbf{v}^b_{b/I}$	Velocity of the body frame, with respect to the world frame, expressed in the body frame
$\mathbf{q}^b_I$	Quaternion describing rotation from the world frame to the body frame
$\boldsymbol{\beta}_a$	Accelerometer bias
$\boldsymbol{\beta}_\omega$	Rate gyro bias
b	Linear drag coefficient
$\boldsymbol{\zeta}^c_{i/c}$	Unit vector directed at the i^{th} feature from the camera origin, expressed in the camera frame
$\mathbf{q}^{\zeta_i}_c$	Quaternion which describes the rotation from the camera $\mathbf{e}_3$ axis to the unit vector $\boldsymbol{\zeta}^c_{i/c}$
ρ_i	Inverse distance to the i^{th} feature

We will also make extensive use of the skew-symmetric matrix operator defined by

$$\mathbf{v}^\wedge \triangleq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix},$$

that is related to the cross-product between two vectors with

$$\mathbf{v} \times \mathbf{w} = \mathbf{v}^{\wedge}\mathbf{w}.$$

To convert back to a vector from a skew-symmetric matrix, we use the $\cdot^{\vee}$ operator, so that

$$\left(\mathbf{v}^{\wedge}\right)^{\vee} = \mathbf{v}.$$

6.2.1 Quaternions

We will use Hamiltonian notation for unit quaternions $\in \mathcal{S}^3$

$$\mathbf{q} = q_0 + q_x\mathbf{e}_1 + q_y\mathbf{e}_2 + q_z\mathbf{e}_3 = \begin{bmatrix} q_0 \\ \bar{\mathbf{q}} \end{bmatrix},$$

which defines the passive rotation matrix based on a unit quaternion as

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0\bar{\mathbf{q}}^{\wedge} + 2\bar{\mathbf{q}}\bar{\mathbf{q}}^{\top} \in SO(3).$$

This definition results in $R_a^b\mathbf{r}^a$ being interpreted as the original vector $\mathbf{r}^a$ expressed in the new coordinate frame b.

The exponential mapping for a unit quaternion is defined as

$$\begin{aligned} \exp : \mathfrak{so}(3)^{\vee} \sim \mathbb{R}^3 &\to \mathcal{S}^3 \\ \exp(\boldsymbol{\delta}) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{\delta}\|}{2}\right) \\ \sin\left(\frac{\|\boldsymbol{\delta}\|}{2}\right)\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|} \end{bmatrix}, \end{aligned}$$

with the corresponding logarithmic map defined as

$$\log : \mathcal{S}^3 \rightarrow \mathfrak{so}(3)^\vee \cong \mathbb{R}^3$$
$$\log(\mathbf{q}) \triangleq 2\ \mathrm{atan2}\left(\|\bar{\mathbf{q}}\|, q_0\right) \frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}.$$

The notion of computing the difference between two group elements leads to defining uncertainty over a member of the Lie manifold. For example, the attitude quaternion $\mathbf{q}_I^b$ has four elements but only three degrees of freedom, so its covariance should be a 3×3 matrix. Using the logarithmic map, we can define the attitude covariance as

$$E\left[\log\left(\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \mathbf{q}_I^b\right) \log\left(\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \mathbf{q}_I^b\right)^\top\right] \in \mathbb{R}^{3\times 3}. \tag{6.1}$$

Eq. (6.1) is significant because the covariance is parameterized in the Lie algebra $\mathfrak{so}(3)$ (which is a vector space) of $SO(3)$ and therefore, can be used in a Kalman filtering framework.

6.2.2 $\boxplus$ and $\boxminus$ operators

Hertzberg et al. [91] describe a new syntax that simplifies working with Lie groups in a filtering and optimization framework by introducing the $\boxplus$ and $\boxminus$ operators. This syntax allows us to work with elements of Lie groups in a notation similar to that of vectors and will be used to describe our filter derivation. The $\boxplus$ and $\boxminus$ operators are defined differently for different groups. For $\mathbb{R}^n$, they are simply defined as the typical addition and subtraction operations. For attitude quaternions $\in \mathcal{S}^3$, these operators are defined by

$$\boxplus : \mathcal{S}^3 \times \mathbb{R}^3 \rightarrow \mathcal{S}^3$$
$$\mathbf{q} \boxplus \boldsymbol{\theta} \triangleq \mathbf{q} \otimes \exp(\boldsymbol{\theta})$$
$$\boxminus : \mathcal{S}^3 \times \mathcal{S}^3 \rightarrow \mathbb{R}^3$$

$$\mathbf{q} \boxminus \mathbf{p} \triangleq \log\left(\mathbf{p}^{-1} \otimes \mathbf{q}\right).$$

One common application of this syntax can be seen below in the discretized quaternion dynamics. With $\boldsymbol{\theta} = \omega_{b/I}^{b}\,dt$, we have

$$\mathbf{q}_I^b\,(t + dt) = \mathbf{q}_I^b\,(t) \boxplus \boldsymbol{\theta}$$
$$\boldsymbol{\theta} = \mathbf{q}_I^b\,(t + dt) \boxminus \mathbf{q}_I^b\,(t).$$

While this syntax is convenient, it is important to note that the dimensionality of $\boldsymbol{\theta}$ and $\mathbf{q}_I^b$ are different in this case. The quaternion is not a vector and has four parameters, while $\boldsymbol{\theta}$ has only three parameters but exists in a vector space.

6.2.3 Feature Bearing Parameterization

As in [80], we parameterize the feature bearing states in the camera frame as rotations $\mathbf{q}_c^{\zeta_i} \in \mathcal{S}^3 \sim R_c^{\zeta_i} \in SO\,(3)$, which describe the rotation from the camera $\mathbf{e}_3$ axis to the unit vector directed at the feature. The unit vector directed at feature i with respect to the camera frame c is then defined by

$$\zeta_{i/c}^{c} = \left(R_c^{\zeta_i}\right)^{\top} \mathbf{e}_3 \in \mathcal{S}^2 \subset \mathbb{R}^3,$$

where we can see that this simply expresses the direction of the feature in the camera frame.

The difference between two unit vectors $\zeta_i \boxminus \zeta_j$ can be described using axis-angle representation, where the direction of the axis of rotation is orthogonal to both of the unit vectors, and its length is scaled by the magnitude of rotation, as shown in Figure 6.2. There are actually only two degrees of freedom in this parameterization because rotation about either feature vector does not change unit vector direction. To remove the redundant degree of freedom, we note that the axis of shortest rotation is always in the plane normal to $\boldsymbol{\zeta}_{i/c}^{c}$ and define a projection matrix

$$T_{\zeta_i} = \left(R_c^{\zeta_i}\right)^\top [\mathbf{e}_1 \quad \mathbf{e}_2] \in \mathbb{R}^{3\times 2},$$

which reduces the dimensionality of the axis-angle representation to this plane. It can be seen that this projection matrix is just the two basis vectors orthogonal to feature direction, defined in the camera reference frame.

We must then define the $\boxplus$ and $\boxminus$ operators associated with feature bearing vectors as

$$\begin{aligned}
\boxplus : & SO(3) \times \mathbb{R}^2 \to SO(3) \\
& \mathbf{q}_c^{\zeta} \boxplus \boldsymbol{\delta} \triangleq \exp(T_\zeta \boldsymbol{\delta}) \otimes \mathbf{q}_c^{\zeta} \\
\boxminus : & SO(3) \times SO(3) \to \mathbb{R}^2 \\
& \mathbf{q}_c^{\zeta_j} \boxminus \mathbf{q}_c^{\zeta_i} \triangleq \theta T_{\zeta_i}^\top \mathbf{s},
\end{aligned}$$

where the axis $\mathbf{s}$ and angle θ between the two feature direction vectors are given by

$$\begin{aligned}
\theta &= \cos^{-1}\left(\boldsymbol{\zeta}_i^\top \boldsymbol{\zeta}_j\right) \\
\mathbf{s} &= \frac{\boldsymbol{\zeta}_i \times \boldsymbol{\zeta}_j}{\|\boldsymbol{\zeta}_i \times \boldsymbol{\zeta}_j\|}.
\end{aligned}$$

With only two degrees of freedom, and with all feature vectors referencing the camera $\mathbf{e}_3$ axis, there are an infinite number of unit quaternions which can be used to represent the same unit vector. The difference between these rotations is some angle of rotation about the bearing vector itself. This is removed by the projection operation and can therefore be neglected. Reference [92] explores more deeply the validity of the $\boxplus$ and $\boxminus$ operators under this assumption.

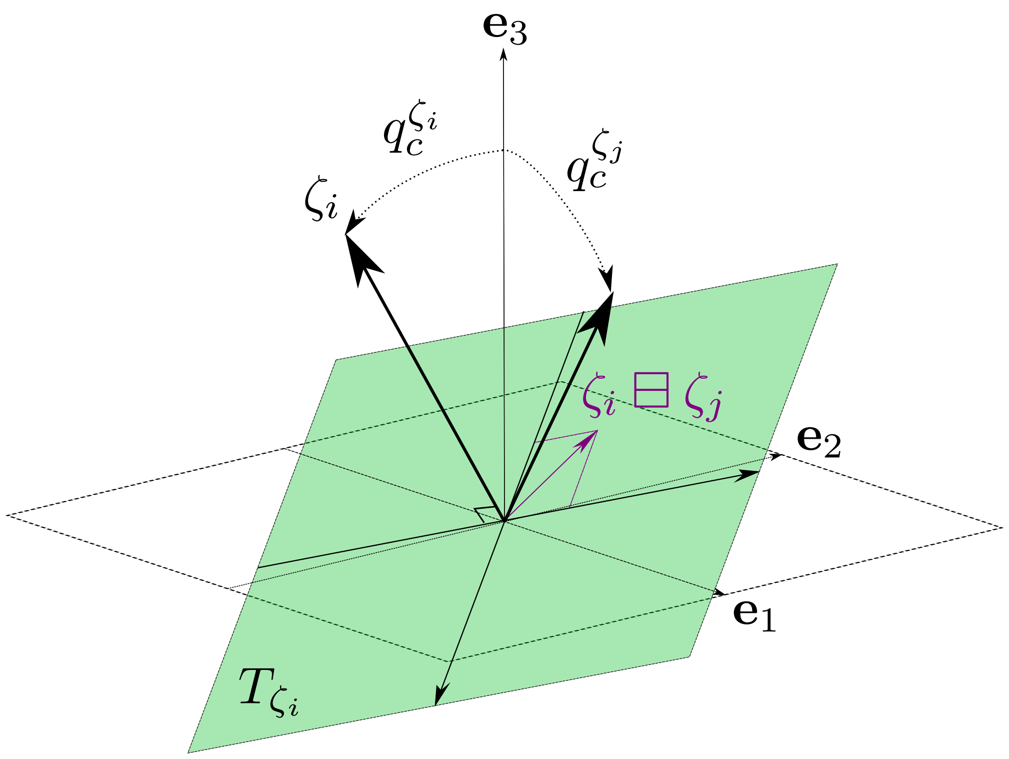


Figure 6.2: Illustration of feature bearing vector geometry.

6.3 Derivation

In this section, we derive the relevant geometry and dynamics to fully describe and implement the filter proposed in this paper.

6.3.1 State Definition and Kinematics

Let the state $\mathbf{x} \in \mathbb{R}^6 \times \mathcal{S}^3 \times \mathbb{R}^7 \times \mathcal{S}^3 \times \mathbb{R} \times \cdots \times \mathcal{S}^3 \times \mathbb{R}$ be defined by

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_{b/I}^I & \mathbf{v}_{b/I}^b & \mathbf{q}_I^b & \boldsymbol{\beta}_a & \boldsymbol{\beta}_\omega & b & \mathbf{q}_c^{\zeta_1} & \rho_1 & \cdots & \mathbf{q}_c^{\zeta_n} & \rho_n \end{bmatrix},$$

with n tracked features. The corresponding covariance matrix P is then defined as

$$P = E\left[(\mathbf{x} \boxminus \hat{\mathbf{x}})(\mathbf{x} \boxminus \hat{\mathbf{x}})^\top\right] \in \mathbb{R}^{(16+3n)\times(16+3n)},$$

where $\boxminus$ for objects composed of multiple group elements implies the use of the appropriate $\boxminus$ operator for each element.

Given measured acceleration $\bar{\mathbf{a}}_{b/I}^{b}$ and measured angular velocity $\bar{\omega}_{b/I}^{b}$, the state has kinematics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u} + \boldsymbol{\eta})$ with the elements of f given by [50] and defined as

$$\begin{aligned}
\dot{\mathbf{p}}_{b/I}^{I} &= \left(R_I^b\right)^\top \mathbf{v}_{b/I}^{b} \\
\dot{\mathbf{v}}_{b/I}^{b} &= \mathbf{e}_3\mathbf{e}_3^\top \mathbf{a}_{b/I}^{b} + R_I^b \mathbf{g}^I - b\left(I - \mathbf{e}_3\mathbf{e}_3^\top\right)\mathbf{v}_{b/I}^{b} - \left(\boldsymbol{\omega}_{b/I}^{b}\right)^\wedge \mathbf{v}_{b/I}^{b} \\
\dot{\mathbf{q}}_I^b &= \boldsymbol{\omega}_{b/I}^{b} \\
\dot{\boldsymbol{\beta}}_a &= \mathbf{0} \\
\dot{\boldsymbol{\beta}}_\omega &= \mathbf{0} \\
\dot{b} &= \mathbf{0} \\
\dot{\mathbf{q}}_c^{\zeta_i} &= -T_{\zeta_i}^\top \left(\boldsymbol{\omega}_{c/I}^{c} + \rho_i \left(\boldsymbol{\zeta}_{i/c}^{c}\right)^\wedge \mathbf{v}_{c/I}^{c}\right) \\
\dot{\rho}_i &= \rho_i^2 \left(\boldsymbol{\zeta}_{i/c}^{c}\right)^\top \mathbf{v}_{c/I}^{c},
\end{aligned} \tag{6.2}$$

where b is a linear drag term [50], $\mathbf{u} = \begin{bmatrix}\mathbf{a}_{b/I}^{b} & \boldsymbol{\omega}_{b/I}^{b}\end{bmatrix}$ is the input, $\boldsymbol{\eta} = \begin{bmatrix}\boldsymbol{\eta}_a & \boldsymbol{\eta}_\omega\end{bmatrix}$ is input noise, and

$$\begin{aligned}
\mathbf{a}_{b/I}^{b} &= \bar{\mathbf{a}}_{b/I}^{b} - \boldsymbol{\beta}_a - \boldsymbol{\eta}_a \\
\boldsymbol{\omega}_{b/I}^{b} &= \bar{\boldsymbol{\omega}}_{b/I}^{b} - \boldsymbol{\beta}_\omega - \boldsymbol{\eta}_\omega.
\end{aligned}$$

Camera linear and angular velocities are also given by

$$\begin{aligned}
\mathbf{v}_{c/I}^{c} &= R_b^c\left(\mathbf{v}_{b/I}^{b} + \left(\boldsymbol{\omega}_{b/I}^{b}\right)^\wedge \mathbf{p}_{c/b}^{b}\right) \\
\boldsymbol{\omega}_{c/I}^{c} &= R_b^c \boldsymbol{\omega}_{b/I}^{b},
\end{aligned}$$

where R_b^c is the fixed rotation from body to camera frame and $\mathbf{p}_{c/b}^{b}$ is the fixed translation from body to camera in the body frame.

In the proposed filter, we employ the typical continuous-discrete Extended Kalman Filter (EKF) equations. However, the use of $\boxplus$ and $\boxminus$ operators requires a slightly different treatment of the propagation and update equations. We propagate the filter forward in time and apply discrete updates according to

$$\hat{\mathbf{x}}(t+dt) = \hat{\mathbf{x}}(t) \boxplus f(\hat{\mathbf{x}}(t), \mathbf{u}(t))\,dt$$
$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}} \boxplus K(\mathbf{z} \boxminus h(\hat{\mathbf{x}})),$$

where K is the Kalman gain, $\mathbf{z}$ is a measurement, and $h(\hat{\mathbf{x}})$ is a measurement model.

6.3.2 Camera Measurement Model

Given a pixel measurement (u, v), pixel location of the camera's optical axis (u_0, v_0), camera focal lengths (f_x, f_y), and relative landmark location in the camera frame, the pin-hole camera model may be written in terms of $\mathbf{x}$ as

$$h_{cam}(\mathbf{x}) = \frac{1}{\mathbf{e}_3^\top \boldsymbol{\zeta}^c} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix} \boldsymbol{\zeta}^c + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}. \tag{6.3}$$

The Jacobian $\partial h_{cam}/\partial \mathbf{x}$ of the camera measurement model is given by

$$H_{cam} = [\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad 0 \quad H_1 \quad 0 \quad \cdots \quad H_n \quad 0],$$

where using the chain rule, we have

$$H_i = \frac{1}{\mathbf{e}_3^\top \boldsymbol{\zeta}_{i/c}^c} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix} \left(\frac{\boldsymbol{\zeta}_{i/c}^c \mathbf{e}_3^\top}{\mathbf{e}_3^\top \boldsymbol{\zeta}_{i/c}^c} - I_{3\times 3} \right) \left(\boldsymbol{\zeta}_{i/c}^c \right)^\wedge T_{\zeta_i}.$$

6.3.3 Accelerometer Measurement Model

Using the multirotor drag model from [50] in (6.2) provides the benefit that velocity becomes directly observed by the accelerometer (assuming a linear drag constant). It is assumed that the accelerometer measures total acceleration of the body, neglecting gravity, in addition to a constant bias $\boldsymbol{\beta}_a$ and zero-mean white noise $\boldsymbol{\eta}_a$. If we also assume that thrust T acts only along the body $\mathbf{e}_3$ axis, we can consider just the body $\mathbf{e}_1$ and $\mathbf{e}_2$ axes, removing any dependence of the measurement on T. The measurement model is then given by

$$h_{acc}(\mathbf{x}) = I_{2\times 3}\left(-b\mathbf{v}_{b/I}^{b} + \boldsymbol{\beta}_a + \boldsymbol{\eta}_a\right). \tag{6.4}$$

The Jacobian $\partial h_{acc}/\partial \mathbf{x}$ is given by

$$H_{acc} = \begin{bmatrix} \mathbf{0} & -bI_{2\times 3} & \mathbf{0} & I_{2\times 3} & \mathbf{0} & -I_{2\times 3}\mathbf{v}_{b/I}^{b} & \mathbf{0} & \cdots \end{bmatrix}.$$

6.3.4 Keyframe Reset

As shown in [51] and [90], performing a keyframe reset when global states are unobservable can dramatically improve filter consistency and accuracy. A keyframe reset is performed by resetting the global position and heading states to zero and updating the covariance matrix appropriately. Each reset step results in a new *node* being declared in a pose graph structure, which can then incorporate loop closures and other measurements as part of a global optimization routine. Figure 6.3 shows an illustration of the coordinate frames involved in the keyframe reset. Here, we note that our setup slightly differs from [51] and [90] in that there is no altimeter measurement available, so altitude is also unobservable, and we must also reset that state. Therefore, we can see in Figure 6.3 that node frames are co-located with keyframes, instead of on a ground plane.

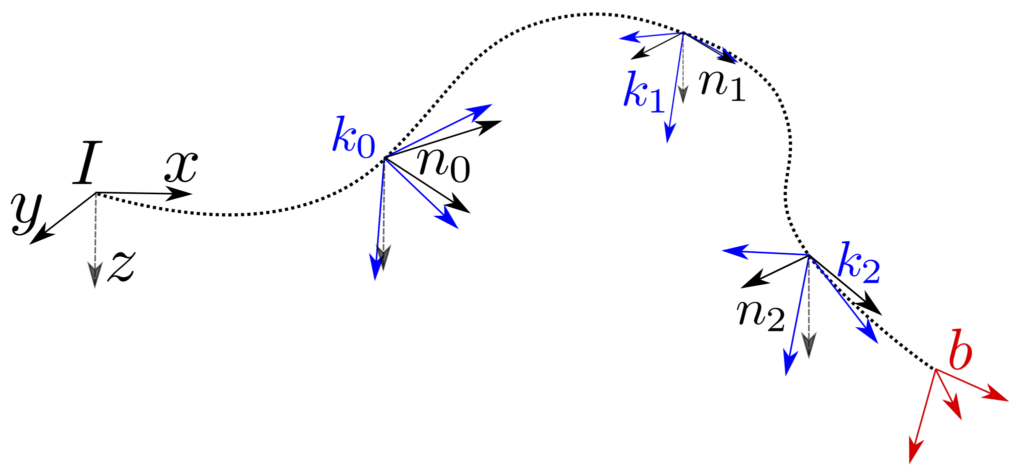


Figure 6.3: Keyframes k_i are declared at periodic intervals along the trajectory flown by the MAV, while node frames n_i are associated with each keyframe and are gravity-aligned but co-located with each keyframe. The current body frame b is estimated with respect to the most recent keyframe. New keyframes are declared when less than 25 percent of the features present in the previous keyframe are still present. This promotes observability of the transform between b and the most recent keyframe.

6.3.5 Partial Update

A common difficulty faced in visual-inertial navigation is the estimation of nuisance states which may only be partially observable during many maneuvers. In the filter derived in this paper, these states include the inverse depth to each feature ρ_i, accelerometer and gyro biases $\boldsymbol{\beta}_a$ and $\boldsymbol{\beta}_\omega$, and the linear drag term b. As noted in [89], estimating these terms in the traditional manner can cause filter divergence but ignoring them or considering them as known constants may produce an overconfident estimate. Because of the abundance of these states in our system, we employ a version of the partial-update Schmidt-Kalman filter proposed by [89]. This method allows the designer to tune the effect of a measurement update on the i^{th} state with a scalar gain γ_i, while correctly estimating uncertainty in these partially-updated states. While this method loses optimality guarantees in estimating these states in a linear Kalman-filtering framework, it has been shown to speed up convergence of these nuisance states by limiting the effect of linearization errors when applied to the non-linear IMU-camera extrinsics estimation problem [89].

A drawback of the formulation given in [89] is the intermediate calculation of $\hat{\mathbf{x}}^+$and P^+. We can manipulate these equations to remove this intermediate calculation and maintain algebraic equivalence. Let us first define $\lambda_i = 1 - \gamma_i$, and for N states, also define

$$\boldsymbol{\lambda} = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_N],$$

which contains our tuning parameters. The values in this vector range from zero to one with ones indicating a full update to those particular states. The state and covariance updates may now be given by

$$\begin{aligned}
\hat{\mathbf{x}}^{++} &= \hat{\mathbf{x}}^{-} \boxplus \left(\boldsymbol{\lambda} \odot K\left(\mathbf{z} \boxminus \mathbf{h}\left(\hat{\mathbf{x}}^{-}\right)\right)\right) \\
P^{++} &= P^{-} + \Lambda \odot \left(\left(I - KH\right) P^{-} \left(I - KH\right)^{\top} + KRK^{\top} - P^{-}\right),
\end{aligned}$$

where we've employed the numerically stable Joseph form of the covariance update, $\odot$ is the Hadamard product, and

$$\begin{aligned}
\mathbf{1} &= [1 \quad 1 \quad \cdots \quad 1]^{\top} \\
\Lambda &= \mathbf{1}\boldsymbol{\lambda}^{\top} + \boldsymbol{\lambda}\mathbf{1}^{\top} - \boldsymbol{\lambda}\boldsymbol{\lambda}^{\top}.
\end{aligned}$$

6.4 Results

To identify improvements to consistency and accuracy, we employed a Monte Carlo (MC) simulation of a MAV with a nonlinear aerodynamic model. The multirotor was commanded to fly approximately five meters above a simulated ground plane at a constant forward velocity of one meter per second. The commanded heading for each iteration evolved according to a random walk. A fourth-order Runge Kutta integration scheme was used for the truth comparison. A sample trajectory is shown in Figure 6.1.

Camera measurements consisted of static landmarks projected onto a simulated image plane via the pin-hole camera model and were corrupted by a small amount of white noise. Landmarks were chosen by randomly selecting enough features in the camera's field of view to fill the state vector. These same features were then selected in subsequent time steps until they left the camera's field of view, at which point another landmark was randomly generated in the field of view. This removes any dependence on a feature tracker in the MC simulation and results in ideal performance because there are no data association errors. However, this approach is appropriate for filter comparisons in an MC simulation because we wish to identify differences in filter performance under ideal conditions. Accelerometer and gyro measurements were corrupted with Gauassian noise and slowly varying biases similar to the observed noise in hardware experiments.

We implemented four different filters for comparison. The *baseline* (BL) filter is the same filter derived in [80] except with the measurement model for features given as (6.3) rather than the patch-based model in the original work. This was primarily done to simplify modeling in the simulation environment and to guarantee that all filters received the same measurements. The second filter modifies the baseline with a linear drag term (DT) as shown in (6.4), while the third filter modifies the baseline with keyframe resets (KF) given in Section 6.3.4. The fourth filter augments the baseline with a drag term, keyframe reset, and a partial update (KF+DT+PU). Each of these filters were given identical inputs and measurements for each MC iteration, and the relevant process and sensor noise covariance matrices used in each filter were derived from the corresponding simulation parameters.

Inverse depth to each feature was initialized using the recommended values in [93] of $\rho_0 = {}^1\!/_{2d_{min}}$ and $R_0 = {}^1\!/_{16d_{min}}$ with a minimum distance to each feature assumed to be $d_{min} = 2$ meters. To deal with negative depth estimates, we used the method in [94], where any negative depth estimates were immediately re-initialized to d_{min} and the covariance appropriately expanded to account for the additional uncertainty. Because keyframes are not tied to a specific image in this estimator (as opposed to the implementation in [90]) new keyframes were declared when more than one half of the features present at the declaration of the previous keyframe were lost.

Absolute accuracy of each filter was compared using the root mean squared error (RMSE) of the position and attitude states. Because the filters with a keyframe reset step estimate this transform with respect to a local keyframe, each time a new keyframe was declared, (or each time a new node was created) both the true state $\mathbf{x}^n$ and the estimated state $\hat{\mathbf{x}}^n$ of each filter were saved, even in the filters with no keyframe reset step. We then calculated the RMSE of the estimated relative transform (position and attitude) between the previously declared node frame and the current body frame T_n^b for each filter

$$\begin{aligned} J_{RMS} &= \left\| \hat{T}_n^b \boxminus T_n^b \right\| \\ &= \left\| \begin{bmatrix} \hat{\mathbf{p}}_{b/n}^n - \mathbf{p}_{b/n}^n \\ \hat{\mathbf{q}}_n^b \boxminus \mathbf{q}_n^b \end{bmatrix} \right\| . \end{aligned}$$

This method not only ensures that we perform a fair comparison between filters, but it also ensures that the sometimes large heading errors accumulated before accelerometer and gyroscope bias measurements converge do not confound RMSE calculations later on in the trajectory.

Filter consistency was analyzed using normalized estimator error squared (NEES) or the Mahalanobis distance of the position and attitude states. Because NEES is weighted by the current covariance matrix of each estimator, the NEES of a filter with a keyframe reset is calculated with respect to relative pose, while the NEES of a filter without a keyframe reset is calculated with respect to global pose. Therefore, NEES is calculated according to

$$\epsilon = \left\{ \begin{array}{ll} \left(\hat{T}_n^b \boxminus T_n^b\right)^\top P_{T_n^b} \left(\hat{T}_n^b \boxminus T_n^b\right) & \text{if KF} \\ \left(\hat{T}_I^b \boxminus T_I^b\right)^\top P_{T_I^b} \left(\hat{T}_I^b \boxminus T_I^b\right) & \text{otherwise} \end{array} \right\} .$$

Because NEES is calculated over the transform states with 6-DOF (position and attitude), a histogram of the NEES of an ideal filter should fit a χ^2 distribution with six degrees of freedom and remain constant over time.

We performed 2016 MC iterations of a five-minute simulation study and calculated the RMSE and NEES at each time step (250 Hz). The average RMSE and NEES over time for each filter in the MC simulation study are shown in Figure 6.6. In this plot, we see that the RMSE of each filter decreases as each filter evolves in time and converges on the unknown biases. A histogram of the RMSE and NEES for each estimator at the final time is given in Figure 6.7.

It is clear from the results of this study that using keyframe resets dramatically affects RMSE and NEES, resulting more accurate and consistent pose estimates. In filters without a keyframe reset step, the unobservable position and heading states cause the filter to become increasingly inconsistent over time, resulting in large linearization errors and suboptimal sensor fusion [51].

It appears that while the drag term improves pose accuracy, it degrades consistency. This is not altogether unexepected as the drag term is only partially observable and the resulting linearization error on the drag term measurement update (6.4) causes the filter to become overconfident. The improved accuracy, however comes from better state integration which arises from the improved dynamic model.

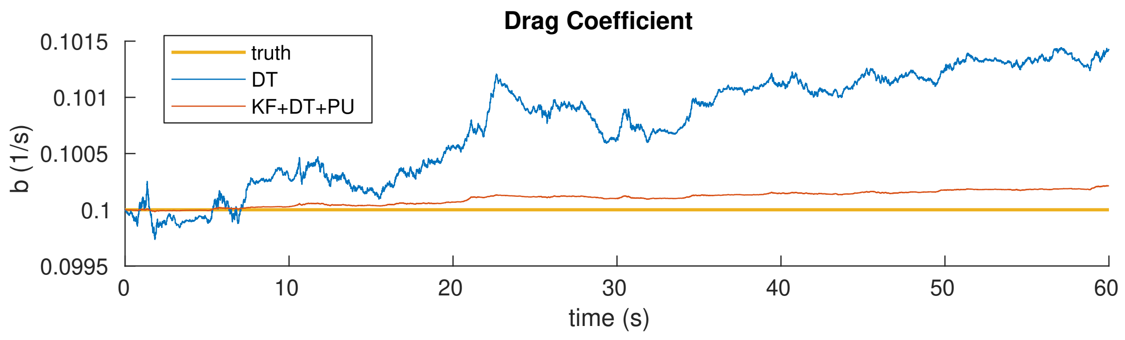


Figure 6.4: Drag term estimates of a single MC iteration with and without the partial update

The overconfidence caused by the drag term can be mitigated by using a partial update. In the (KF+DT+PU) filter, γ_b was set to 0.02, which reduced the effect of linearization error on the state and covariance. Figure 6.4, shows a single run of the drag term with and without the partial update. In this plot, the drag term without the partial update produces oscillations corresponding to changes in attitude. This is most certainly incorrect as we have no reason to believe that the constant drag

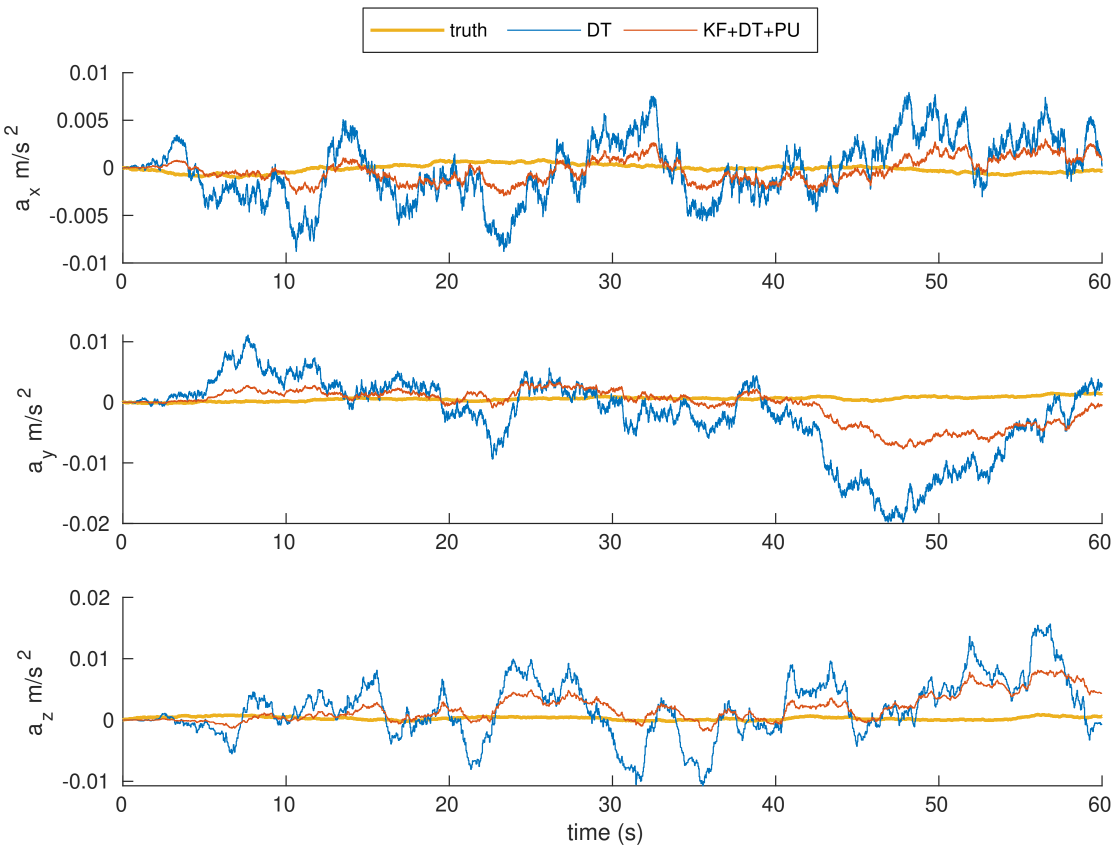


Figure 6.5: Accelerometer biases of a single MC iteration with and without the partial update

term should be correlated with attitude. The partial update attenuates these oscillations and allows us to benefit from the improved dynamic model. A similar effect is observed in accelerometer and gyroscope bias estimates. We see in Figure 6.5, that without the drag term, acclerometer bias estimates become strongly correlated with attitude. Again, the partial update damps this oscillatory response and keeps the estimate more aligned with truth.

6.5 Conclusions

We have shown that augmenting visual-inertial extended Kalman filtering with keyframe resets, an improved dynamic model, and partial updates greatly improves accuracy and consistency in VI filtering. This is clearly demonstrated in Figures 6.6 and 6.7. The use of keyframe resets improves filter consistency and accuracy without any observed negative consequences. Augmenting the dynamic model with a linear drag term also improves accuracy but at the expense of degraded

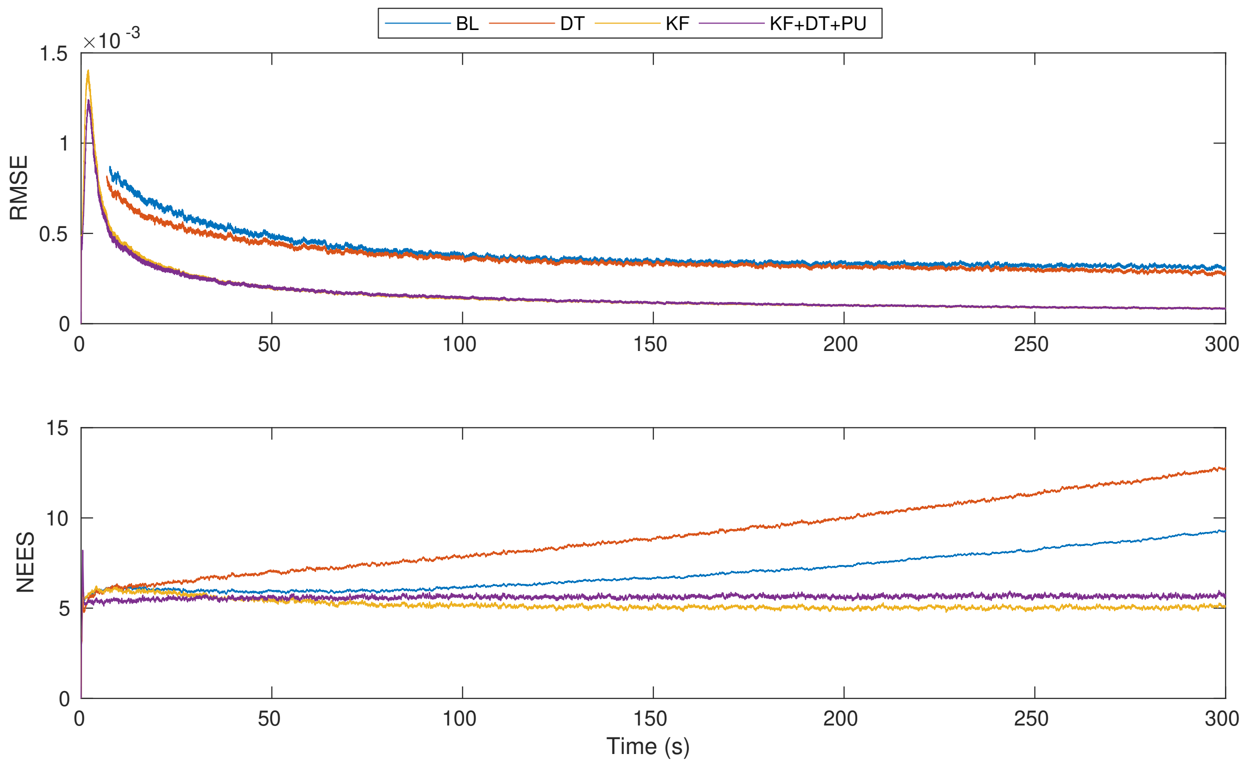


Figure 6.6: Average RMSE of the transform from the most recent keyframe (top) and average NEES (bottom) for each filter over the entire simulation time over 2016 runs.

consistency. This inconsistency can be directly mitigated through the use of a partial update, thus, providing better accuracy from the improved dynamic model, while maintaining filter consistency. Finally, the combination of all three proposed improvements was shown to improve filter accuracy and consistency over the baseline filter.

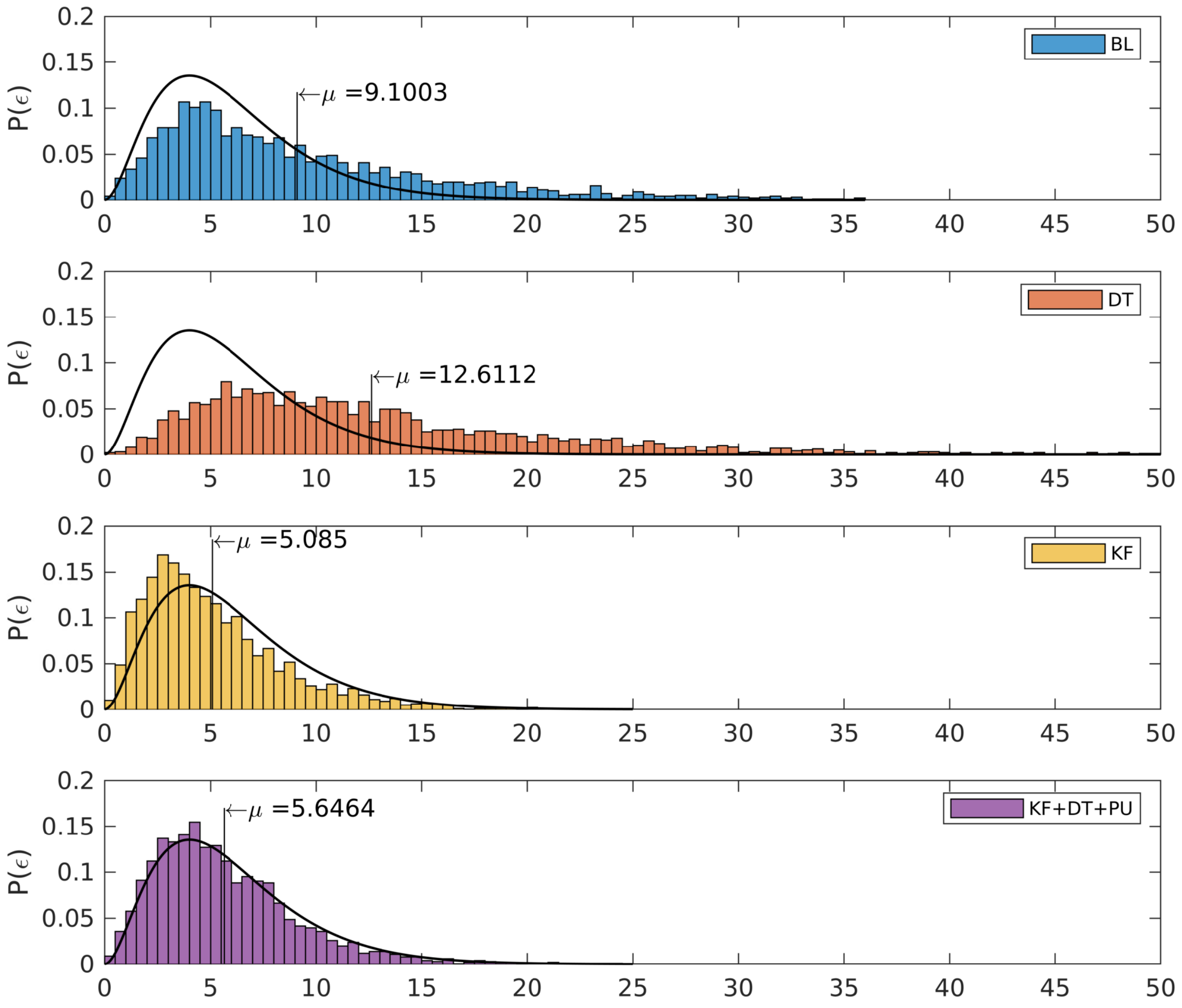


Figure 6.7: The χ^2 distribution with six degrees of freedom compared against each filter at the final simulation time of 5 minutes using 2016 samples.

CHAPTER 7: GV-INS: FUSING GNSS, VISUAL AND INERTIAL SENSORS IN A MOVING-HORIZON ESTIMATION FRAMEWORK[1]

The autonomous operation of miniature aerial vehicles (MAVs) has in recent years been the focus of many academic and commercial research efforts. This is largely due to recent advances in computational power and sensing capabilities that have enabled the building of smaller and more capable vehicles. Despite the large amount of effort put into improving the capabilities of MAVs, they have yet to be fielded in large-scale industrial applications. There are several reasons for this, but the biggest reasons center on operating autonomously under uncertainty. Localization, mapping, understanding the environment and planning paths through the environment are still areas of active of research seeking to mitigate these problems.

For ground vehicles, a common method to reduce uncertainty in autonomous operation is to use high-fidelity sensors such as LIDAR and radar arrays. The weight and size of these sensors often make them prohibitive for operation on a MAV as larger MAVs are more expensive, dangerous and complicated. In lieu of LIDAR and radar arrays, one potential approach is to use lightweight, less-powerful sensors in conjunction with more powerful algorithms to infer the structure of the environment and current pose.

To keep cost and weight low, one potential sensor suite for MAVs is the combination of a camera, inertial measurement unit (IMU), and global navigation satellite system (GNSS) receiver. All of these sensors are lightweight, relatively inexpensive, and have complementary strengths and weaknesses. GNSS receivers are only effective when in the line of sight of satellites, so they are ineffective indoors, underground, or near buildings. However, they provide accurate global information that does not drift. Inexpensive IMUs typically experience a large amount of drift and

[1]This paper will be submitted to The International Journal of Robotics Research

noise, but their performance is not dependent upon the environment and they can be sampled at high rates. Visual odometry (VO) calculated using cameras experiences drift, but the drift is slower than IMUs. Most VO algorithms, however, require sufficient texture and structure in the field of view (FOV) that is often present indoors and near buildings, but can be absent outdoors away from buildings or at high altitudes. VO also typically relies heavily on matching visual features between subsequent frames so significant changes in exposure during an indoor-to-outdoor transition can cause a loss of feature tracks.

In practice, the level of integration between sensor modalities plays a large role in the synergy between them. Deeper integration can dramatically increase the benefit of using multiple sensors at the expense of significantly increasing the complexity of the fusion. In the more common loosely coupled approach to GNSS/IMU sensor fusion, a GNSS receiver independently calculates position and velocity from pseudorange and Doppler measurements and passes them to a sensor-fusion algorithm. The fusion algorithm then combines these intermediate estimates with inertial measurements that results in a smooth estimate (Figure 7.1). A tightly-coupled GNSS-inertial (G-INS) framework fuses the pseudorange and Doppler measurements directly with the aid of inertial measurements, without the intermediate position and velocity calculation (Figure 7.2). The tightly-coupled approach results in better accuracy [95], but is more complicated to implement.

Visual odometry algorithms have an analogous relationship, where a loosely-coupled system calculates odometry between image frames independent of inertial information (Figure 7.3) and a tightly-coupled visual-inertial (V-INS) framework uses inertial information to aid the visual odometry (Figure 7.4). Most G-INS algorithms have been demonstrated effectively at high altitudes or in open spaces, where V-INS algorithms can struggle. However, V-INS algorithms have been primarily demonstrated indoors or close to buildings where G-INS is degraded or not possible. A common trouble spot is the GNSS-degraded zone, where GNSS signals are blocked by or bounce off structures and give erroneous readings, but where sufficient texture may also be unavailable for robust V-INS operation, or significant changes in lighting levels induce lost feature measurements.

In this work, we fuse information from all three of these sensing modalities (IMU, GNSS and vision) in a tightly-coupled GV-INS framework. We explore the details of this fusion through simulation studies and finally demonstrate it in a hardware experiment of a transition from an open-sky environment to an indoor environment and back through the GNSS-degraded zone in the shadow of a large building. Through these experiments, we demonstrate that the tight coupling between the three modalities allows the MAV to leverage the strengths of all three in a constructive way and overcome their respective shortcomings.

7.1 Background

7.1.1 G-INS History

Some of the first autonomous operations of MAVs used a loosely coupled G-INS estimation approach. These approaches used the position and velocity calculated by a GNSS receiver and fused this measurement with an IMU using an extended Kalman filter [96–101] or nonlinear observer [102]. These methods showed early that MAVs were capable of target tracking, path following, and various other useful tasks, however, they were limited by their reliance on GNSS, leading to brittleness in urban environments.

Development of tightly coupled G-INS solutions for MAVs has been a more recent advancement [103], but was demonstrated on larger ground vehicles earlier [95, 104–106]. Tightly coupled approaches have shown to be superior to loosely coupled architectures in terms of accuracy, time to acquisition, and the ability to deal with degraded GNSS in urban environments [107]. This is largely due to the fact that a tightly coupled estimation architecture is able to provide information even if only a single satellite is observed, and satellite positions can be estimated even in the absence of any measurements between observations [95].

One of the primary reasons for fusing raw GNSS measurements with inertial measurements is that it facilitates the identification and rejection of multipath measurements. Multipath in GNSS

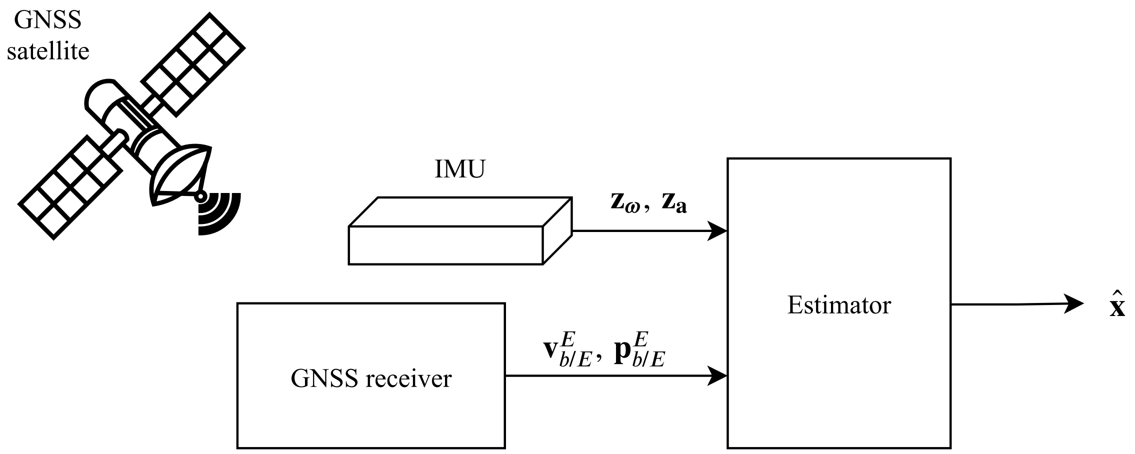

GNSS satellite
IMU
$\mathbf{z}_\omega, \mathbf{z}_\mathbf{a}$
Estimator
$\hat{\mathbf{x}}$
GNSS receiver
$\mathbf{v}_{b/E}^E, \mathbf{p}_{b/E}^E$

Figure 7.1: Block diagram for a loosely coupled GNSS-INS estimation scheme. In this configuration, the GNSS receiver is responsible for calculating an intermediate velocity and position estimate from the satellite pseudorange and doppler measurements. This intermediate measurement is then fused with inertial measurements in a state estimator.

signals appears as a step change in estimated position, and proper fusion of inertial information makes these outliers much less probable and easier to identify. Another multipath mitigation technique relies on performing inference over several measurements. This was shown by Sünderhauf et al. [108], who demonstrated a novel *switching parameter* for incorporating raw pseudorange measurements in a factor graph. This work was further explored in [109] and compared with various other robust optimization techniques. However, neither of these works were inertially-aided, nor did they run in real-time onboard an autonomous agent. We similarly utilize the switching parameter architecture, but in conjunction with our inertial measurements in a sliding-window fashion to enable us to identify and reject multipath measurements in real-time operations onboard a MAV.

7.1.2 V-INS History

Performing robust visual-inertial estimation has been the focus of considerable effort in recent literature. There have been a variety of architectures proposed including loosely coupled filtering VO approaches (as shown in Figure 7.3) such as [51, 90, 110–115], tightly coupled filter-based

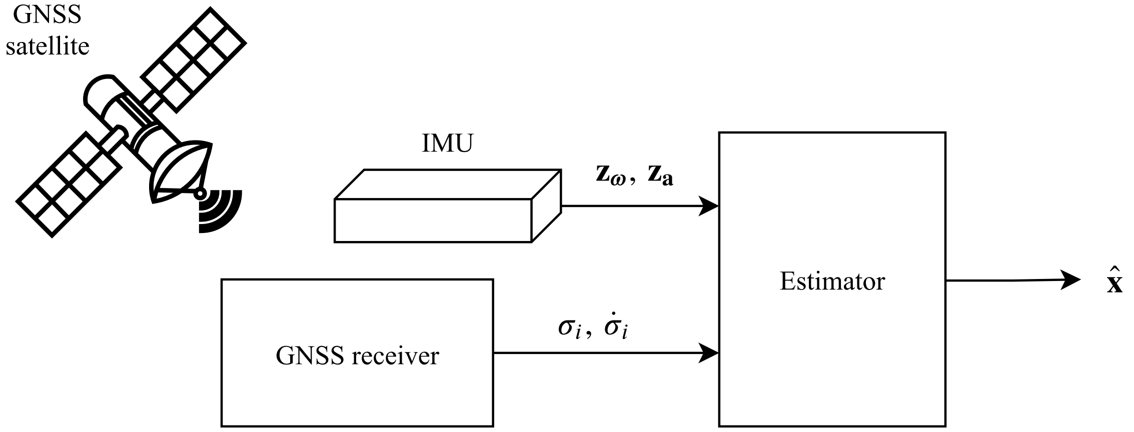

GNSS satellite
IMU
$\mathbf{z}_\omega$, $\mathbf{z}_\mathbf{a}$
Estimator
$\hat{\mathbf{x}}$
σ_i, $\dot{\sigma}_i$
GNSS receiver

Figure 7.2: Block diagram for a tightly coupled GNSS-INS estimation scheme. In this configuration, the state estimator is given pseudorange and doppler measurements that are fused directly with inertial data.

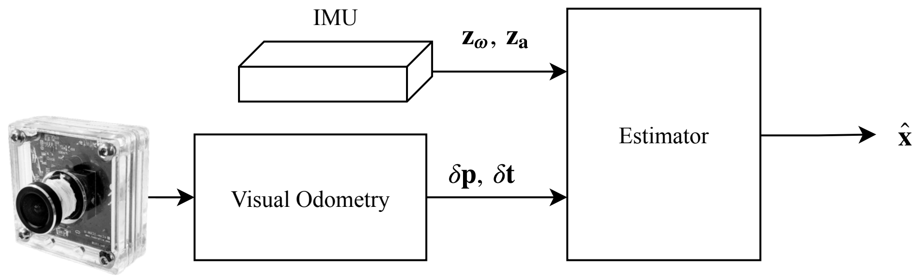

IMU
$\mathbf{z}_\omega$, $\mathbf{z}_\mathbf{a}$
Estimator
$\hat{\mathbf{x}}$
Visual Odometry
$\delta\mathbf{p}$, $\delta\mathbf{t}$

Figure 7.3: Block diagram for a loosely coupled V-INS estimation scheme. In this configuration, the image data is processed by a visual odometry algorithm that calculates an intermediate relative change in position and attitude from image frame to image frame. This intermediate measurement is then fused with inertial measurements in a state estimator.

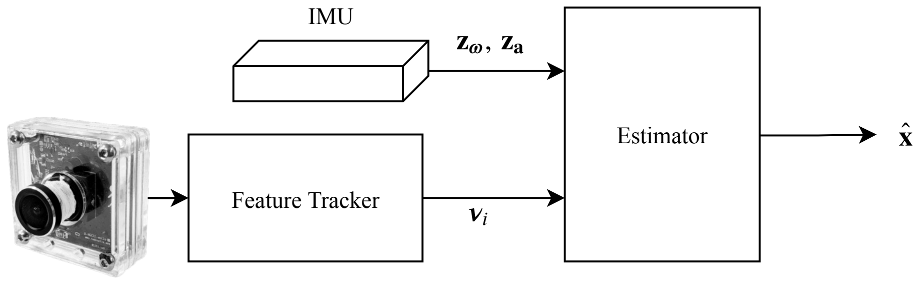


Figure 7.4: Block diagram for a tightly coupled V-INS estimation scheme. In this configuration, the pixel locations of tracked features are given to the estimator directly, and are fused with directly with inertial data.

approaches (shown in Figure 7.4) [81, 82, 84, 92] and moving-horizon-based approaches [82, 86, 116–118].

Tightly coupled, moving horizon estimation (MHE) approaches have shown considerable advantages in terms of accuracy and robustness when compared to loosely coupled architectures and MHE has been shown to be superior to filtering, but is more computationally complex. While filtering approaches have the advantage of being computationally efficient, they can struggle in certain situations to overcome significant nonlinearities and weakly observable states [51, 119]. On the other hand, effort has been put into reducing the computational load required for MHE approaches, such as IMU preintegration [117], incremental smoothing [120, 121] and graph reduction [122]. Recent work comparing MHE approaches with filtering approaches have shown that, per unit of computational effort, MHE approaches have a greater return in terms of accuracy than filtering approaches [123]. Furthermore, modern computational capabilities have arrived to the point that even inexpensive MAVs are capable of performing MHE with onboard computation [124]. Because of these advancements we have chosen to adopt an MHE approach in our tightly coupled GV-INS framework, described in Figure 7.5. Our work is most like [86] in that we use a similar IMU preintegration mechanism and projection factor, however, we have extended the visual-inertial approach described to allow for raw pseudorange measurements and made several other necessary adjustments ensure real-time operations as described later.

The recent work in visual-inertial estimation theory has largely left the line of research in GNSS integration alone. There have been a few loosely coupled GV-INS architectures demonstrated [51, 125, 126], a few hybrid systems with tightly coupled vision but loosely coupled GNSS [127, 128], and also tightly coupled GNSS but loosely coupled vision [106]. However, to our knowledge there has not been a fusion of the new ideas from the state-of-the-art visual-inertial research to the more established tightly coupled GNSS integration. We believe that the fusion of these ideas results in an estimation scheme that can effectively operate in the transition area between between GNSS-denied operation and reliable GNSS operation.

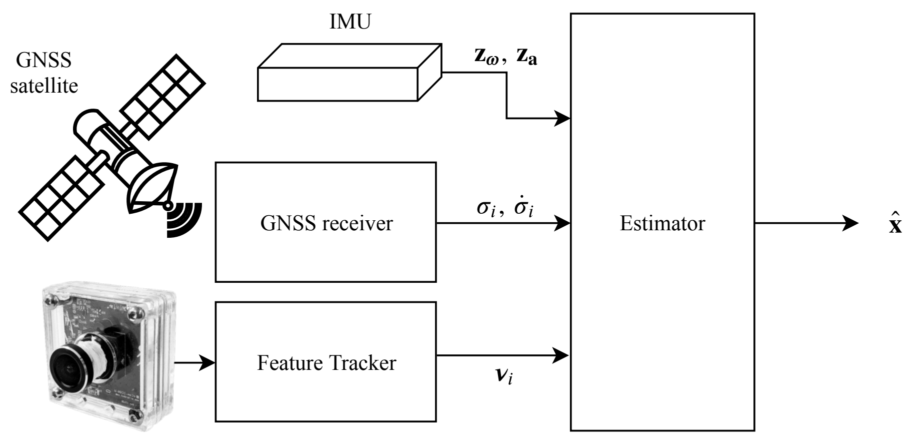


Figure 7.5: Block diagram for proposed tightly coupled GVINS architecture. The estimator is responsible for fusing raw pseudorange measurements as well as the pixel location for tracked features with IMU measurements.

In the following sections we first derive the MHE problem, and then the relevant factors used in our MHE: the IMU preintegration factor, the pseudorange factor, and the feature projection factor. Second, we describe the set up of our MHE and the management of the relevant estimated parameters. We then show the results of some simulation studies and finally the results of a hardware experiment.

7.2 Notation

The following notation is used throughout the rest of the paper. First, we make use of the following coordinate frames:

I	Inertial coordinate frame (origin of trajectory)
E	Earth-centered, earth-fixed coordinate frame as described by WGS84
b	body frame (assume centered at IMU location)
c	camera frame
g	GNSS antenna coordinate frame.

We also use the following variable naming conventions:

$\mathbf{e}_i$	unit vector in the i^{th} direction
$\mathbf{p}$	position
$\mathbf{v}$	velocity
$\mathbf{q}$	unit quaternion rotation
$\mathbf{g}$	gravity
R	rotation matrix
$\boldsymbol{\beta}_a$	accelerometer bias
$\boldsymbol{\beta}_\omega$	rate gyro bias
τ	clock bias
$\boldsymbol{\zeta}_i$	unit vector directed at the i^{th} feature from the camera origin
ρ_i	inverse distance to the i^{th} feature
σ_i	pseudorange to i^{th} satellite
κ_i	psuedorange switching variable for i^{th} satellite
$\boldsymbol{\nu}_i$	x-y pixel measurement of the i^{th} feature,
$\hat{a}$	estimated quantity a
$\bar{a}$	measured quantity a
$\tilde{a}$	error in quantity a

and describe the expression of quantities in terms of their coordinate frames as follows:

$\mathbf{v}_{a/b}^{c}$	velocity of coordinate frame a with respect to frame b, expressed in frame c
$\mathbf{q}_a^b$	quaternion rotation from frame a to frame b.

We also make extensive use of the skew-symmetric matrix operator that is related to the cross-product between two vectors as

$$\mathbf{v} \times \mathbf{w} = \lfloor \mathbf{v} \rfloor_\times \mathbf{w}.$$

7.2.1 Quaternions

We use Hamiltonian notation for unit quaternions $\in \mathcal{S}^3$

$$\mathbf{q} = q_0 + q_x \mathbf{e}_1 + q_y \mathbf{e}_2 + q_z \mathbf{e}_3 = \begin{bmatrix} q_0 \\ \vec{\mathbf{q}} \end{bmatrix},$$

which defines the passive rotation matrix based on a unit quaternion as

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0 \lfloor \vec{\mathbf{q}} \rfloor_\times + 2\vec{\mathbf{q}}\vec{\mathbf{q}}^\top \in SO(3). \tag{7.1}$$

This definition results in $R_a^b \mathbf{r}^a = \mathbf{r}^b$ being interpreted as the original vector $\mathbf{r}^a$ expressed in the new coordinate frame b.

We define the exponential mapping for a unit quaternion as

$$\begin{aligned} \exp_{\mathbf{q}} &: \mathbb{R}^3 \to \mathcal{S}^3 \\ \exp_{\mathbf{q}}(\boldsymbol{\theta}) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{\theta}\|}{2}\right) \\ \sin\left(\frac{\|\boldsymbol{\theta}\|}{2}\right) \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|} \end{bmatrix}, \end{aligned}$$

and define the corresponding logarithmic map as

$$\begin{aligned} \log_{\mathbf{q}} &: \mathcal{S}^3 \to \mathbb{R}^3 \\ \log_{\mathbf{q}}(\mathbf{q}) &\triangleq 2 \operatorname{atan2}\left(\|\vec{\mathbf{q}}\|, q_0\right) \frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|}. \end{aligned}$$

7.2.2 Rigid Transforms

We define a rigid transform $\in SE(3)$ as the tuple of a translation vector and rotation quaternion $T_a^b = \left(\mathbf{p}_{b/a}^a, \mathbf{q}_a^b\right)$ for efficient implementation. If $T_a^b = \left(\mathbf{p}_{b/a}^a, \mathbf{q}_a^b\right)$ and $T_b^c = \left(\mathbf{p}_{c/b}^b, \mathbf{q}_b^c\right)$ then rigid transform multiplication is defined by

$$\begin{aligned} T_a^c &= T_a^b \circ T_b^c \\ &\triangleq \left(\mathbf{p}_{b/a}^a + R\left(\mathbf{q}_a^b\right)^{-1} \mathbf{p}_{c/b}^b, \mathbf{q}_a^b \otimes \mathbf{q}_b^c\right) \\ &= \left(\mathbf{p}_{c/a}^a, \mathbf{q}_a^c\right), \end{aligned}$$

with the inverse given by

$$\left(T_a^b\right)^{-1} = \left(-R\left(\mathbf{q}_a^b\right)\mathbf{p}_{b/a}^a, \left(\mathbf{q}_a^b\right)^{-1}\right),$$

such that

$$T_a^b \circ \left(T_a^b\right)^{-1} = \left(T_a^b\right)^{-1} \circ T_a^b = \left(\mathbf{0}, \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}\right).$$

Given linear and angular changes $\left(\boldsymbol{\delta} \in \mathbb{R}^3, \boldsymbol{\theta} \in \mathbb{R}^3\right) \sim \mathbb{R}^6$, the exponential mapping for a rigid transform is given by

$$\begin{gathered} \exp_T : \mathbb{R}^6 \to SE(3) \\ \exp_T(\boldsymbol{\delta}, \boldsymbol{\theta}) \triangleq \left(V\boldsymbol{\delta}, \exp_{\mathbf{q}} \boldsymbol{\theta}\right), \end{gathered}$$

where

$$V = I + \frac{1 - \cos\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^2} \lfloor\boldsymbol{\theta}\rfloor_\times + \frac{\|\boldsymbol{\theta}\| - \sin\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^3} \lfloor\boldsymbol{\theta}\rfloor_\times \lfloor\boldsymbol{\theta}\rfloor_\times .$$

The corresponding logarithmic map is defined by

$$\begin{gathered} \log_T : SE(3) \to \mathbb{R}^6 \\ \log_T(T) \triangleq \left(V^{-1}\mathbf{p}, \log_{\mathbf{q}}(\mathbf{q})\right), \end{gathered}$$

where

$$V^{-1} = I - \frac{1}{2}\lfloor \boldsymbol{\theta} \rfloor_\times + \frac{1}{\|\boldsymbol{\theta}\|^2}\left(1 - \frac{\|\boldsymbol{\theta}\| \sin \|\boldsymbol{\theta}\|}{2\left(1 - \cos \|\boldsymbol{\theta}\|\right)}\right) \lfloor \boldsymbol{\theta} \rfloor_\times \lfloor \boldsymbol{\theta} \rfloor_\times .$$

7.3 Moving Horizon Estimation

The objective of the moving horizon estimation (MHE) problem is to find the maximum-likelihood estimate of some recent window of states given sensor measurements that occurred in this window. This problem is written as

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} P\left(\mathbf{x}|\mathbf{z}\right), \tag{7.2}$$

where $\mathbf{x}$ are all the states in the sliding window and $\mathbf{z}$ are all the measurements that occurred in this window. If we assume that measurements are independent, then we can factor Eq. 7.2 into a product of the probability of each measurement

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} \prod_i^n P\left(\mathbf{x}|\mathbf{z}_i\right). \tag{7.3}$$

Next, if we assume that we have some measurement model function $\mathbf{z}_i = h_i\left(\mathbf{x}\right) + \boldsymbol{\eta}_i$ where $\boldsymbol{\eta}_i \sim \mathcal{N}\left(0, \Sigma_i\right)$, then

$$P\left(\mathbf{x}|\mathbf{z}_i\right) \propto \exp\left(-\frac{1}{2}\mathbf{r}_i^\top \Sigma_i^{-1} \mathbf{r}_i\right),$$

where the measurement residual $\mathbf{r}_i = \mathbf{z}_i - h_i\left(\mathbf{x}\right)$.

Although our objective is to solve the maximum likelihood problem, computing Eq. 7.3 directly is likely to run into numerical precision issues. Therefore, we transform Eq. 7.3 with the negative logarithm and search for the global minima of the result.

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \left(-\log \left(\prod_n \exp\left(-\frac{1}{2} \mathbf{r}_i^\mathsf{T} \Sigma_i^{-1} \mathbf{r}_i \right) \right) \right)$$

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \sum_n \frac{1}{2} \mathbf{r}_i^\mathsf{T} \Sigma_i^{-1} \mathbf{r}_i \qquad (7.4)$$

Eq. 7.4 is the well-known nonlinear least-squares optimization problem. There are a wide variety of high-performance solvers that can be used to solve this problem [8, 85, 121, 129, 130], most of which rely on some Newton-based method. In our implementation, we utilize the ceres solver [129] that has been optimized for this type of problem.

The factoring of the problem in Eq. 7.3 when applied to the equivalent Bayesian network gives rise to the concept of a *factor graph*, where the factors are literally the factors of the product in Eq. 7.3, or the measurement models. The factor graph of the GPS-inertial MHE problem with preintegrated IMU is illustrated in Figure 7.6 as an example.

To fully describe our approach, we now need to define the relevant factors for our problem that are expressed in Eq. 7.3. These are derived in the next three sections. We first derive the preintegrated IMU factor (Section 7.4), then the projection factor used for fusing visual information (Section 7.5), and finally the pseudorange factor (Section 7.6).

7.4 IMU Preintegration

The first and most complicated factor relates two subsequent states using IMU measurements. A challenge when fusing IMU measurements is the high rate that measurements occur, and the associated computational complexity that arises from having a large number of parameters to optimize. Therefore, to reduce the computational burden associated with inference we can combine a bunch of IMU measurements into a single factor so we do not end up having an unreasonable number of nodes to optimize over. In the factor graph literature, this process is known as IMU

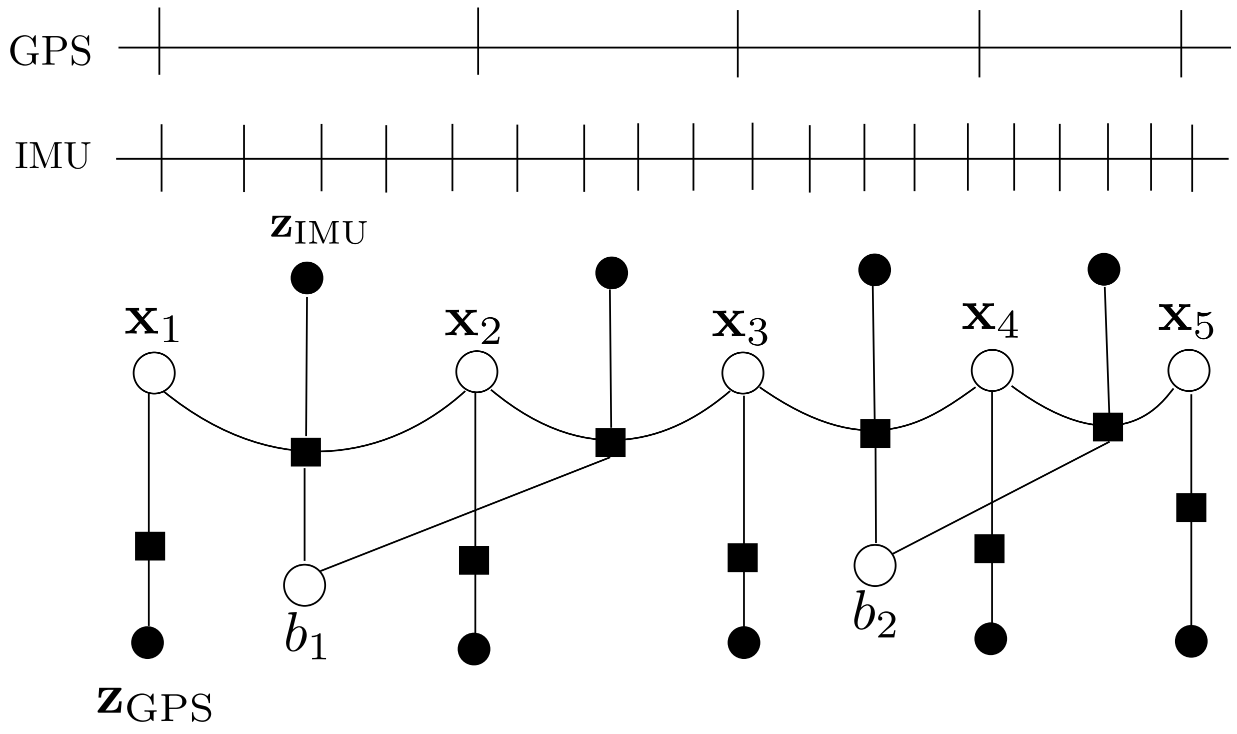


Figure 7.6: An example factor graph of a sliding window with preintegrated IMU. Filled circles represent measurement values, while unfilled circles represent estimated values. Squares represent the independent components of Eq. 7.3 and are known as *factors*. These nodes encode the relationship between the variables in the graph.

preintegration, as popularized for use in Visual-Inertial factor graphs by [116]. However, it very similar to well-established strapdown inertial navigation IMU integration techniques that use coning or sculling compensation [131] to reduce the computational burden of fusing high-rate IMU on constrained platforms.

Our formulation integrates the raw IMU measurements on the rotation manifold, like [86, 116], but uses the concept of the error-state to derive the covariance in a straight-forward manner.

To derive the method of IMU preintegration, let us first consider the change in state between two subsequent poses i and j given by

$$\begin{bmatrix} \mathbf{p}^I_{j/I} \\ \mathbf{v}^j_{j/I} \\ \mathbf{q}^j_I \end{bmatrix} = \begin{bmatrix} \mathbf{p}^I_{i/I} + \iint_{t_i}^{t_j} \left(R^I_\tau \mathbf{a}^\tau_{\tau/I} + \mathbf{g}^I \right) d\tau\, d\tau \\ R^j_i \left(\mathbf{v}^i_{i/I} + \int_{t_i}^{t_j} \left(R^i_\tau \mathbf{a}^\tau_{\tau/I} + R^i_I \mathbf{g}^I \right) d\tau \right) \\ \mathbf{q}^i_I \otimes \exp_{\mathbf{q}} \left(\int_{t_i}^{t_j} \boldsymbol{\omega}^\tau_{\tau/I} d\tau \right) \end{bmatrix}. \tag{7.5}$$

We would like to separate the part of this integration that is dependent on the state at node i from the rest of the integration. We call the remainder of the integration $\mathbf{y}^i_{j/i}$, which is the component dependent only on the IMU measurements. Isolating this part of the interval allows us to change the pose of node i without being required to re-compute our preintegrated measurement. We can separate the the dependent part of the transition by first pulling the constant quantities out of the integrals. If $\delta t = t_j - t_i$, then Eq. 7.5 can be written as

$$\begin{bmatrix} \mathbf{p}^I_{j/I} \\ \mathbf{v}^j_{j/I} \\ \mathbf{q}^j_I \end{bmatrix} = \begin{bmatrix} \mathbf{p}^I_{i/I} + \frac{1}{2}\mathbf{g}^I \delta t^2 + \mathbf{v}^I_{i/I} \delta t + \iint_{t_i}^{t_j} R^I_\tau \mathbf{a}^\tau_{\tau/I} d\tau\, d\tau \\ R^j_i \left(\mathbf{v}^i_{i/I} + R^i_I \mathbf{g}^I \delta t + \int_{t_i}^{t_j} R^i_\tau \mathbf{a}^\tau_{\tau/I} d\tau \right) \\ \mathbf{q}^i_I \otimes \exp_{\mathbf{q}} \left(\int_{t_i}^{t_j} \boldsymbol{\omega}^\tau_{\tau/I} d\tau \right) \end{bmatrix}.$$

We then can define the IMU-dependent part of the transition as

$$\mathbf{y}^i_{j/i} = \begin{bmatrix} \boldsymbol{\alpha}^i_{j/i} \\ \boldsymbol{\beta}^i_{j/i} \\ \boldsymbol{\gamma}^j_i \end{bmatrix} = \begin{bmatrix} \iint_{t_i}^{t_j} R^I_\tau \mathbf{a}^\tau_{\tau/I} d\tau\, d\tau \\ \int_{t_i}^{t_j} R^i_\tau \mathbf{a}^\tau_{\tau/I} d\tau \\ \exp_{\mathbf{q}} \left(\int_{t_i}^{t_j} \boldsymbol{\omega}^\tau_{\tau/I} d\tau \right) \end{bmatrix},$$

which is our preintegrated IMU measurement.

Given this definition, can write the continuous-time dynamics of $\mathbf{y}^i_{j/i}$ over the interval as

$$\dot{\mathbf{y}}^i_{j/i} = \begin{bmatrix} \dot{\boldsymbol{\alpha}}^i_{b/i} \\ \dot{\boldsymbol{\beta}}^i_{b/i} \\ \dot{\boldsymbol{\gamma}}^b_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{\beta}^i_{b/i} \\ R^i_b \mathbf{a}^b_{b/i} \\ \frac{1}{2}\left(\boldsymbol{\gamma}^b_i \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b_{b/i} \end{bmatrix}\right) \end{bmatrix},$$

and because we assume that we have an IMU measurement $\begin{bmatrix} \bar{\mathbf{a}} & \bar{\omega} \end{bmatrix}^\top$ with constant bias $\begin{bmatrix} \mathbf{b}_a & \mathbf{b}_\omega \end{bmatrix}^\top$ and noise $\begin{bmatrix} \boldsymbol{\eta}_a, \boldsymbol{\eta}_\omega \end{bmatrix}^\top$, the dynamics of $\mathbf{y}^i_{j/i}$ over the interval are given as

$$\dot{\mathbf{y}}^i_{j/i} = \begin{bmatrix} \dot{\boldsymbol{\alpha}}^i_{b/i} \\ \dot{\boldsymbol{\beta}}^i_{b/i} \\ \dot{\boldsymbol{\gamma}}^b_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{\beta}^i_{b/i} \\ R^i_b \left(\bar{\mathbf{a}} - \mathbf{b}_a + \boldsymbol{\eta}_a\right) \\ \frac{1}{2}\left(\boldsymbol{\gamma}^b_i \otimes \begin{bmatrix} 0 \\ \bar{\omega} - \mathbf{b}_\omega + \boldsymbol{\eta}_\omega \end{bmatrix}\right) \end{bmatrix}.$$

7.4.1 Covariance Propagation

Given $\mathbf{y}^i_{j/i}$ and the estimates of nodes i and j we now have the information required to calculate the residual $\mathbf{r}_{IMU}$ for Eq. 7.4, however, we also need the covariance of this residual before we can properly weight it in the optimization. To derive the covariance of the preintegrated measurement we will utilize what is known as the *error state* dynamics of $\mathbf{y}$. The covariance of the preintegrated IMU measurement is given as

$$\Sigma_{\mathbf{y}} = E\left[\tilde{\mathbf{y}}\tilde{\mathbf{y}}^\top\right].$$

If the preintegrated measurement were a vector like other residuals, then we would expect the $\tilde{\mathbf{y}}$ to be computed using vector subtraction as

$$\tilde{\mathbf{y}} = \mathbf{y} - \hat{\mathbf{y}}.$$

However, because the rotation component of $\mathbf{y}$ is a quaternion, we must use a more generalized notion of differencing, where we use the quaternion logarithm to map the difference between two rotations into a vector space. This results in $\tilde{\mathbf{y}}$ being defined as follows:

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\alpha}^{i}_{j/\hat{j}} \\ \tilde{\beta}^{i}_{j/\hat{j}} \\ \tilde{\gamma}^{j}_{\hat{j}} \end{bmatrix} = \begin{bmatrix} \alpha^{i}_{j/i} - \hat{\alpha}^{i}_{\hat{j}/i} \\ \beta^{i}_{j/i} - \hat{\beta}^{i}_{\hat{j}/i} \\ \log_{\mathbf{q}}\left(\left(\hat{\gamma}^{\hat{j}}_{i}\right)^{-1} \otimes \gamma^{j}_{i}\right) \end{bmatrix}. \tag{7.6}$$

We can use the dynamics of $\tilde{\mathbf{y}}$ to propagate the covariance over the interval. Unfortunately, the dynamics of $\tilde{\mathbf{y}}$ are not trivial, but we will derive them in the following paragraphs.

Before we can derive the dynamics of the error state of $\mathbf{y}$, however, we must consider the error-state inputs to our system. Like the error state, the error state inputs are defined as

$$\tilde{\mathbf{u}} = \mathbf{u} - \hat{\mathbf{u}}.$$

In our case, we have accelerometer and rate gyroscope measurements that are the inputs to our IMU preintegration system. We also have a current estimate of the constant bias for both of these sensors $\hat{\mathbf{b}}_a$ and $\hat{\mathbf{b}}_\omega$.

Let us first consider the angular velocity input. The true angular velocity is assumed to take the form of

$$\omega = \bar{\omega} - \mathbf{b}_\omega + \boldsymbol{\eta}_\omega,$$

where $\boldsymbol{\eta}_\omega$ is some Gaussian process and $\mathbf{b}_\omega$ is the true gyroscope bias. Our estimated angular velocity input is also defined as

$$\hat{\omega} = \bar{\omega} - \hat{\mathbf{b}}_\omega,$$

which means that the error-state input is given as

$$\begin{aligned}\tilde{\omega} &= \omega - \hat{\omega} \\ &= \bar{\omega} - \mathbf{b}_\omega + \boldsymbol{\eta}_\omega - \left(\bar{\omega} - \hat{\mathbf{b}}_\omega\right) \\ &= -\tilde{\mathbf{b}}_\omega + \boldsymbol{\eta}_\omega .\end{aligned}$$

The accelerometer error-state model is derived in a similar way, and is given as

$$\begin{aligned}\mathbf{a} &= \bar{\mathbf{a}} - \mathbf{b}_a + \boldsymbol{\eta}_a \\ \hat{\mathbf{a}} &= \bar{\mathbf{a}} - \hat{\mathbf{b}}_a, \\ \tilde{\mathbf{a}} &= -\tilde{\mathbf{b}}_a + \boldsymbol{\eta}_a .\end{aligned}$$

Now we can look at the error state of the preintegrated measurement. First let us consider $\tilde{\boldsymbol{\alpha}}$. This term turns out to be pretty simple, because $\boldsymbol{\beta}^i_{j/i}$ is already expressed in the origin frame of the preintegration interval i. Therefore,the dynamics of $\tilde{\boldsymbol{\alpha}}$ are given as

$$\begin{aligned}\dot{\tilde{\boldsymbol{\alpha}}}^i_{j/\hat{j}} &= \boldsymbol{\beta}^i_{j/i} - \hat{\boldsymbol{\beta}}^i_{\hat{j}/i} \\ &= \tilde{\boldsymbol{\beta}}^i_{j/\hat{j}} .\end{aligned} \tag{7.7}$$

The velocity-like term $\boldsymbol{\beta}$, however, requires us to rotate our IMU measurements into the origin frame i. If we use the identity $R\left(\exp_{\mathbf{q}}(\boldsymbol{\theta})\right) \approx I - \lfloor \boldsymbol{\theta} \rfloor_\times$ (Eq. 7.1) and drop all terms where error-quantities are multiplied together, then we can derive the dynamics of $\tilde{\boldsymbol{\beta}}$ as follows:

$$\begin{aligned}\dot{\tilde{\boldsymbol{\beta}}}^i_{j/\hat{j}} &= \dot{\boldsymbol{\beta}}^i_{j/i} - \dot{\hat{\boldsymbol{\beta}}}^i_{\hat{j}/i} \\ &= \left(R^j_i\right)^\top \left(\mathbf{a}^j_{j/i}\right) - \left(R^{\hat{j}}_i\right)^\top \left(\hat{\mathbf{a}}^{\hat{j}}_{\hat{j}/i}\right) \\ &= \left(\hat{R}^{\hat{j}}_i\right)^\top \left(\tilde{R}^{\hat{j}}_j\right)^\top \left(\hat{\mathbf{a}}^{\hat{j}}_{\hat{j}/i} + \tilde{\mathbf{a}}^{\hat{j}}_{\hat{j}/b}\right) - \left(R^{\hat{j}}_i\right)^\top \left(\hat{\mathbf{a}}^{\hat{j}}_{\hat{j}/i}\right)\end{aligned}$$

$$\begin{aligned}
&= \left(\hat{R}_i^{\hat{j}}\right)^\top R\left(\exp_{\mathbf{q}}\left(\tilde{\gamma}_j^{\hat{j}}\right)\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} + \tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) - \left(R_i^{\hat{j}}\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}}\right) \\
&\approx \left(\hat{R}_i^{\hat{j}}\right)^\top \left(I - \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} + \tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) - \left(R_i^{\hat{j}}\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}}\right) \\
&= \left(\hat{R}_i^{\hat{j}}\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} + \tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) + \left(\hat{R}_i^{\hat{j}}\right)^\top \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} + \tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) - \left(R_i^{\hat{j}}\right)^\top \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}}\right) \\
&= \left(\hat{R}_i^{\hat{j}}\right)^\top \left(\tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) + \left(\hat{R}_i^{\hat{j}}\right)^\top \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times \left(\hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} + \tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) \\
&\approx \left(\hat{R}_i^{\hat{j}}\right)^\top \left(\tilde{\mathbf{a}}_{j/\hat{j}}^{\hat{j}}\right) + \left(\hat{R}_i^{\hat{j}}\right)^\top \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times \hat{\mathbf{a}}_{\hat{j}/i}^{\hat{j}} \\
&= \left(\hat{R}_i^{\hat{j}}\right)^\top \left(-\tilde{\mathbf{b}}_a + \eta_{\mathbf{a}}\right) + \left(\hat{R}_i^{\hat{j}}\right)^\top \left\lfloor \tilde{\gamma}_{\hat{j}}^{j} \right\rfloor_\times \left(\bar{\mathbf{a}} - \hat{\mathbf{b}}_a\right) \\
&= -R\left(\gamma_i^{\hat{j}}\right)^\top \left(\left\lfloor \bar{\mathbf{a}} - \hat{\mathbf{b}}_a \right\rfloor_\times \tilde{\gamma}_j^{\hat{j}} - \tilde{\mathbf{b}}_a + \eta_{\mathbf{a}}\right).
\end{aligned} \tag{7.8}$$

This derivation assumes that error-state quantities are small, which is a reasonable assumptions, as we will only be using the Jacobian of these dynamics to propagate our covariance.

Finally, the derivative of our attitude portion $\tilde{\gamma}$ is derived using the same approximations as

$$\begin{aligned}
\dot{\tilde{\gamma}}_j^{\hat{j}} &= R\left(\exp_{\mathbf{q}}\left(\tilde{\gamma}_j^{\hat{j}}\right)\right) \omega_{j/i}^{j} - \hat{\omega}_{\hat{j}/i}^{\hat{j}} \\
&\approx \left(I - \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times\right)^\top \left(\hat{\omega}_{\hat{j}/i}^{\hat{j}} + \tilde{\omega}_{j/\hat{j}}^{\hat{j}}\right) - \hat{\omega}_{\hat{j}/i}^{\hat{j}} \\
&= \left(\hat{\omega}_{\hat{j}/i}^{\hat{j}} + \tilde{\omega}_{j/\hat{j}}^{\hat{j}}\right) - \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times^\top \left(\hat{\omega}_{\hat{j}/i}^{\hat{j}} + \tilde{\omega}_{j/\hat{j}}^{\hat{j}}\right) - \hat{\omega}_{\hat{j}/i}^{\hat{j}} \\
&= \tilde{\omega}_{j/\hat{j}}^{\hat{j}} - \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times^\top \left(\hat{\omega}_{\hat{j}/i}^{\hat{j}} + \tilde{\omega}_{j/\hat{j}}^{\hat{j}}\right) \\
&\approx \tilde{\omega}_{j/\hat{j}}^{\hat{j}} - \left\lfloor \tilde{\gamma}_j^{\hat{j}} \right\rfloor_\times^\top \hat{\omega}_{\hat{j}/i}^{\hat{j}} \\
&= \left(-\tilde{\mathbf{b}}_\omega + \boldsymbol{\eta}_\omega\right) + \left\lfloor \tilde{\gamma}_{\hat{j}}^{j} \right\rfloor_\times \left(\bar{\omega} - \hat{\mathbf{b}}_\omega\right) \\
&= -\left\lfloor \bar{\omega} - \hat{\mathbf{b}}_\omega \right\rfloor_\times \left(\tilde{\gamma}_{\hat{j}}^{j}\right) - \tilde{\mathbf{b}}_\omega + \boldsymbol{\eta}_\omega.
\end{aligned} \tag{7.9}$$

We can now linearize Eqs. 7.7, 7.8 and 7.9 to get our continuous-time state-space Jacobians

$$A = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & -R\left(\gamma_i^b\right)^\top \lfloor \bar{\mathbf{a}} - \mathbf{b}_a \rfloor_\times \\ 0 & 0 & -\lfloor \bar{\omega} - \mathbf{b}_\omega \rfloor_\times \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ -R\left(\gamma_i^b\right)^\top & 0 \\ 0 & I \end{bmatrix}$$

such that

$$\dot{\tilde{\mathbf{y}}} \approx A(\mathbf{y}, \mathbf{u})\,\tilde{\mathbf{y}} + B(\mathbf{y})\,\boldsymbol{\eta}. \tag{7.10}$$

We wish to integrate $\Sigma_{\mathbf{y}}$ according to our discrete sample time δt. Therefore we discretize Eq. 7.10 with

$$\mathbf{y}[t+\delta t] = \bar{A}\mathbf{y}[t] + \bar{B}\mathbf{z}[t],$$

where

$$\begin{bmatrix} \bar{A} & \bar{B} \\ \mathbf{0} & I \end{bmatrix} = \exp\left(\begin{bmatrix} A & B \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \delta t\right). \tag{7.11}$$

Eq. 7.11 can be approximated as

$$\bar{A} = I + A\delta t + \frac{A^2}{2}\delta t^2$$
$$\bar{B} = \left(I + \frac{1}{2}A\delta t\right) B\delta t$$

without significant loss of precision given small δt. Now, with these discrete Jacobians, we can propagate our covariance

$$\Sigma_{\mathbf{y}}[t+\delta t] = \bar{A}\Sigma_{\mathbf{y}}[t]\,\bar{A}^\top + \bar{B}\Sigma_{\mathbf{u}}\bar{B}^\top \tag{7.12}$$

with $\Sigma_{\mathbf{y}}[0] = \mathbf{0}$.

7.4.2 IMU Bias Adjustment

The reason we derived $\mathbf{y}$ in the manner we did was to eliminate dependence of $\mathbf{y}$ on the pose of node i. Otherwise, we have to recompute $\mathbf{y}$ every time we change i during optimization. Unfortunately, there is no known way to eliminate our dependence on our initial estimate of the IMU bias, $\mathbf{b}$. However, if we assume that our initial estimate of $\mathbf{b}$ is close to its actual value, then we can use a linear approximation of how $\mathbf{y}$ changes with respect to changes in $\mathbf{b}$ and compensate for the deviation in real-time.

Let us assume that the integration of $\mathbf{y}_0$ occurred with $\mathbf{b} = \mathbf{b}_0$, but during optimization, we want to find the value of $\mathbf{y}'$ given $\mathbf{b}' = \mathbf{b}_0 + \delta\mathbf{b}$. The linear approximation is then

$$\mathbf{y}' = \mathbf{y}_0 + \frac{\partial \mathbf{y}}{\partial \mathbf{b}} \delta\mathbf{b}.$$

We can compute $\frac{\partial \mathbf{y}}{\partial \mathbf{b}}$ along with the covariance as we get new IMU measurements. If we let $J = \frac{\partial \mathbf{y}}{\partial \mathbf{b}}$, and $J[0] = \mathbf{0}$, then J propagates according to

$$J[t + \delta t] = \bar{A} J[t] + \bar{C}\delta t.$$

where C is the Jacobian of our dynamics with respect to the bias. Conveniently, it is pretty easy to see through inspection of Eqs. 7.7, 7.8 and 7.9 that $\bar{C} = -\bar{B}$.

7.4.3 IMU Residual Calculation

Finally, now that we have computed our preintegrated measurement $\mathbf{y}^i_{j/i}$, compensated for any change in IMU bias, and calculated the covariance $\Sigma_{\mathbf{y}}$ we can calculate the residual to use in Eq. 7.4.

The measurement model for $\mathbf{y}$ given the estimated pose and velocity for nodes i and j is given as

$$\hat{\mathbf{y}}^i_{j/i} = \begin{bmatrix} \hat{\alpha}^i_{j/i} \\ \hat{\alpha}^i_{j/i} \\ \hat{\gamma}^j_i \end{bmatrix} = \begin{bmatrix} \left(\hat{R}^i_I\right)\left(\hat{\mathbf{p}}^I_{j/I} - \hat{\mathbf{p}}^I_{i/I} - \frac{1}{2}\mathbf{g}^I\delta t^2\right) - \hat{\mathbf{v}}^i_{i/I}\delta t \\ \left(\hat{R}^j_i\right)^\top \mathbf{v}^j_{j/I} - \hat{R}^i_I\mathbf{g}^I\delta t \\ \left(\left(\hat{\mathbf{q}}^i_I\right)^{-1} \otimes \hat{\mathbf{q}}^j_I\right) \end{bmatrix}.$$

We compute the difference our measured and estimated quantities of $\mathbf{y}$ using the same generalized difference procedure used in our error-state derivation Eq. 7.6 to map the rotation error into a vector space as follows:

$$\mathbf{r}_{IMU} = \begin{bmatrix} \bar{\alpha}^i_{j/i} - \hat{\alpha}^i_{j/i} \\ \bar{\beta}^i_{j/i} - \hat{\beta}^i_{j/i} \\ \log_{\mathbf{q}}\left(\left(\bar{\gamma}^j_i\right)^{-1} \otimes \hat{\gamma}^j_i\right) \end{bmatrix}.$$

This residual is then weighted by the covariance we compute recursively with Eq. 7.12.

7.5 Computer Vision

The next factor we need to derive for our MHE problem (Eq. 7.4) is used for fusing visual information. At a high level, we assume that we can observe the same landmark several times over subsequent image frames. We use the pixel measurements to these landmarks to both infer the inverse depth to the landmark in the first frame that it is observed, and the translation and rotation between subsequent frames. In this section, we will first describe the image processing involved in generating landmark measurements and then derive the projection factor used in our MHE problem.

7.5.1 Image Processing

We use a global-shutter camera rigidly attached to the MAV with hardware synchronization between the IMU and camera shutter. We assume pinhole camera geometry, and that observed landmarks are stationary.

Features are extracted using a Shi-Thomasi corner detector [132], and are tracked from frame to frame with Lucas-Kanade pyramidal optical flow [133]. Tracked features are then filtered with two methods. The first checks how close features are to one another. If two features are found to be too close, one of them is dropped so we can acquire a new feature in a different part of the image. Next, the relative pose between the most recent keyframe and the current image is found up to a scale factor using the five-point RANSAC algorithm [134]. If a tracked feature is not found to be an inlier of the RANSAC pose solution, then it is also removed.

7.5.2 Keyframe Declaration

Like [86], we use the notion of *keyframes*, where features are compared to a keyframe image, and new keyframes are periodically declared based on specific criteria. New keyframes are selected under three conditions. First, the first image provided to the system is declared the first keyframe. Second, new keyframes are selected when the average parallax of tracked features exceeds some threshold after compensating for rotation. The average derotated parallax between frames i and j of N pixel measurements is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left\|\pi\left(R_j^i\pi^{-1}\left(\boldsymbol{\nu}_n^j\right)-\boldsymbol{\nu}_n^i\right)\right\|_2,$$

where π is the camera projection function which maps pixel measurements $\boldsymbol{\nu}^i$ into the associated unit vector $\boldsymbol{\zeta}_{i/c}^c$ as in

$$\pi\left(\boldsymbol{\nu}^i\right)=\boldsymbol{\zeta}_{i/c}^c,$$

and R_j^i is the rotation between frames. Third, we also select keyframes when the number of tracked pixels between keyframes drops below a specified value.

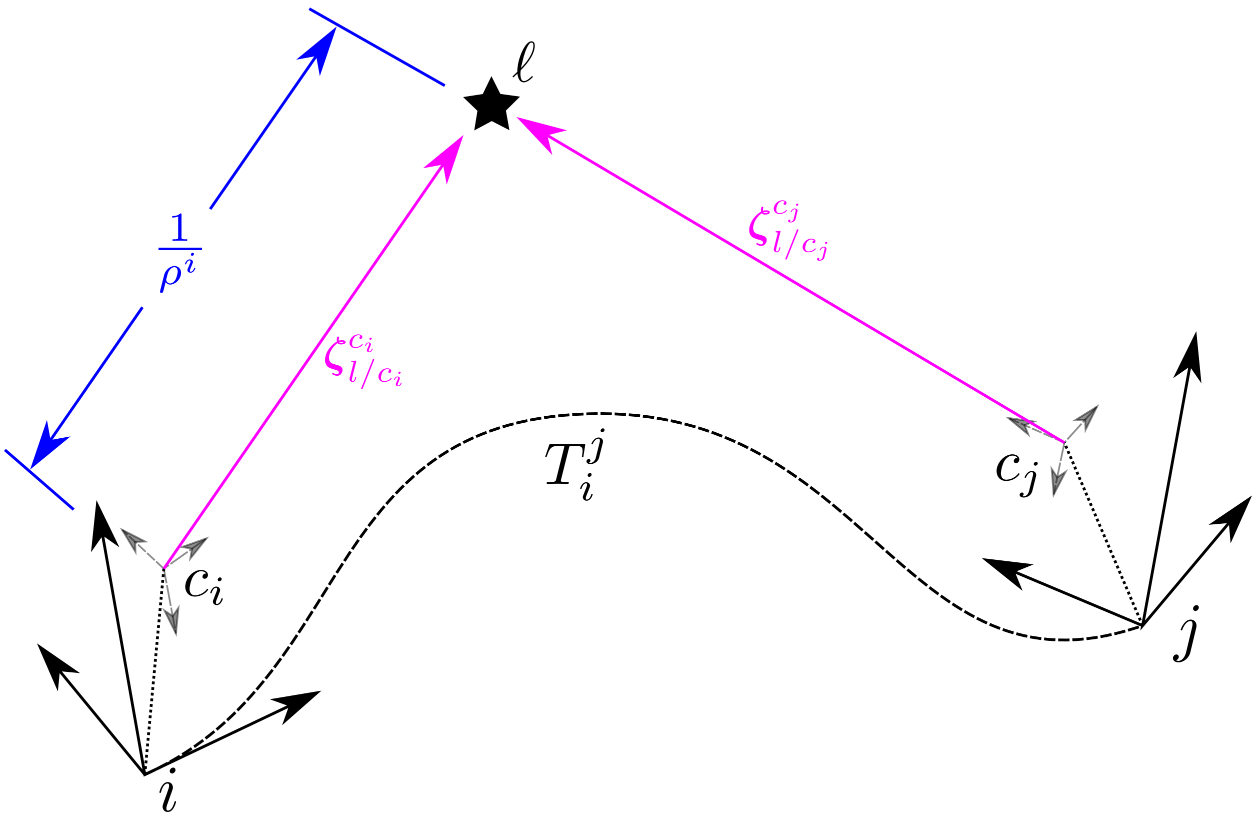


Figure 7.7: Projection factor geometry

7.5.3 Projection Factor Derivation

The measurement model for the second observation of a landmark ℓ at some coordinate frame j after the first observation in frame i is given as

$$\begin{aligned}
\hat{\zeta}^{c_j}_{\ell/c_j} &= R^{c_j}_I \left(\mathbf{p}^I_{\ell/I} - \mathbf{p}^I_{c_j/I} \right) \\
&= R^{c_j}_j R^j_I \left[\mathbf{p}^I_{i/I} + \mathbf{p}^I_{\ell/i} - \left(R^I_j \mathbf{p}^j_{c_j/j} + \mathbf{p}^I_{j/I} \right) \right] \\
&= R^{c_j}_j R^j_I \left[\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{\ell/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right] - \mathbf{p}^j_{c_j/j}.
\end{aligned}$$

We project the residual for this measurement onto the plane normal to $\bar{\zeta}^{c_i}_{\ell/c_i}$ and get

$$r_\zeta = \mathbb{P}_{\bar{\zeta}} \left(\bar{\zeta}^j_{\ell/j} - \hat{\zeta}^j_{\ell/j} \right), \tag{7.13}$$

where

$$\mathbb{P}_{\zeta} = \left[\frac{\zeta \times \mathbf{e}_3}{\|\zeta \times \mathbf{e}_3\|}^{\top} \quad \frac{\zeta \times \zeta \times \mathbf{e}_3}{\|\zeta \times \zeta \times \mathbf{e}_3\|}^{\top} \right]^{\top}.$$

The Jacobians for the projection factor are derived in Appendix 7.10.2.

The covariance of our feature bearing measurement Σ_{ζ} is computed based on the physical capabilities of the camera and the operating environment. With the residual and covariance, we can now fully compute the projection factor to put into Eq. 7.4 as

$$f_{\zeta} = \frac{1}{2}\mathbf{r}_{\zeta}\Sigma_{\zeta}^{-1}\mathbf{r}_{\zeta}^{\top}.$$

7.6 GNSS Measurement Processing

The last factors we need to derive are used for fusing GNSS measurements. We assume that we receive pseudorange and pseudorange rate measurements from satellites with known positions. This information allows us to infer our global pose and velocity, and given our other relative measurements (IMU and vision), the global pose of the origin frame. In this section, we first give a brief overview of GNSS signal processing. We then derive the relevant factors used in our implementation, both with and without a switching factor used to mitigate multipath effects. Finally, we describe some important details for successfully fusing global GNSS information with the relative information supplied by IMU and vision.

7.6.1 Pseudorange Measurement

The pseudorange measurement to a satellite s is given by the following model:

$$\sigma = \begin{bmatrix} \hat{\sigma}_s \\ \dot{\hat{\sigma}}_s \end{bmatrix}$$

$$= \begin{bmatrix} \left\| \mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E} \right\| + \frac{1}{c}\boldsymbol{\omega}_E^\top \left\lfloor \mathbf{p}^E_{s/E} \right\rfloor_\times \mathbf{p}^E_{g/E} + \boldsymbol{\delta}^{atm} + c\left(\tau_b - \tau_s\right) \\ \left\| \dot{\mathbf{p}}^E_{s/E} - \dot{\mathbf{p}}^E_{g/E} \right\| + \frac{1}{c}\boldsymbol{\omega}_E^\top \left\lfloor \mathbf{p}^E_{s/E} \right\rfloor_\times \dot{\mathbf{p}}^E_{g/E} + \frac{1}{c}\boldsymbol{\omega}_E^\top \left\lfloor \dot{\mathbf{p}}^E_{s/E} \right\rfloor_\times \mathbf{p}^E_{g/E} + c\left(\dot{\tau}_b - \dot{\tau}_s\right) \end{bmatrix}, \tag{7.14}$$

where $\boldsymbol{\delta}^{atm}$ is modeled by the sum of the Klobuchar [135] and Saastamoinen [136] atmospheric models, $\boldsymbol{\omega}_E$ is the rotation of the earth, c is the speed of light, and τ_b, τ_s are the clock bias of the receiver and satellite, respectively. Therefore, the naive pseudorange residual (without the switching factor) is given by

$$\mathbf{r}_{\sigma_s} = \bar{\boldsymbol{\sigma}}_s - \hat{\boldsymbol{\sigma}}_s, \tag{7.15}$$

and is weighted in Eq. 7.4 by the measurement covariance Σ_σ.

In implementation, if we assume that the receiver contains a decent estimate of the current time (i.e., the initial clock bias is low), then we can assume that a linear approximation of the effects of the atmospheric error, Sagnac compensation, rotation of the earth, and the satellite velocity and position are sufficient for the residual calculation.

7.6.2 Clock Bias Dynamics

In Eq. 7.14, we see a strong dependence upon accurate estimates of the receiver clock bias τ_b and its derivative $\dot{\tau}_b$. The satellite clock bias and clock bias rate are given by the broadcast ephemeris, but the stability of a consumer-grade GNSS receiver cannot be guaranteed to nanosecond precision, and instead must be estimated. We therefore estimate the clock bias for each state in our moving horizon and use the following state transition error function from time t to $t + \delta t$:

$$\mathbf{r}_\tau = \begin{bmatrix} \tau_b[t] + \delta t\, \dot{\tau}_b[t] \\ \dot{\tau}_b[t] \end{bmatrix} - \begin{bmatrix} \tau_b\left[t + \delta t\right] \\ \dot{\tau}_b\left[t + \delta t\right] \end{bmatrix}. \tag{7.16}$$

This residual is then weighted by Σ_τ, which is chosen based on the characteristics of the GNSS receiver, and is used in Eq. 7.4.

7.6.3 Switching Pseudorange Measurement Residual

As shown in [108], use of the naive pseudorange residual (Eq. 7.15) can introduce large estimation errors in the presence of multipath signals. This is because the distribution of multipath signals is very poorly approximated by the normal distribution assumed in the derivation of our MHE problem (Eq. 7.3). Therefore, because we intend to operate in a multipath environment, we utilize the switching pseudorange factor described in [108]. This is done by adding a parameter $\kappa \in [0, 1]$ that the optimizer can use to opportunistically remove pseudorange residuals from the optimization. To avoid the case that the optimizer simply removes all pseudorange measurements, we also augment the residual with a penalty for removing the pseudorange from the optimization. This logic is codified in the following expression for the switching pseudorange residual as

$$\mathbf{r}'_{\sigma_s} = \begin{bmatrix} \kappa_s \left(\boldsymbol{\sigma}_s - \bar{\boldsymbol{\sigma}}_s \right) \\ 1 - \kappa_s \end{bmatrix}. \tag{7.17}$$

The covariance used to weight Eq. 7.17 is given by

$$\Sigma'_\sigma = \begin{bmatrix} \Sigma_\sigma & 0 \\ 0 & \Sigma_\kappa \end{bmatrix},$$

where Σ_κ, is chosen by manual tuning. In our implementation, the bounds on κ_s are enforced by the optimization algorithm and the same clock bias dynamic constraint (Eq. 7.16) is applied when using either the naive form or switching form of the pseudorange residual. For reference, the Jacobians of Eq. 7.17 are derived in Appendix 7.10.1

Like [108], we also assume that if a previous measurement was invalid, then there is a high likelihood that the next measurement will also be invalid. To avoid our optimization switching too often, we apply the following dynamic constraints to the switching factors applied to the same satellite:

$$\mathbf{r}_{\kappa} = \kappa_s\left[t + \delta t\right] - \kappa_s\left[t\right].$$

This residual is weighted by $\Sigma_{\dot{\kappa}}$, which is also chosen by manual tuning. Although the switching factor introduces new parameters in the optimization and therefore increases the solution time, it has been shown to significantly stabilize the optimization and yield smooth results (see Sec. 7.8.1,7.8.2, [108, 109]) and is therefore worth the extra computational cost when operating in multipath environments.

7.6.4 Global and Relative Information Fusion

When simultaneously fusing GNSS with inertial and visual measurements, we must carefully consider how to deal with the difference between global and relative information. Because satellite positions (and therefore pseudorange measurements) are defined in the ECEF frame, the GNSS measurements supply us global information. The inertial and visual measurements provide relative information, defined in the local frame. To fuse this information cohesively, we must augment our estimation approach with the location of the ECEF frame with respect to our inertial coordinate frame, but there is some question about whether to pin our estimation approach at the ECEF origin, or at some locally-defined frame, such as the trajectory origin.

Initial attempts at this fusion pinned the estimation approach at the ECEF frame, and attempted to estimate the location of the inertial frame (Figure 7.8, left). The problem with this approach is that GNSS measurements can induce discontinuities in the relative transforms used by the IMU and vision residuals. This can degrade the performance of the nonlinear least-squares solution and cause it to converge to local minima. We found it was more robust to hold the inertial frame constant, and instead estimate the transform between the origin and the ECEF frame (Figure 7.8, right).

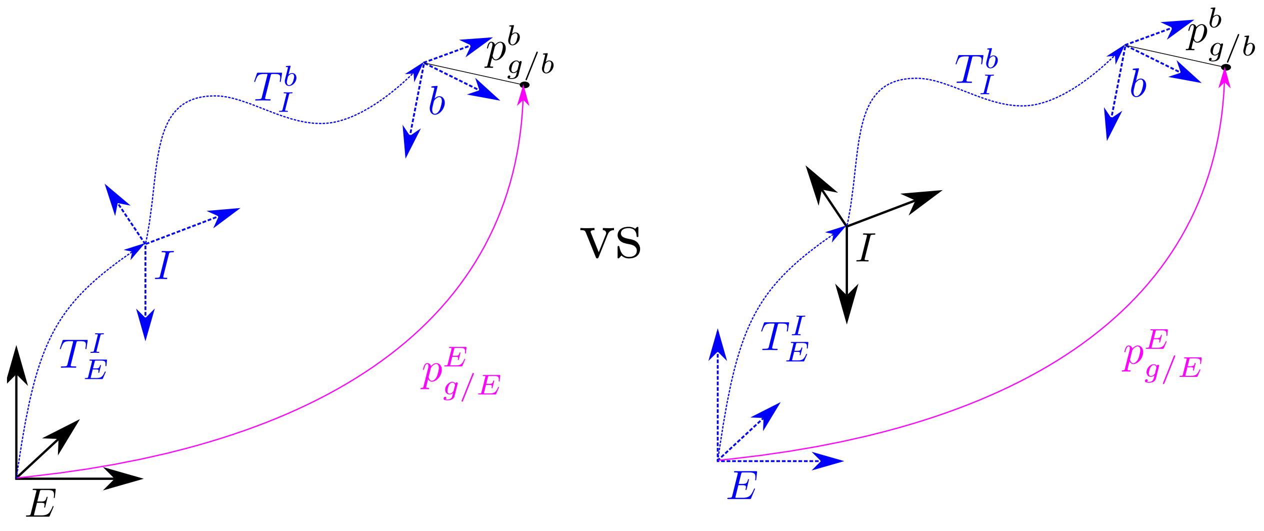


Figure 7.8: Illustration of some of the coordinate frames used in GV-INS state estimation. Frames drawn in blue dashed lines are estimated, while black lines are fixed. Our original approach is shown on the left, where the ECEF frame was held constant. This induced large jumps in relative position, that caused poor performance when also optimizing relative vision and IMU information. The approach on the right was found to be much more effective.

7.7 Factor Graph Management

Now that we have derived the factors for the three main components of our MHE approach, we can discuss the actual implementation of our factor graph. Because the nodes of our factor graph are the optimization design parameters, having too many nodes could increase the solver time beyond real-time constraints, but having too few could shorten our window enough that insufficient motion is contained in the window to properly observe all the desired states. In addition to node management, another important detail to acheiving good performance in vision-aided MHE approaches is the initialization of the IMU bias estimate. In this section, we first discuss the management of our factor graph so that we can meet real-time requirements given the computational limitations of our MAV. We then describe the initialization strategy used in our hardware experiments. We found that both of these details were critical to good estimation performance given our platform, sensor suite and environment.

7.7.1 Node Declaration

In our graph, we carry a window of *base states*, that contain the position, velocity, attitude and clock bias at some time t. $\left(\mathbf{x} = \begin{bmatrix} \mathbf{p}^I_{b/I} & \mathbf{v}^b_{b/I} & \mathbf{q}^b_I & \tau \end{bmatrix}^\top\right)$, a single IMU bias node for our entire window $\left(\mathbf{b} = \begin{bmatrix} \mathbf{b}_a & \mathbf{b}_\omega \end{bmatrix}^\top\right)$ and a node for the transform from the earth origin to our trajectory origin (T^I_E). We also carry a node for the inverse depth to each feature at the time it is first seen in our window (ρ_i) and the switching parameter (κ_i) for every pseudorange measurement.

Because we must have a node to tie measurements to in our graph, we must have nodes both for GNSS and camera measurements. However, having both sensors feeding our state estimator significantly increases the number nodes in our window and has a detrimental effect on real-time performance. Therefore, in contrast to [86], instead of creating a new node at every image and removing factors that do not connect to keyframes, we create a new node after a keyframe declaration, but then re-use it from frame to frame until either a GNSS measurement occurs, or until another keyframe is declared (See Figures 7.9 and 7.10). This significantly reduces the number of nodes held in the window and makes optimization with both sensors manageable.

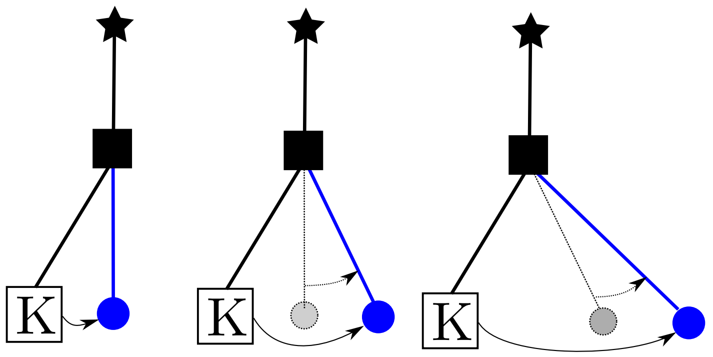


Figure 7.9: Diagram of the creation of nodes over a keyframe interval without GNSS. A new node is created on the first image after a keyframe is declared, and this node is moved at every subsequent image until a new keyframe is declared.

7.7.2 Node Removal

To ensure real-time performance, we must also remove nodes from the sliding window as they become too old. In our sliding window, we hold either a maximum number of keyframes,

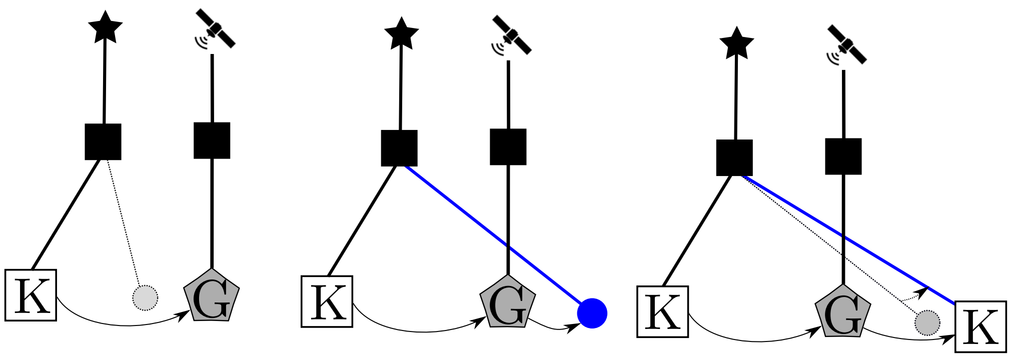


Figure 7.10: Diagram of the creation of nodes over a keyframe interval with GNSS. A new node is created at either the first image or GNSS measurement after either a keyframe is declared or after a GNSS measurement. This node is moved at every subsequent image until a new node is declared.

or maximum number of nodes, whichever results in the fewest number of total nodes. We do this because GNSS measurements can often accrue much faster than keyframes, and limiting our window on keyframes alone can result in poor real-time performance.

Removing keyframes from our sliding window is complicated by the fact that feature projection factors are all tied to both an origin keyframe, or the *anchor*, and also the subsequent keyframes. This is further complicated because we estimate the inverse depth to the feature in the anchor frame. Therefore, if we wish to remove a keyframe from the window that serves as an anchor to keyframes still in the window, we need to slide the anchor from the keyframe to be removed to the next keyframe in the window. This is illustrated in Figure 7.11.

There are two main phases to moving an anchor from frame i to frame j. The first step is to calculate the inverse depth ρ^j in the new anchor frame using the current estimate of ρ^i as

$$\begin{aligned}\frac{1}{\rho_j}\zeta^{c_j}_{l/c_j} &= R^{c_j}_{j}\left(R^{j}_{I}\left(R^{I}_{i}\left(R^{i}_{c_i}\left(\frac{1}{\rho^i}\zeta^{c_i}_{l/c_i}\right)+\mathbf{p}^{i}_{c_i/i}\right)+\mathbf{p}^{I}_{i/I}-\mathbf{p}^{I}_{j/I}\right)-\mathbf{p}^{j}_{c_j/j}\right)\\ \frac{1}{\rho^j} &= \left\|R^{c_j}_{j}\left(R^{j}_{I}\left(R^{I}_{i}\left(R^{i}_{c_i}\left(\frac{1}{\rho^i}\zeta^{c_i}_{l/c_i}\right)+\mathbf{p}^{i}_{c_i/i}\right)+\mathbf{p}^{I}_{i/I}-\mathbf{p}^{I}_{j/I}\right)-\mathbf{p}^{j}_{c_j/j}\right)\right\|.\end{aligned}$$

The second step is to update all the factors with the new $\zeta^{c_j}_{l/c_j}$ from the new anchor.

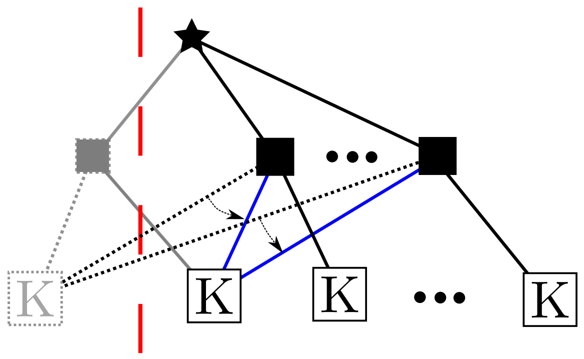


Figure 7.11: Diagram of the management of nodes in the sliding window. If keyframes are removed that are serving as anchors for future measurements, the factors are recomputed after moving the anchor to next keyframe in the window.

7.7.3 Graph Initialization

As noted in other V-INS literature, proper initialization of a visual-inertial MHE system is crucial to good performance. V-INS systems are particularly sensitive to the initialization of IMU biases, so in [86], graph initialization takes place through a rigorous three-step process that includes mapping the local environment up to a scale factor, then using this map to resolve the gyroscope and accelerometer biases. Because we are performing our state estimation onboard a MAV, we assume that the agent is not moving so long as propellers are not spinning. Therefore, we adopt a much simpler *static start* procedure.

To perform the static start, we turn on the system and wait for approximately 30 seconds before liftoff. During this period, we disable image updates and instead add residuals that enforce a zero-position, velocity, and yaw constraint. If we assume that the MAV starts at the origin with zero yaw, then the static start residual is given as

$$\mathbf{r}_0 = \begin{bmatrix} \mathbf{p}^I_{j/I} \\ \mathbf{v}^I_{j/I} \\ \psi^j_I \end{bmatrix},$$

where ψ is the current yaw angle. This residual is weighted by Σ_0, which is tuned manually. GNSS factors may also be active during the static hold period if GNSS measurements are available to initialize T_E^I. Initial attitude is impossible to determine precisely during the static start, as there is ambiguity between accelerometer bias and roll and pitch, however, the accelerometer scale and gyroscope biases very quickly converge to their true values. The static hold period is finshed upon receiving an IMU measurement with magnitude greater than a specific threshold, which we assume to be the take-off event. After this point, the static hold residuals are no longer applied.

Finally, because our MAV begins with the camera lens less than an inch from the ground, the first few images are typically out of focus. Therefore, after the static start, we keep the image updates disabled for a short period until we get high enough that we can observe trackable landmarks. After we start tracking a sufficient number of useful landmarks, then the system is fully initialized and image updates are enabled

7.8 Results

In the following sections, we evalulate the performance of our described approach against the state-of-the art in V-INS [86] and a sliding-window, inertially-aided adaptation of [108, 109], which we believe to be equivalent to state-of-the-art in G-INS with a consumer-grade, single frequency receiver. Two other estimators were also used to show the influence of the switching parameter and visual fusion. These five estimators, their differences and identifiers are enumerated in Table 7.1.

Comparison studies were performed both in simulation (Section 7.8.1) and in a hardware experiment (Section 7.8.2). In both the simulation and hardware studies, all measurements (feature pixel locations, IMU and raw GNSS measurements) were fed to each of the five estimators. However, in the case of hardware results, only the GV+κ estimator was run in real-time, while results from all other estimators were post-processed using recorded data.

Table 7.1: Description and identifiers of estimator variations used in experimentation.

G	GNSS-only with naive version of the pseudorange measurement model (Eq. 7.15).
G+κ	GNSS-only sliding window adaption of [108, 109] with inertial fusion.
V	Vision only. Implementation of [86].
GV	GNSS and vision, but using the naive pseudorange measurement model.
GV+κ	GNSS and vision, as well as the switching parameter pseudorange model given by Eq. 7.17.

7.8.1 Simulation Results

Simulation results are instructive in this case, as the only realistic reference truth for hardware results is a high-end inertial navigation system or LIDAR-based SLAM, which are both much too heavy for our MAV. Furthermore, simulation studies allow us to isolate the differences in various systems, and control for various variables of interest. Finally, we also found that the simulation study was useful for evaluating the switching parameter performance, as real-life multipath in hardware experiments can be difficult to identify.

7.8.1.1 Simulation Description

Our simulator employs a nonlinear drag model for our multirotor agent, an IMU with a slowly-wandering bias and additive Gaussian noise, and a downward-facing camera. Simulated landmarks are tracked perfectly (i.e. no data association errors), but Gaussian noise is added to the individual pixel measurements.

GNSS pseudorange and pseudorange rate measurements are simulated by placing satellites according to broadcast ephemeris data collected in hardware and simulating the measurement with the model given in Eq. 7.18. We then add Gaussian noise to arrive at the simulated pseudorange and pseudorange rate. As the multirotor enters certain areas of our simulated environment, multipath error to satellites is simulated using Algorithm 1. These areas are placed adjacent to GNSS-denied areas where measurements are not generated at all.

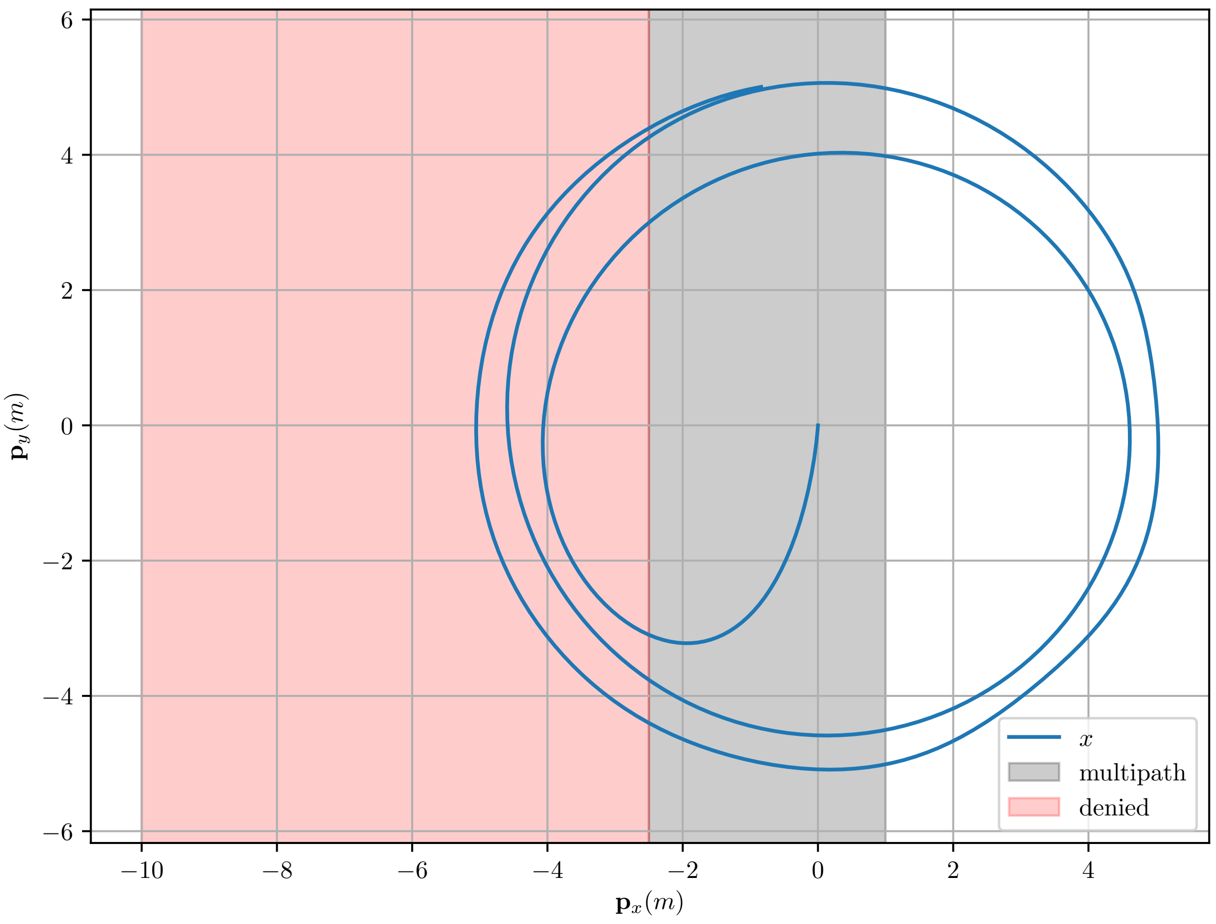


Figure 7.12: Top-down view of simulated trajectory showing multipath and GPS denied areas

We simulated the quadrotor starting at rest at the origin, then flying a 5 meter, circular orbit around the origin. The orbit also included a rise and fall in altitude of 4 meters, and a slow oscillation in heading of approximately 90 degrees. This trajectory was chosen to excite all the relevant motions ideal for monocular SLAM. Landmarks were chosen such that features in the image were roughly evenly spaced. A multipath region was placed at $\mathbf{p}_x \in [-2.5, 1]$ and GNSS measurements were denied when $\mathbf{p}_x \in (-\infty, -2.5)$. The MAV started the simulation at $\mathbf{p} = \mathbf{0}$, which was inside the multipath area. Figure 7.12 shows a top-down view of this trajectory, including the multipath and GNSS-denied regions.

Algorithm 1 Simulation of multipath in simulation environment. $P(m)$ is the probability of a multipath measurement, $\max(m)$ is the maximum multipath error and σ_s is calculated using Eq. 7.18. The simulated measurement $\hat{\sigma}_s$ includes the multipath error and noise.

```
1: for $s \in$ Satellites do
2:     if $\mathbf{p}_{b/I}^{I} \in$ Multipath Area then
3:         if $m_s = 0$ and $sample(\mathcal{U}(0,1)) < P(m)\,\delta t$ then
4:             $m_s = sample(\mathcal{U}(0, \max(m)))$
5:     else
6:         $m_s = 0$
7:     $\hat{\sigma}_s = \sigma_s + m_s + sample(\mathcal{N}(0, \Sigma_\sigma))$
```

7.8.1.2 Results

The simulation results illustrate the effect of the different variations in estimator design. As expected, Figures 7.13 and 7.14 show that prolonged periods of GNSS outage result in extreme position and velocity errors in the GNSS-only estimators G and G+κ. However, after initial convergence, the G+κ estimator is quicker to return to the true trajectory when GNSS measurements are restored, even if they are multipath measurements.

Position and velocity estimation using vision alone has been well demonstrated in existing work [86]. Our performance is comparable, resulting in decent velocity estimation, but wandering position and heading. This is expected, as absolute position and heading are unobservable given vision and inertial measurements only. However, a bit surprising are the results from the combination of vision and GNSS without the switching parameter GV. Without the switching parameters, multipath causes significant jumps in the position estimation and degrades the whole solution.

The GV+κ estimator performs the best in this simulation. The vision component allows the estimator to perform well when denied GNSS, and multipath is better identified in multipath areas than the G+κ estimator. At some points during the simulation the G+κ estimator has smaller position error, however, the vision-aided solution has a more accurate solution overall and consistently does a better job of estimating velocity, even where good GNSS measurements are available. This is important for MAVs that perform feedback control with velocity estimates.

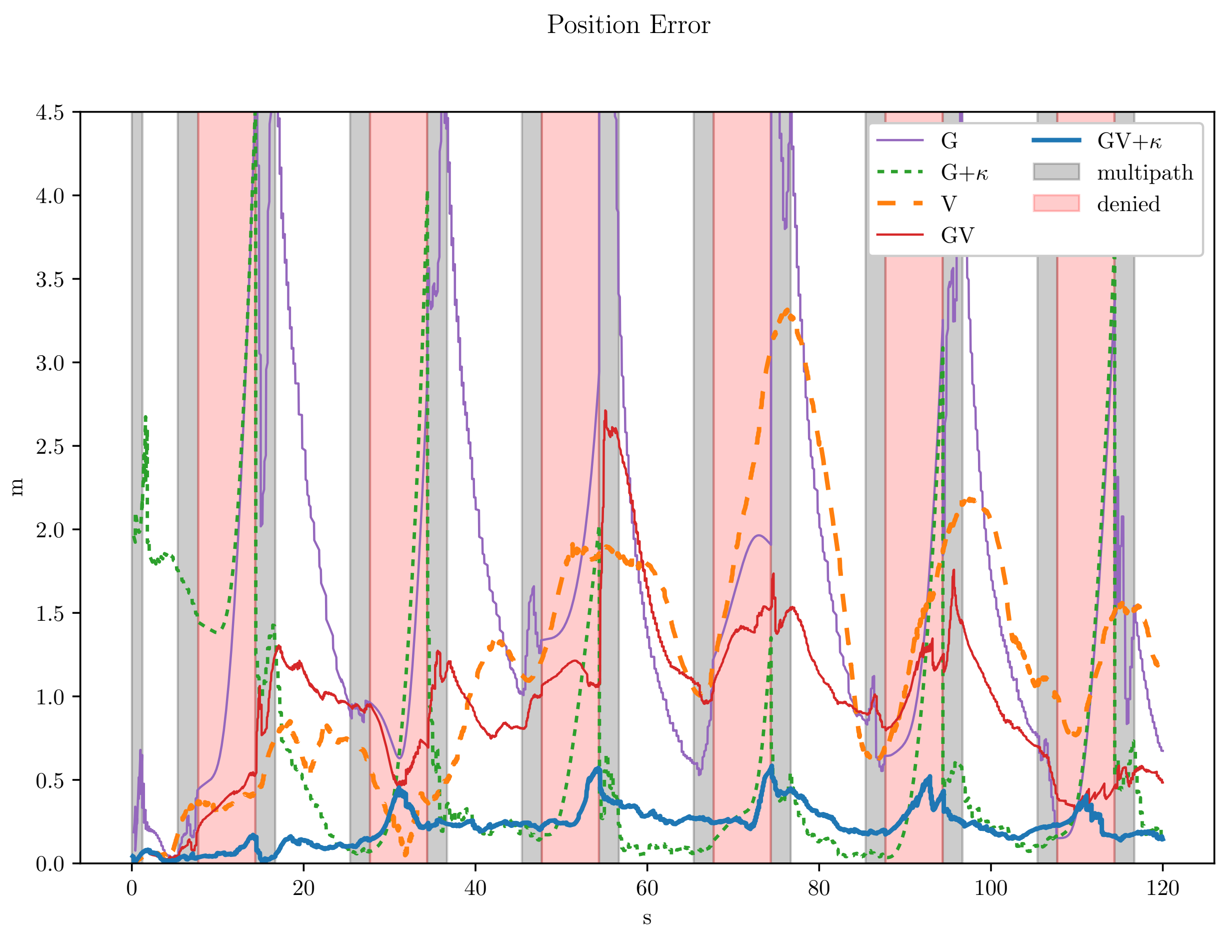


Figure 7.13: Position error of the five estimators in the simulation study

Switching parameter performance observed is similar, if not better than shown in [108, 109], although this is difficult to quantify as neither [108] or [109] performed simulation studies to quantify their switching parameter estimation performance. However, we suspect that the inertial feedback in our approach makes multipath measurements quite costly to include as they introduce large discontinuities in position and velocity. Therefore, the switching architecture becomes even more advantageous in inertially-aided and vision-aided fusion of pseudorange. The estimates of κ in the GV+κ estimator for the first eight satellites are shown in Figure 7.15.

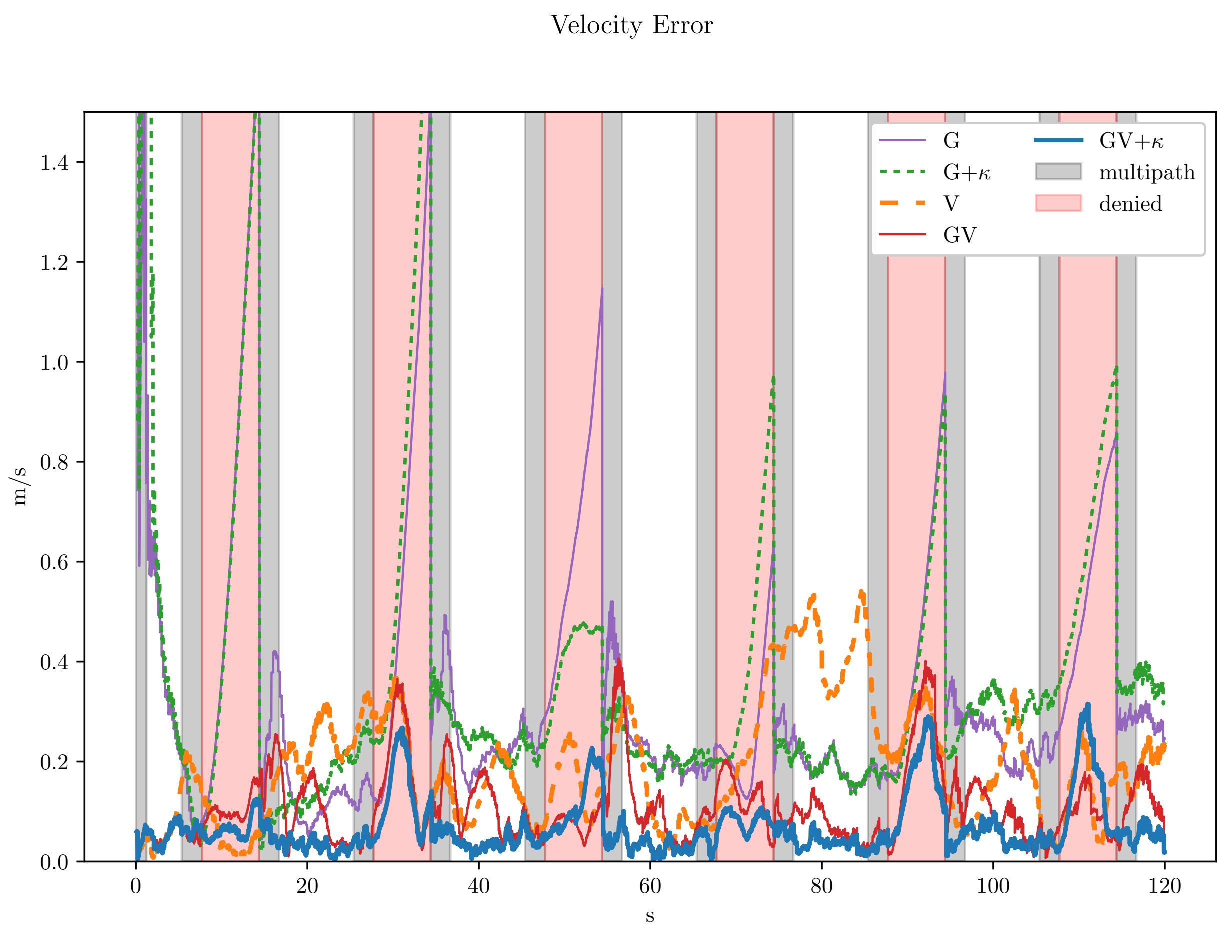


Figure 7.14: Velocity error of the five estimators in the simulation study

7.8.2 Hardware Results

Hardware experimentation was performed using the MAV shown in Figure 7.16. We used an InertialSense µINS3 for collecting raw GNSS measurements, and a MYNT Eye Standard stereo visual-inertial sensor. We used the left camera and IMU from the MYNT camera and removed the aluminum case and infrared light projectors to reduce weight. Computation was performed onboard using an NVIDIA TX2.

We configured the camera to report a 754×480 image at 10 Hz and inertial measurements at 200 Hz. The GNSS receiver was configured to report measurements at 5 Hz. The lower frame rate and image resolution were required because of our use of a USB 2.0 interface to communicate with

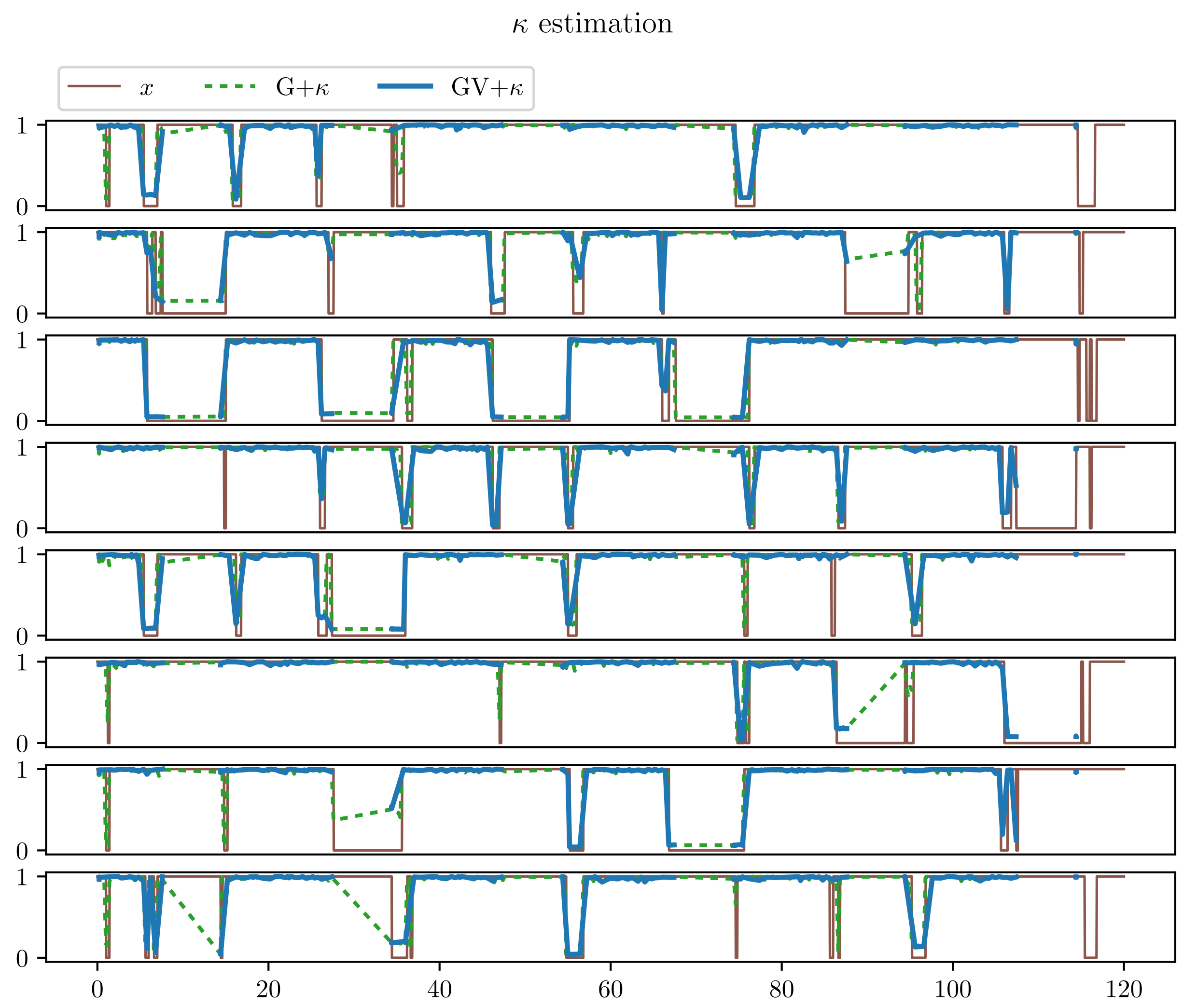


Figure 7.15: Switching parameter performance over the simulation with respect to the first eight satellites. (There were 15 satellites that all showed similar performance, but the plot is truncated for clarity). The blue line drops to zero if the measurement is truely multipath, green is the resulting κ value.

the camera. A USB 3.0 interface allows for higher framerate and higher resolution, however USB 3.0 jams GNSS signals at close range so we could not use it for this experiment. The camera was configured with auto-exposure enabled for the outdoor to indoor transition and was oriented at a 40 degree downward pitch from level.

The hardware experiment took place on the east side of the BYU engineering building. This building has a five story face on the east side, with a large alcove on the ground floor with loading bay access. There is also a large parking lot just off the east side where a clear GNSS signal is

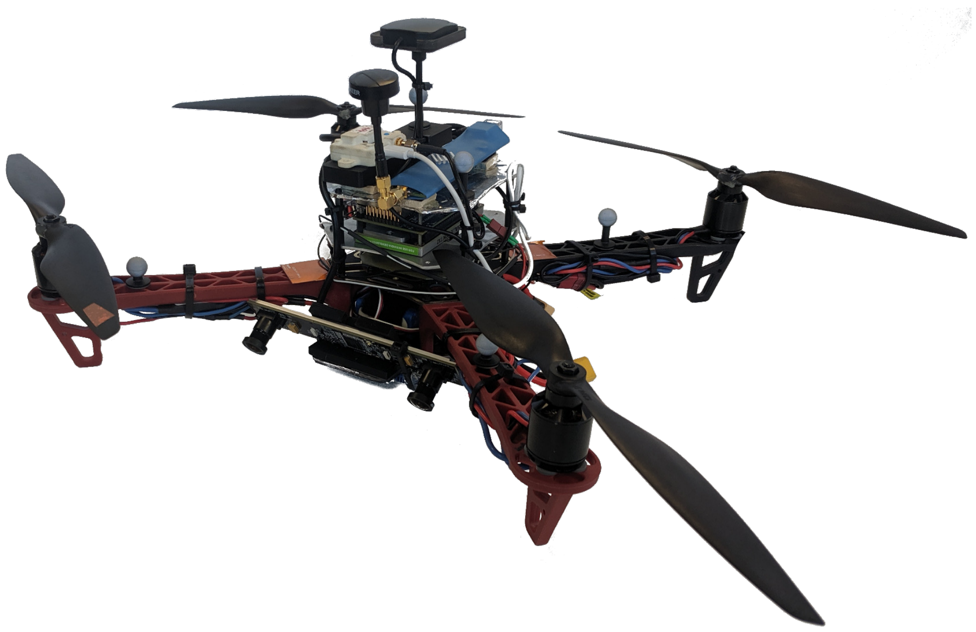


Figure 7.16: Multirotor used in hardware experiment

accessible. This area provides easy access to clear-sky conditions (parking lot) multipath (near the building) and GNSS-denied conditions (in the alcove) within a short distance of each other.

The MAV started on the sidewalk near the parking lot, where we waited for the broadcast ephemeris for eight satellites while performing the static start calibration. We then flew towards the building, into the alcove, and back out again, performing two loops, landing close to where the flight started.

The results of our experiment are shown in Figures 7.17 through 7.20, where the GV+κ estimates are plotted against the GNSS-only version (an inertially-aided, MHE adaptation of [108, 109]) and the vision-only approach [86] without loop closures. A video of the experiment can be found at `https://youtu.be/dUJOT0D3L3I`.

Figure 7.17 shows the top-down view of the estimated position for the G+κ, V and GV+κ estimators, and is the most instructive figure for identifying differences between the approaches. In this figure, we can see that the vision-only estimator accumulates some heading and scale error during initialization. This is normal for most VI approaches without loop closure [79]. Also, at

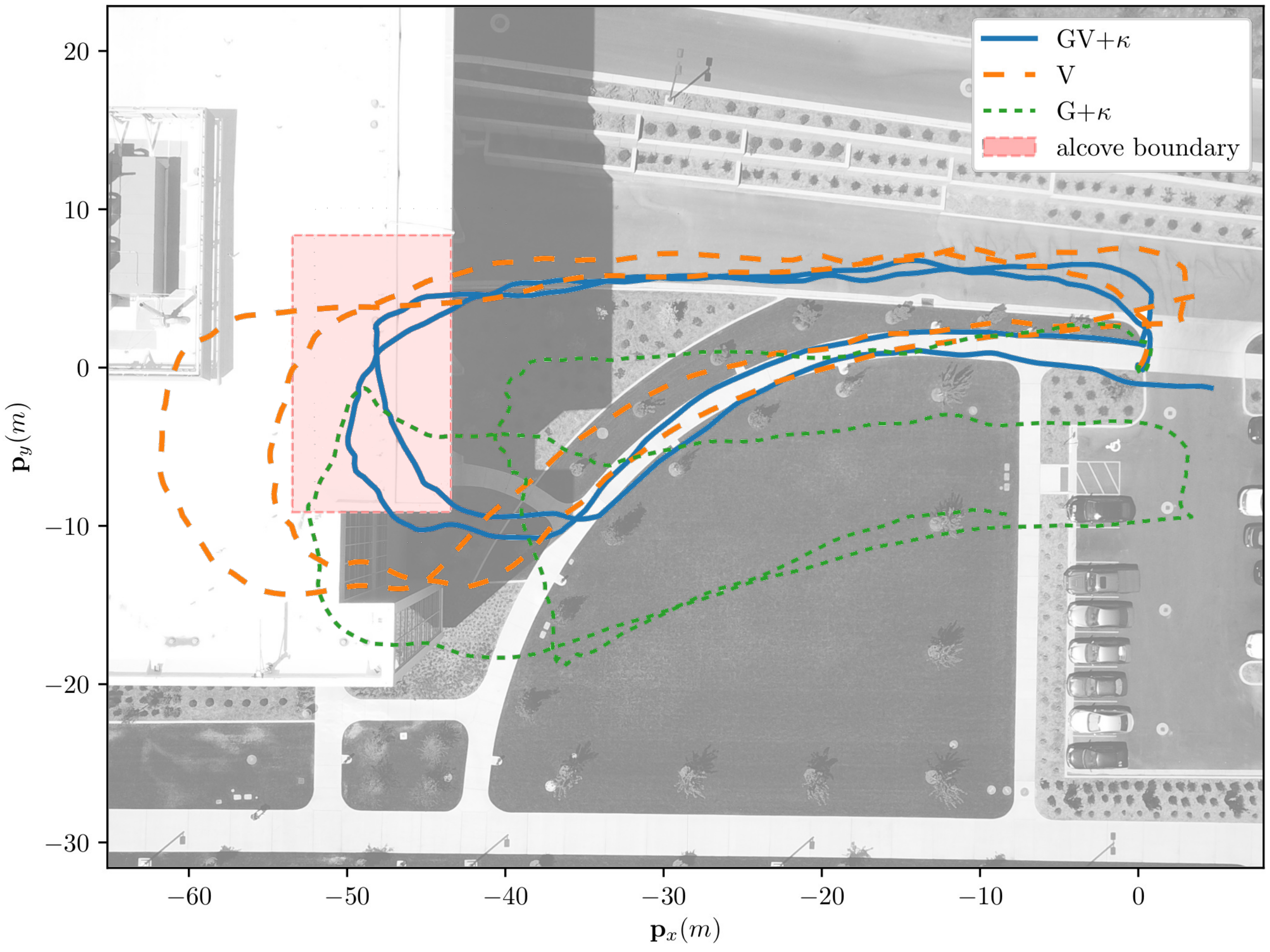


Figure 7.17: Position estimation results in hardware experiment. The shaded red area indicates the boundary of the alcove on the ground floor of the building.

around 150 seconds, feature tracking fails (this is best observed in the associated video), during which the vision-only estimator experiences accumulates velocity error (seen in Figure 7.18), so the position estimates of the second loop in Figure 7.17 for the V estimator are offset.

The G+κ estimator accumulates considerably more position error than the V and GV+κ estimators during the experiment. Firstly, in Figure 7.17, it appears that the G+κ estimator incorrectly estimated the initial transform from the ECEF origin to the trajectory origin T_E^I. During the flight, the error in this transform was pushed into the position estimates, which results in the offset loops observed in Figure 7.17. Furthermore, the G+κ estimator experiences significant position error

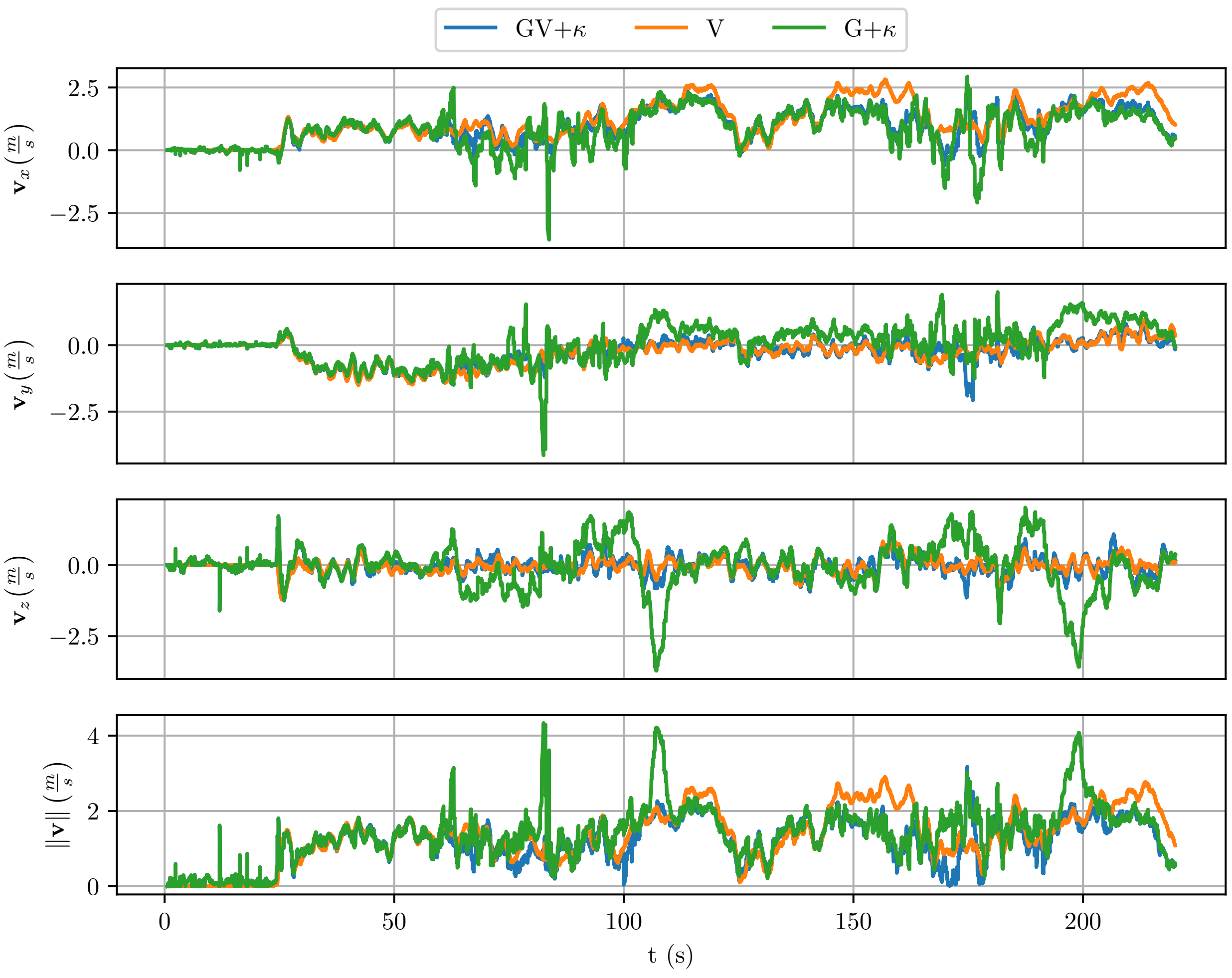


Figure 7.18: Velocity estimation results in hardware experiment

during the GNSS-degraded transition. This is because without GNSS, the G+κ estimator either must dead-reckon using IMU or fuse multipath GNSS measurements, neither of which produce good position or velocity estimates (see also the large velocity errors in Figure 7.18).

While the G+κ estimator does not perform as well as the V or GV+κ approaches, Figure 7.21 illustrates how introducing the switching parameter significantly improves position estimation. The G+κ estimator accumulates about 15 m of error at the worst point, while the G estimator without the switching parameter regularly experiences well over 100 m of error, and would likely be completely unusable for any autonomous operation in this environment.

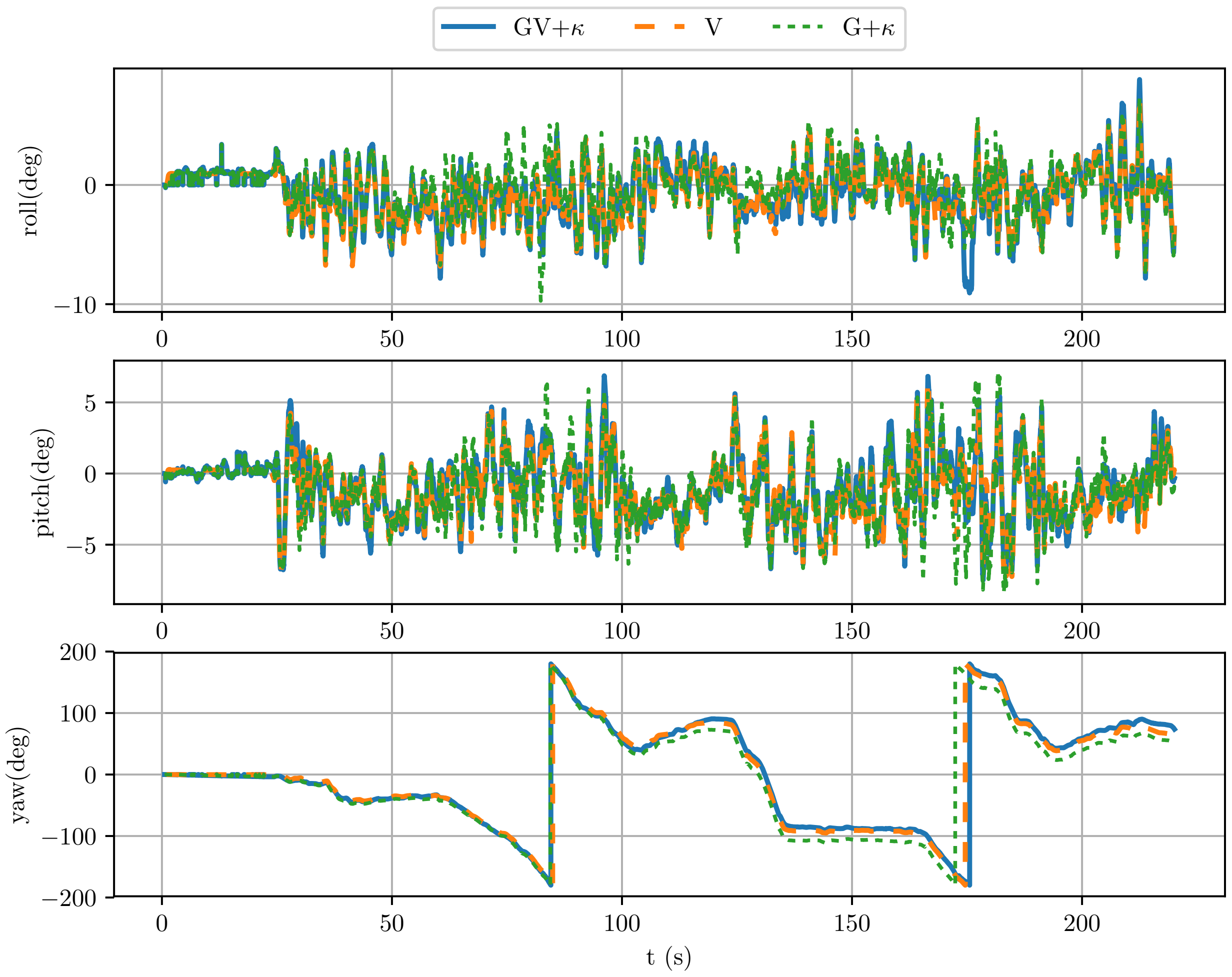


Figure 7.19: Attitude estimation results in hardware experiment

Finally, we can say that the GV+κ estimator performs the best in the hardware experiment, and that the hardware experiment validates our simulation results. Position, velocity and attitude estimation of the full proposed system GV+κ is superior to both the vision-only estimator (V) and the GNSS-only estimator with switching parameters (G+κ). This is likely due to three primary factors: First, the global nature of the GNSS measurement constrains the global drift that typically occurs in a visual-inertial system. Therefore, GV+κ does not accrue the same initial scale and heading bias that occurs in the visual-inertial system. Secondly, the doppler measurement of GNSS signals provides direct velocity feedback so the period of feature tracking failure does not significantly degrade velocity estimation like it does for the V approach. Finally, in contrast to the GNSS-inertial solution, the visual feedback makes multipath signals more expensive to include in

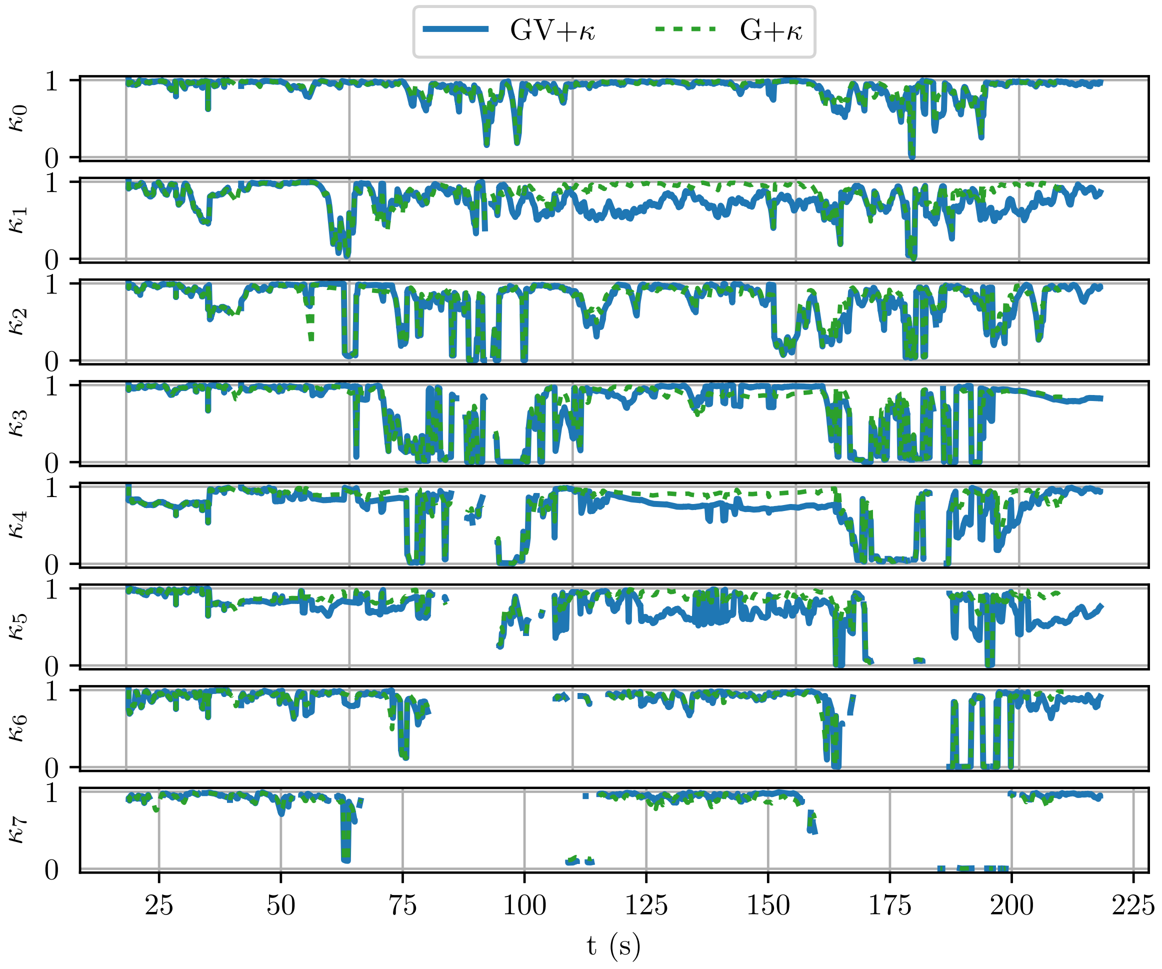


Figure 7.20: Switching parameter results for the first eight satellites in hardware experiment.

the solution so they are better identified by the switching parameter architecture. This happens because position jumps significantly increase the cost associated with the projection factor residuals

It is also clear from this experiment that the switching parameter is critical to good performance in the presence of degraded GNSS. Figure 7.21 shows the comparison of the GNSS-inertial estimator both with and without the switching parameter, and Figure 7.22 shows the number of satellites being fused during the different parts of the trajectory. Although we do not have access to accurate understanding of which GNSS measurements are experiencing multipath, we expect the number of valid satellite measurements to decrease as we approach the building. This is validated by Figure 7.22, where the number of satellites being fused smoothly drops to zero or one as we

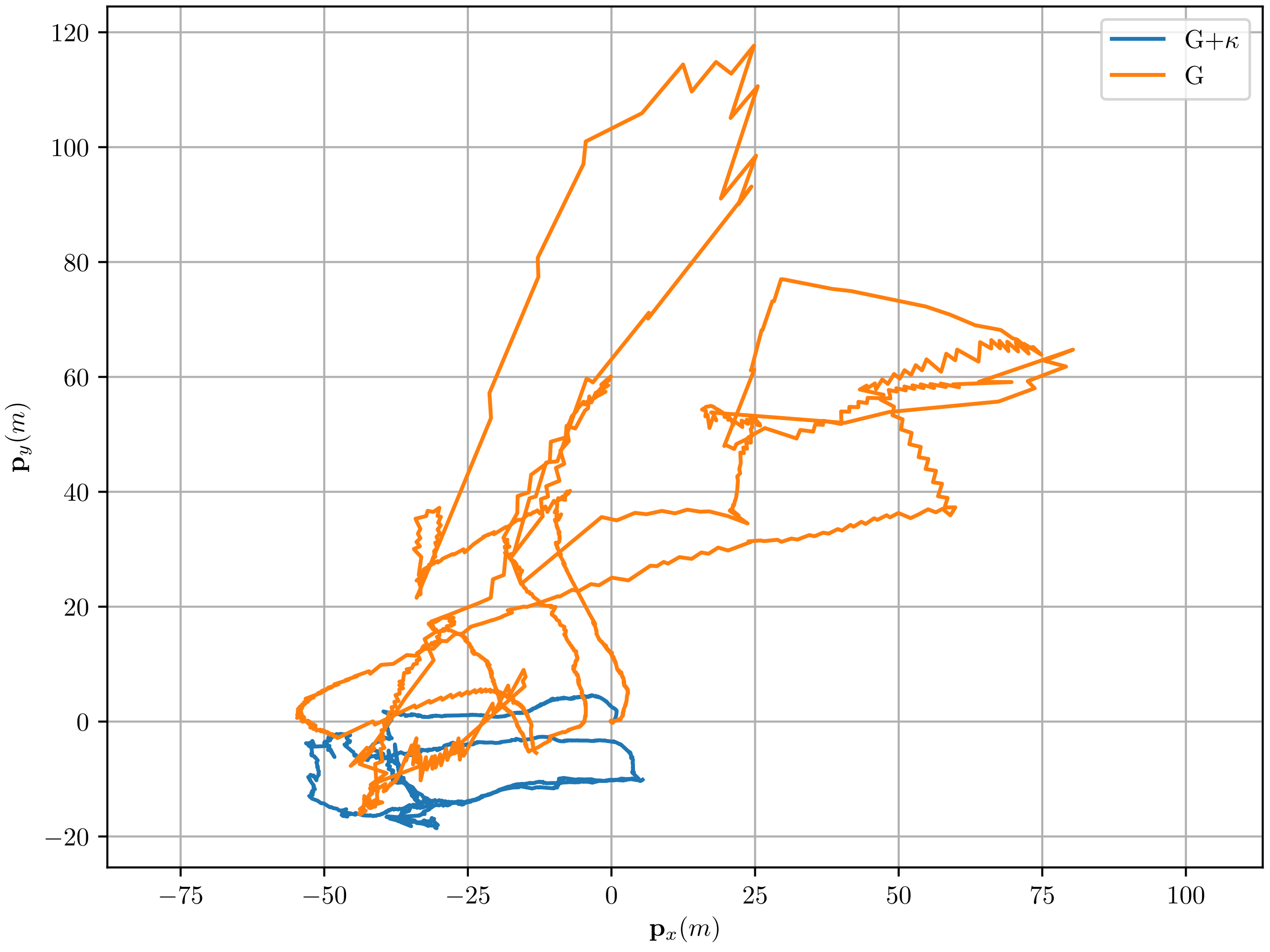


Figure 7.21: Comparison of GNSS-inertial position estimation with and without switching parameter

enter the building, and then climbs back up to eight near the parking lot. Because GV+κ is able to correctly identify multipath signals, it can correctly fuse global information from the GNSS to reduce the global drift experienced by V while at the same time using the visual information to correctly estimate position during GNSS outage. One interesting note is that the GNSS-Visual-inertial estimator without the switching parameter (GV) actually experienced divergent behavior about ten seconds into the flight and eventually accrues several kilometers of error. Upon close inspection of this data, we have concluded that multipath errors from the GNSS measurements cause large errors in depth estimation to all observable features. After this occurs, the solver gets caught in local minima and does not recover at any time during the flight. Because the errors in GV are so large, they have been omitted from the results plots for clarity.

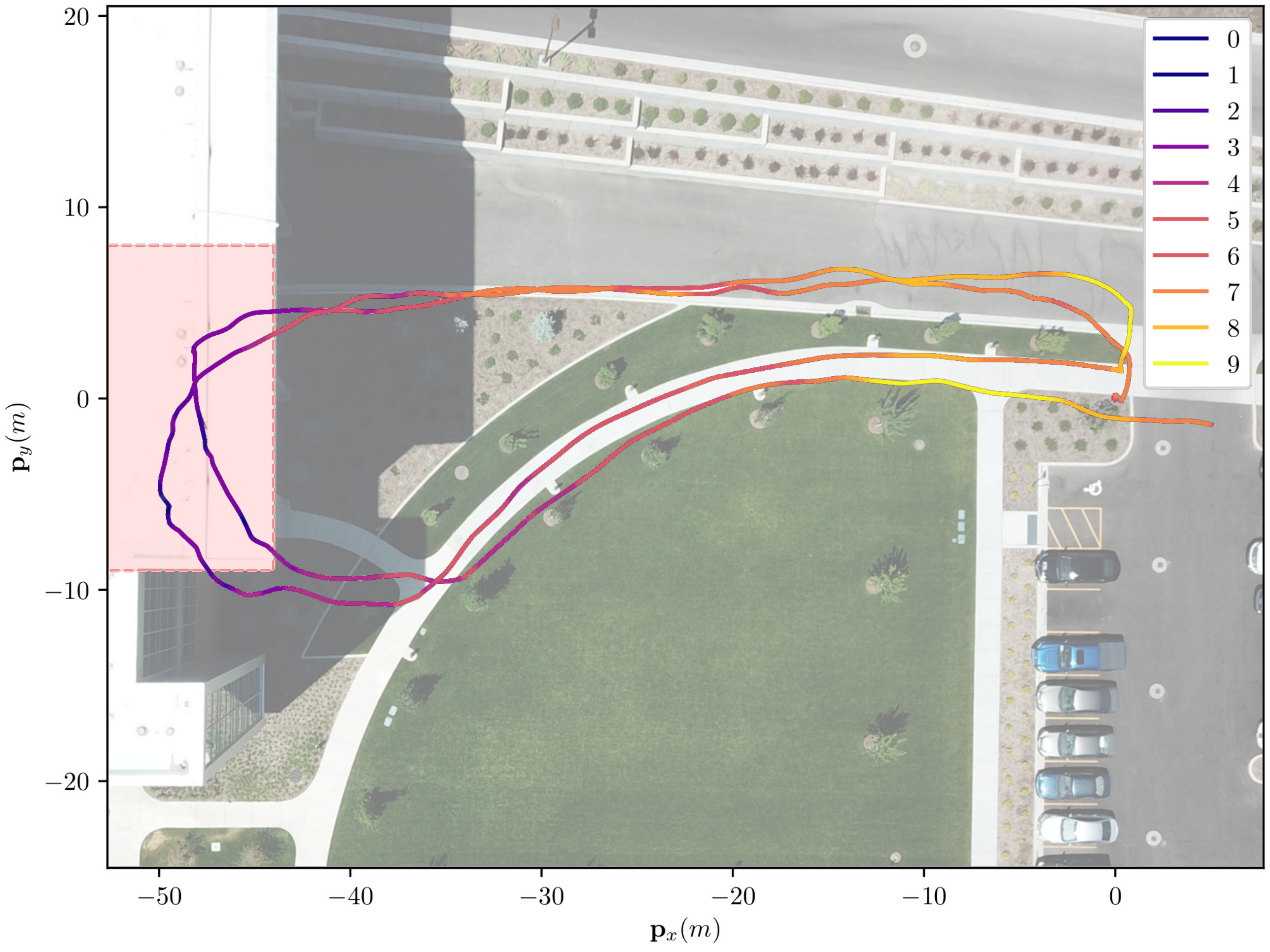


Figure 7.22: Number of satellites used in GV+κ solution during the trajectory. The shaded red area indicates the boundary of the alcove on the ground floor of the building

7.9 Conclusion

In this work, we present GV-INS, a method to fuse GNSS, visual and inertial information in a unified moving-horizon estimation framework. We demonstrated the algorithm in simulation where we could prove the effectiveness of the switching parameter framework to identify and reject multipath, and also in a hardware experiment where GV-INS was shown to be superior to both G-INS and V-INS alone.

Future work in this area could investigate the use of the carrier phase signal in the moving-horizon estimation framework as a relative measurement between subsequent states, as well as the incorporation of loop closures to provide even more global information and accuracy.

7.10 Appendix

The following sections derive the Jacobians for the pseudorange and projection factors that are required for efficient implementation of the GV-INS estimation framework.

7.10.1 Pseudorange Residual Jacobian

The residual of the pseudorange measurement with the switching factor is given in Eq. 7.17. For reference, this residual is given as

$$\mathbf{r}'_{\sigma_s} = \begin{bmatrix} \kappa_s \left(\boldsymbol{\sigma}_s - \bar{\boldsymbol{\sigma}}_s\right) \\ 1 - \kappa_s \end{bmatrix},$$

where

$$\begin{aligned} \boldsymbol{\sigma} &= \begin{bmatrix} \hat{\sigma}_s \\ \dot{\hat{\sigma}}_s \end{bmatrix} \\ &= \begin{bmatrix} \left\|\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}\right\| + \frac{1}{c}\omega_E^\top \left\lfloor \mathbf{p}^E_{s/E} \right\rfloor_\times \mathbf{p}^E_{g/E} + \delta^{atm} + c\left(\tau_b - \tau_s\right) \\ \left\|\dot{\mathbf{p}}^E_{s/E} - \dot{\mathbf{p}}^E_{g/E}\right\| + \frac{1}{c}\omega_E^\top \left\lfloor \mathbf{p}^E_{s/E} \right\rfloor_\times \dot{\mathbf{p}}^E_{g/E} + \frac{1}{c}\omega_E^\top \left\lfloor \dot{\mathbf{p}}^E_{s/E} \right\rfloor_\times \mathbf{p}^E_{g/E} + c\left(\dot{\tau}_b - \dot{\tau}_s\right) \end{bmatrix}. \end{aligned} \quad (7.18)$$

If we let the unit vector to the satellite be $\mathbf{e}_s = \frac{\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}}{\left\|\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}\right\|}$, define an intermediate calculation

$$Z = \frac{\partial}{\partial \mathbf{p}^E_{g/E}} \mathbf{e}_s = \frac{\left(\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}\right)\mathbf{e}_s^\top - I\left\|\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}\right\|}{\left\|\mathbf{p}^E_{s/E} - \mathbf{p}^E_{g/E}\right\|^2},$$

remember that

$$\dot{\mathbf{p}}^E_{g/E} = R^E_I R^I_b \mathbf{v}^I_{b/I},$$

and ignore Sagnac compensation in our Jacobian calculations, then the Jacobians of the residual are given by

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \mathbf{p}^I_{b/I}} = \kappa_s \begin{bmatrix} -\mathbf{e}_s^\top R^E_I \\ -\left(\dot{\mathbf{p}}^E_{s/E} - \dot{\mathbf{p}}^E_{b/E}\right)^\top Z R^E_I + \frac{1}{c}\omega_E^\top \left\lfloor \dot{\mathbf{p}}^E_{s/E} \right\rfloor_\times \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \mathbf{q}^b_I} = \kappa_s \begin{bmatrix} \mathbf{e}_s^\top R^E_I R\left(\mathbf{q}^b_I\right)^\top \left\lfloor \mathbf{p}^b_{g/b} \right\rfloor_\times \\ \left(R^E_I R^I_b \left\lfloor \mathbf{v}^I_{b/I} \right\rfloor_\times\right)^\top \mathbf{e}_s + \left(\dot{\mathbf{p}}^E_{s/E} - R^E_I R^I_b \mathbf{v}^I_{b/I}\right)^\top Z R^E_I R\left(\mathbf{q}^b_I\right)^\top \left\lfloor \mathbf{p}^b_{g/b} \right\rfloor_\times \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \mathbf{v}^b_{b/I}} = \kappa_s \begin{bmatrix} 0 \\ -\mathbf{e}_s^\top R^E_I R^I_b + \frac{1}{c}\omega_E^\top \left\lfloor \mathbf{p}^E_{s/E} \right\rfloor_\times R^E_I R^I_b \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \mathbf{p}^E_{I/E}} = \kappa_s \begin{bmatrix} -\mathbf{e}_s^\top I \\ -\left(\dot{\mathbf{p}}^E_{s/E} - \dot{\mathbf{p}}^E_{b/E}\right)^\top Z I \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \mathbf{q}^I_E} = \kappa_s \begin{bmatrix} -\mathbf{e}_s^\top I \\ \mathbf{e}_s^\top R_I^E \left\lfloor \left(\mathbf{p}^I_{b/I} + R^I_b \mathbf{p}^b_{g/b} \right) \right\rfloor_\times \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \boldsymbol{\tau}_b} = \kappa_s \begin{bmatrix} c & 0 \\ 0 & c \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{r}'_{\sigma_s}}{\partial \kappa_s} = \begin{bmatrix} \boldsymbol{\sigma}_s - \bar{\boldsymbol{\sigma}}_s \\ -1 \end{bmatrix}.$$

7.10.2 Projection Residual Jacobian

The projection residual is given by Eq. 7.13 and is repeated here for reference:

$$r_\zeta = \mathbb{P}_{\bar{\zeta}} \left(\bar{\zeta}^j_{\ell/j} - \hat{\zeta}^j_{\ell/j} \right),$$

where

$$\hat{\zeta}^{c_j}_{\ell/c_j} = R^{c_j}_j R^j_I \left[\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{\ell/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right] - \mathbf{p}^j_{c_j/j}.$$

We also use the following identities to help us in the derivation of this Jacobian

$$\frac{\partial}{\partial \mathbf{v}} \|\mathbf{v}\| = \frac{\mathbf{v}^\top}{\|\mathbf{v}\|} \qquad\qquad \frac{\partial}{\partial \mathbf{v}} \frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{I\|\mathbf{v}\| - \mathbf{v}\left(\frac{\mathbf{v}}{\|\mathbf{v}\|}\right)^\top}{\mathbf{v}^\top \mathbf{v}}$$

$$\frac{\partial}{\partial \mathbf{q}} R(q)\,\mathbf{v} = \lfloor R(\mathbf{q})\,\mathbf{v} \rfloor_{\times} \qquad \frac{\partial}{\partial \mathbf{q}} R(q)^{\top}\,\mathbf{v} = -R(q)^{\top} \lfloor \mathbf{v} \rfloor_{\times},$$

and define an intermediate variable

$$Z = \frac{I\left\|\zeta_j\right\| - \zeta_j\left(\frac{\zeta_j}{\|\zeta_j\|}\right)^{\top}}{\zeta_j^{\top}\zeta_j}.$$

We start first with the partial derivative of Eq. 7.13 with respect to the position of the origin node i. This is derived as follows:

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\zeta}}{\partial \mathbf{p}^I_{i/I}} &= \frac{\partial}{\partial \mathbf{p}^I_{i/I}} \mathbb{P}_{\zeta}\left(\frac{\hat{\zeta}}{\|\hat{\zeta}\|} - \bar{\zeta}^j_{l/j}\right) \\
&= \mathbb{P}_{\zeta_i}\left(Z\frac{\partial}{\partial \mathbf{p}^I_{i/I}}\left(R^{c_j}_j R^j_I\left(\mathbf{p}^I_{i/I} + R^I_i\left(R^i_{c_i}\frac{1}{\rho}\zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i}\right) - \left(R^I_j \mathbf{p}^j_{c_j/j} + \mathbf{p}^I_{j/I}\right)\right)\right)\right) \\
&= \mathbb{P}_{\zeta_i} Z R^{c_j}_j R^j_I.
\end{aligned}$$

Next, let us find the partial derivative of Eq. 7.13 with respect to the orientation of the origin node. This is derived as follows:

$$\begin{aligned}
\frac{\partial \mathbf{r}_{\zeta}}{\partial \mathbf{q}^i_I} &= \frac{\partial}{\partial \mathbf{q}^i_I} \mathbb{P}_{\zeta}\left(\frac{\hat{\zeta}}{\|\hat{\zeta}\|} - \bar{\zeta}^j_{l/j}\right) \\
&= \mathbb{P}_{\zeta}\left(Z\frac{\partial}{\partial \mathbf{q}^i_I}\left(R^{c_j}_j R^j_I\left(\mathbf{p}^I_{i/I} + R^I_i\left(R^i_{c_i}\frac{1}{\rho}\zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i}\right) - \left(R^I_j \mathbf{p}^j_{c_j/j} + \mathbf{p}^I_{j/I}\right)\right)\right)\right) \\
&= \mathbb{P}_{\zeta} Z\frac{\partial}{\partial \mathbf{q}^I_{i/I}}\left(R^{c_j}_j R^j_I\left(\mathbf{p}^I_{i/I} + R^I_i\left(R^i_{c_i}\frac{1}{\rho}\zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i}\right) - \left(R^I_j \mathbf{p}^j_{c_j/j} + \mathbf{p}^I_{j/I}\right)\right)\right) \\
&= \mathbb{P}_{\zeta} Z R^{c_j}_j R^j_I \frac{\partial}{\partial \mathbf{q}^i_I} R\left(\mathbf{q}^i_I\right)^{\top}\left(R^i_{c_i}\frac{1}{\rho}\zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i}\right) \\
&= -\mathbb{P}_{\zeta} Z R^{c_j}_j R^j_I R\left(\mathbf{q}^i_I\right)^{\top}\left\lfloor R^i_{c_i}\frac{1}{\rho}\zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i}\right\rfloor_{\times}.
\end{aligned}$$

Similary, let us find the partial derivative of Eq. 7.13 with respect to the position of the destination node j, which is given as

$$\begin{aligned}
\frac{\partial \mathbf{r}_\zeta}{\partial \mathbf{p}^I_{j/I}} &= \frac{\partial}{\partial \mathbf{p}^I_{j/I}} \mathbb{P}_\zeta \left(\frac{\hat{\zeta}}{\|\hat{\zeta}\|} - \bar{\zeta}^j_{l/j} \right) \\
&= \mathbb{P}_\zeta Z \frac{\partial}{\partial \mathbf{p}^I_{j/I}} \left(R^{c_j}_j R^j_I \left(\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i} \right) - \left(R^I_j \mathbf{p}^j_{c_j/j} + \mathbf{p}^I_{j/I} \right) \right) \right) \\
&= \mathbb{P}_\zeta Z \frac{\partial}{\partial \mathbf{p}^I_{j/I}} \left(R^{c_j}_j R^j_I \left(-\mathbf{p}^I_{j/I} \right) \right) \\
&= -\mathbb{P}_\zeta Z R^{c_j}_j R^j_I,
\end{aligned}$$

and the orientation of node j, which is given as

$$\begin{aligned}
\frac{\partial \mathbf{r}_\zeta}{\partial \mathbf{q}^j_I} &= \frac{\partial}{\partial \mathbf{q}^j_I} \mathbb{P}_\zeta \left(\frac{\hat{\zeta}}{\|\hat{\zeta}\|} - \bar{\zeta}^j_{l/j} \right) \\
&= \mathbb{P}_\zeta Z \frac{\partial}{\partial \mathbf{q}^j_I} \left(R^{c_j}_j R^j_I \left(\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right) - \mathbf{p}^j_{c_j/j} \right) \\
&= \mathbb{P}_\zeta Z R^{c_j}_j \left(\frac{\partial}{\partial \mathbf{q}^j_I} R \left(\mathbf{q}^j_I \right) \left(\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right) - \mathbf{p}^j_{c_j/j} \right) \\
&= \mathbb{P}_\zeta Z R^{c_j}_j \left\lfloor R \left(\mathbf{q}^j_I \right) \left(\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right) \right\rfloor_\times .
\end{aligned}$$

Finally, we give the derivation of the partial derivative of Eq. 7.13 with respect to the inverse depth to the feature in the i node frame. This is given as

$$\begin{aligned}
\frac{\partial \mathbf{r}_\zeta}{\partial \rho} &= \frac{\partial}{\partial \rho} \mathbb{P}_\zeta \left(\frac{\hat{\zeta}}{\|\hat{\zeta}\|} - \bar{\zeta}^j_{l/j} \right) \\
&= \mathbb{P}_\zeta Z \frac{\partial}{\partial \rho} \left(R^{c_j}_j R^j_I \left(\mathbf{p}^I_{i/I} + R^I_i \left(R^i_{c_i} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} + \mathbf{p}^i_{c_i/i} \right) - \mathbf{p}^I_{j/I} \right) - \mathbf{p}^j_{c_j/j} \right) \\
&= \mathbb{P}_\zeta Z R^{c_j}_j R^j_I R^I_i R^i_{c_i} \frac{\partial}{\partial \rho} \frac{1}{\rho} \zeta^{c_i}_{l/c_i} \\
&= \mathbb{P}_\zeta Z R^{c_j}_j R^j_I R^I_i R^i_{c_i} \frac{1}{\rho^2} \zeta^{c_i}_{l/c_i}.
\end{aligned}$$

CHAPTER 8: DIRECT RELATIVE EDGE OPTIMIZATION, A ROBUST ALTERNATIVE FOR POSE GRAPH OPTIMIZATION[1]

8.1 Introduction

A pose graph is a data structure that encodes relative transform constraints between arbitrary poses of one or multiple agents. The nodes of a pose graph represent the pose of each node and the edges represent the relative transformations between nodes, both sequentially and non-sequentially [138]. When using nonlinear optimization techniques to determine the maximum-likelihood configuration of the pose graph, constraints can render the problem as over-constrained.

In the robotics community, pose graphs are commonly used to solve the full simultaneous localization and mapping (SLAM) problem. The construction of pose graphs to solve the SLAM problem is well researched and has been demonstrated extensively. Some notable examples include [138–141]. In these applications, pose graph optimization is used to obtain a maximum likelihood estimate for all the poses in the map. Different techniques have been proposed to improve the performance and robustness of pose graph optimization including graph reduction [122, 142], relaxation [143] incremental-pose parameterization [121, 144, 145] and relative parameterization [146–148].

One drawback of global-pose optimization is a strong dependence on the quality of the initialization point for the graph, and there are several instances when large initialization errors may arise. For example, small heading errors in a trajectory compounded over time can cause large errors in the initial pose estimates [145, 149]. This phenomenon is illustrated in Figure 8.1. Large

[1]This paper was written by James S. Jackson, Kevin Brink, Brendon Forsgren, David O. Wheeler, and Timothy W. McLain, and published in Robotics and Automation Letters in 2019 and was presented at at the International Conference of Robotics and Automation in 2019 [137]

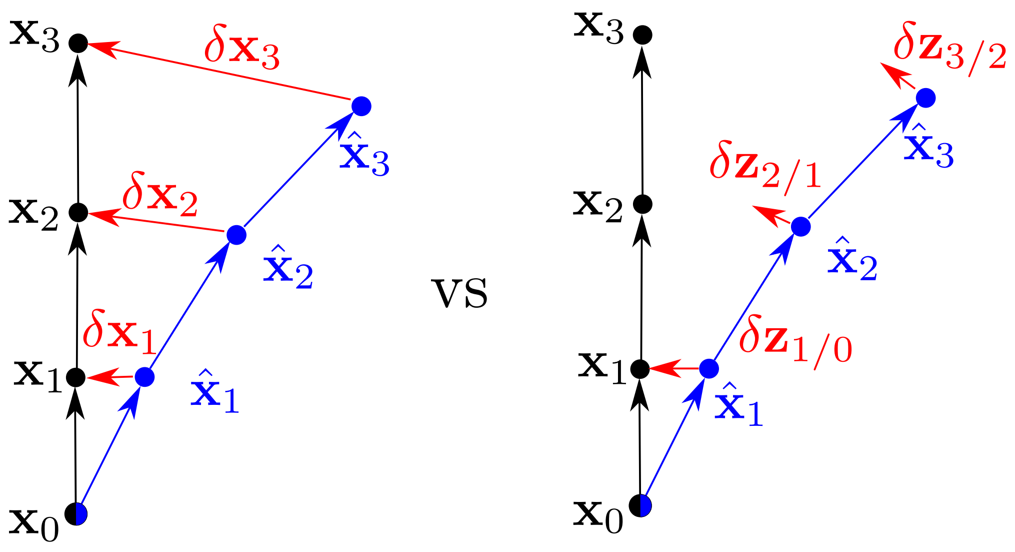


Figure 8.1: Comparison of typical errors in global poses (left) vs individual edge constraints (right) given a trajectory with heading error.

initialization errors also result when an agent is initialized without global information but later gains global information during operation. In this case all initial pose estimates can be arbitrarily far from their true position. Finally, in multi-agent problems, the initial alignment of each agent's trajectory is unknown and can be arbitrarily far from truth. Kim et al. [150] proposed a method to mitigate this initial error between multiple agents, but requires additional complexity to handle edges between the graphs.

Graph initialization is an active area of research and there are several prominent solutions that have been proposed [150–156]. Of particular note, [155] and [156] have shown the guaranteed optimization of pose graphs even given large initialization errors. These approaches restructure the cost function of a pose graph optimization problem into a globally convex form with constraints on the involved variables and use semi-definite programming techniques to guarantee the optimal solution.

In this paper, we discuss the benefits of optimizing directly over edge constraints, or *relative edge optimization* (REO), as opposed to the more popular global pose optimization (GPO) and we extend previously described relative approaches [146–148] to allow optimization over all edges in pose graph and eliminate the map tears that occur from optimizing over a local subset. We also show that posing the problem in a relative context side-steps the above mentioned initialization problems and keeps the problem well-conditioned even in the worst of these situations due to its more linear representation.

Performing pose graph optimization over all degrees of freedom of the agent and over all measurements can be computationally prohibitive over long distances with high sensor update rates. In cases where an IMU is available, the roll and pitch of an agent become locally observable so to reduce computational requirements sometimes only the locally-unobservable states are estimated using loop closures or global information in an optimization problem over $SE(2) \times \mathbb{R}$ [9, 51, 157]. This optimization also often occurs on a reduced form of marginalized state and measurement information [9, 51, 157] to further reduce computational requirements. While the work we present here is designed to be applied in such a situation, it could also be applied to other SLAM problems, such as those in $SE(3)$.

The following sections first derive relative edge optimization and describe how to perform relative batch optimization over all edges in a cyclic graph. Comparisons between REO and GPO are then performed and discussed with the use of a simple simulation study. Finally, results from a hardware experiment with multirotor agents are shown and discussed.

8.2 Derivation

Pose graph optimization is often formulated as a least-squares optimization problem and this formulation can be derived using a classical least-squares optimization approach [8] or from a Bayesian perspective in a factor graph [120]. If the noise about graph edges is assumed to be Gaussian, both derivations ultimately lead to the following expression for the global cost function of the optimization given in [138]

$$
\begin{aligned}
F\left(\hat{\mathbf{z}}, \bar{\mathbf{z}}\right) =& \bar{\mathbf{x}}_0^\top \Omega_0^{-1} \bar{\mathbf{x}}_0 + \sum_{i \in \mathcal{N}} \left(\hat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\right)^\top \Omega_i^{-1} \left(\hat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\right) \\
&+ \sum_{i} \sum_{j \in S_i} \left(\hat{\mathbf{z}}_{j/i} - \bar{\mathbf{z}}_{j/i}\right)^\top \Omega_{j/i}^{-1} \left(\hat{\mathbf{z}}_{j/i} - \bar{\mathbf{z}}_{j/i}\right), \qquad (8.1)
\end{aligned}
$$

where $\mathbf{z}_{j/i}$ is the edge connecting nodes i and j, calculated as $\mathbf{x}_j \boxminus \mathbf{x}_i$; $[\hat{\cdot}]$, $[\bar{\cdot}]$ are the estimated and measured quantities, respectively; S_i is the set of nodes, x_j, connected to node x_i; $\mathcal{N}$ is the set of all nodes in the graph and Ω is the covariance, or weighting assigned to the respective measurements. Note that if $x_j \in S_i$, then $x_i \notin S_j$, else a $1/2$ weighting on $\mathbf{z}_{j/i}$ would be required to avoid double counting edge constraints.

The problem is then to find the optimal set of poses

$$
\hat{\mathbf{x}}^* = \arg\min_{\hat{\mathbf{x}}} F\left(\bar{\mathbf{x}}, \hat{\mathbf{x}}, \bar{\mathbf{z}}, \hat{\mathbf{z}}\right). \qquad (8.2)
$$

To solve Eq. 8.2 in a global context the pose graph is initialized by first defining an initial global pose estimate for each vertex $\hat{\mathbf{x}}_i$ then determining the estimated edges $\hat{\mathbf{z}}_{j/i}$ and then optimizing using an appropriate method. In general, this is done using some variety of Gauss–Newton or Levenberg–Marquardt optimization. Significant work has been done in reducing the complexity of this problem [121, 122, 138] so that it can be performed in real time under realistic computational constraints.

Optimizing over $\hat{\mathbf{x}}$ in Eq. 8.2 inherently casts the optimization problem into a privileged coordinate frame. While this is often appropriate, when only relative information is available the initial global pose estimates can be arbitrarily far from their true position. If these systems then encounter global inputs, or are operating in a multi-agent environment with unknown intial configuration and encounter constraints between agents, the initial error can cause the Newton method to converge to a local minimum because of linearization error in pose-based Jacobians.

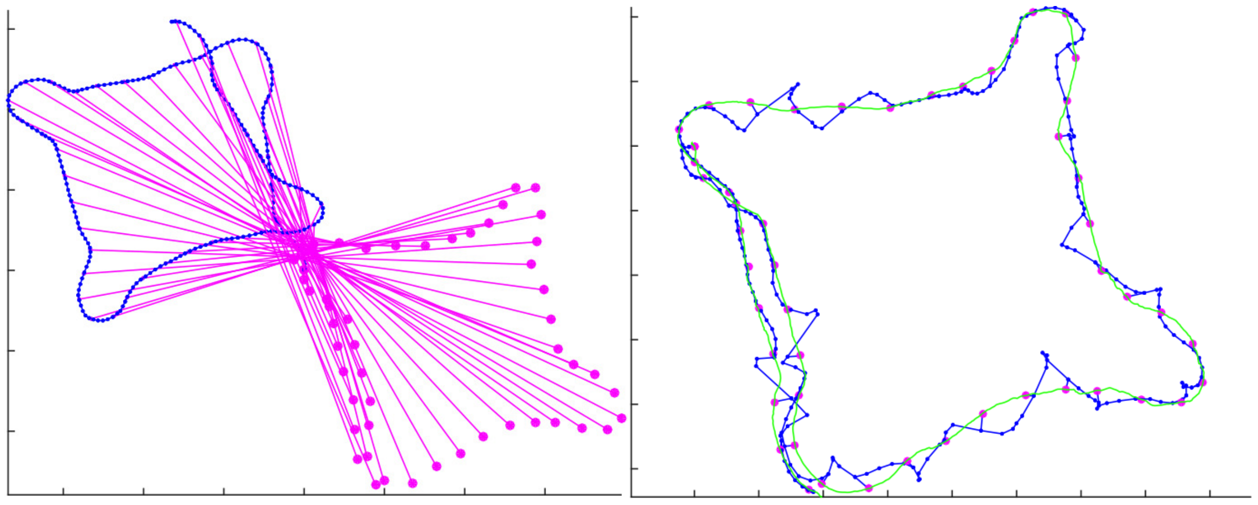


Figure 8.2: Divergent behavior observed in global pose optimization of a multirotor flight around a baseball diamond due to poor initialization. Green lines indicate the original trajectory, magenta dots and lines indicate GPS measurements and their associated poses. The blue lines indicate the initial trajectory before optimization on the left and post-optimization on the right.

Figure 8.2 demonstrates this phenomenon in an exaggerated example. The figure shows results of global-pose optimization on a hardware experiment where a multirotor agent was first given only relative odometry measurements but later receives several delayed GPS inputs. Because no global information was initially available to the system, the agent was initialized with zero heading, although its true heading was approximately 180 degrees. The initial heading error induced large initial errors in global pose and the combination of poor linearization and strong GPS measurement constraints caused the optimization to converge to an incorrect local minima. While this example is exaggerated and there are methods intended to account for this [150–154], it illustrates the poor linearization characteristics of global poses, susceptibility to local minima, and a general lack of robustness when operating with sparse globally defined inputs.

Generally speaking, these situations can be difficult to detect and recover from, and are best avoided all together through either better a priori information, or more robust approaches such as the one suggested in this paper.

8.2.1 Derivation of Relative Edge Optimization

If one assumes the system only has measurements of transforms between nodes (i.e. $\bar{\mathbf{z}}_{j/i} = \mathbf{x}_j \boxminus \mathbf{x}_i$) with covariance $\Omega_{j/i}$, and that edges are only considered once, then Eq. 8.1 reduces to the weighted sum of edge constraints and can be rewritten as

$$\begin{aligned} F\left(\hat{\mathbf{x}}, \bar{\mathbf{z}}\right) &= \sum_i \sum_{j \in S_i} \left(\left(\hat{\mathbf{x}}_j \boxminus \hat{\mathbf{x}}_i\right) - \bar{\mathbf{z}}_{j/i}\right)^\top \\ &\qquad \Omega_{j/i}^{-1} \left(\left(\hat{\mathbf{x}}_j \boxminus \hat{\mathbf{x}}_i\right) - \bar{\mathbf{z}}_{j/i}\right) \\ &= \sum_i \sum_{j \in S_i} \left\| \left(\left(\hat{\mathbf{x}}_j \boxminus \hat{\mathbf{x}}_i\right) - \bar{\mathbf{z}}_{j/i}\right) \right\|_{\Omega_{j/i}^{-1}} . \end{aligned} \tag{8.3}$$

To further generalize the above case, pose-based portions of Eq. 8.1, $\left(\hat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\right)^\top \Omega_i^{-1} \left(\hat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\right)$, can be rewritten as relative constraints [9, 142, 158]. This makes the simplification in Eq. 8.3 quite reasonable. Further, if we reparameterize our state in terms of the estimated relative transform between nodes, $\hat{\mathbf{z}}_{j/i} = \hat{\mathbf{x}}_j \boxminus \hat{\mathbf{x}}_i$, as opposed to estimating the nodes themselves, Eq. 8.3 can be expressed as

$$F\left(\hat{\mathbf{z}}, \bar{\mathbf{z}}\right) = \sum_i \sum_{j \in S_i} \left\| \left(\hat{\mathbf{z}}_{j/i} - \bar{\mathbf{z}}_{j/i}\right) \right\|_{\Omega_{j/i}^{-1}} . \tag{8.4}$$

Theorem 8.2.1. *Given a pose graph that can be expressed purely in terms of full relative constraints (not including partial constraints such as range or bearing), the poses reconstructed from the relative optima arrived at by minimizing Eq. 8.4 starting at the initial node position* $\mathbf{x}_0$ *are identical to the global optima arrived at by minimizing Eq. 8.3.*

Proof. Let us assume that our estimated poses are at the optimal value, $\hat{\mathbf{x}} = \hat{\mathbf{x}}^* = \arg\min_{\hat{\mathbf{x}}} F\left(\hat{\mathbf{x}}, \bar{\mathbf{z}}\right)$ and define a compounding function

$$\hat{\mathbf{x}}_i = g_i^k \left(\hat{\mathbf{z}} \in \mathbf{E}_k, \mathbf{x}_0\right)$$

that compounds some subset of edges $\mathbf{E}_k$ in the appropriate manner from an arbitrary origin (assume $\mathbf{x}_0$ without loss of generality) up to the node $\mathbf{x}_i$. This is illustrated with the green arrow in Figure 8.3 where the position of $\mathbf{x}_4$ is computed using the function g_4^k with $\mathbf{E}_k = \left(\mathbf{z}_{1/0}, \mathbf{z}_{4/1}\right)$. For non-trivial graphs, there could be multiple selections of $\mathbf{E}_k$ that could construct g_i^k, and these paths can be combined to generate the equation

$$\begin{aligned} \hat{\mathbf{x}}_i &= G_i^M\left(\hat{\mathbf{z}}, \mathbf{x}_0\right) \\ &= \frac{1}{m} \sum_{\mathbf{E}_k \in M} g_i^k\left(\hat{\mathbf{z}}, \mathbf{x}_0\right) \end{aligned} \tag{8.5}$$

where M is a set of possible edges that begin at $\mathbf{x}_0$ and terminate at $\mathbf{x}_i$ with cardinality $m = |M|$. At the optimum we have that $g_i^k\left(\hat{\mathbf{z}}^*, \mathbf{x}_0\right) = \hat{\mathbf{x}}_i^* \; \forall \, \mathbf{E}_k$, (i.e. every possible edge sequence results in the optimal $\hat{\mathbf{x}}_i^*$ solution when evaluated at $\hat{\mathbf{z}}^*$). Therefore, $G_i^M\left(\hat{\mathbf{z}}^*, \mathbf{x}_0\right)$ will also have the same optimal solution for $\hat{\mathbf{x}}_i$ regardless of which edge paths are used.

If we re-write Eq. 8.3 using Eq. 8.5 we get an expression for our global optimization function in terms of only the relative constraints and the initial pose

$$F\left(\hat{\mathbf{z}}, \mathbf{x}_0, \bar{\mathbf{z}}\right) = \sum_i \sum_{j \in S_i} \left\| \left(\left(G_j^M\left(\hat{\mathbf{z}}, \mathbf{x}_0\right) \boxminus G_i^M\left(\hat{\mathbf{z}}, \mathbf{x}_0\right) \right) - \bar{\mathbf{z}}_{j/i} \right) \right\|_{\Omega_{j/i}^{-1}} \tag{8.6}$$

If we assume that we have optimized Eq. 8.6 with respect to edges (i.e. $\hat{\mathbf{z}} = \hat{\mathbf{z}}^* = \operatorname{argmin}_{\hat{\mathbf{z}}}\left(F\left(\hat{\mathbf{z}}, \mathbf{x}_0, \bar{\mathbf{z}}\right)\right)$ then

$$\frac{\partial F}{\partial \hat{\mathbf{z}}} = 0,$$

which implies that

$$\frac{\partial F}{\partial \hat{\mathbf{x}}} \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{z}}} = 0,$$

and because $\partial \hat{\mathbf{x}} / \partial \hat{\mathbf{z}} \neq 0$, this implies that

$$\frac{\partial F}{\partial \hat{\mathbf{x}}} = 0.$$

□

The above proof shows that the cost functions Eq. 8.6 and Eq. 8.1 share the same minima, regardless of parameterizing with respect to pose or edges. This fact also allows for leeway in the selection of which paths, $\mathbf{E}_k$, are selected. The only requirement is that all edges in the graph are included, or else one cannot fully reconstruct Eq. 8.6. In practice, the choice of which edges are used could have a significant effect on the resulting Jacobians used in optimization. The choice of edges will therefore affect optimization performance, however the impact of this choice appears to be small compared to the nonlinearities induced by large errors in a pose-based optimization and regardless of loops/paths chosen, the global minima will remain unchanged.

In practice, actually solving Eq. 8.6 for arbitrary edge paths is not straightforward, especially if loops are present. Other relative approaches to loop closure [146, 147], use a sliding window, or an *active set* of edges that do not traverse the full loop. The advantage to these approaches is that the algorithms run in constant time, even at loop closure (because only a limited set of nodes in the loop are considered). However, these approaches result in map tears at the boundary of the active set of nodes that lead to global map inconsistency. If we instead construct M in such a way that $\mathbf{E}_k$ forms non-trivial loops that includes $\mathbf{z}_{j/i}$ for the evaluation of G_i^M and G_j^M, then we can rewrite Eq. 8.6 as a sum over loops rather than individual edge constraints and optimize each loop independently. This is possible because any common edges involved in the pose reconstruction of G_i^M and G_j^M are subtracted out in Eq. 8.6, and only a simple loop remains (see Figure 8.3). Because some edges may be traversed by more than one loop, edges must be weighted within a loop inversely proportional to the total number of loops in the optimization which traverse that edge. This ultimately leads to the final form of the cost function for relative edge optimization given as

$$F\left(\hat{\mathbf{z}}, \mathbf{x}_0, \bar{\mathbf{z}}\right) = \sum_{\mathcal{L}\in M} \sum_{j,i\in\mathcal{L}} \frac{1}{n_{j/i}} \left\|\hat{\mathbf{z}}_{j/i} - \bar{\mathbf{z}}_{j/i}\right\|_{\Omega_{j/i}^{-1}}, \tag{8.7}$$

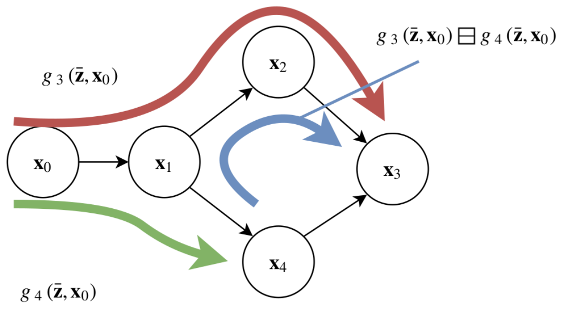

$g_3(\bar{\mathbf{z}}, \mathbf{x}_0)$
$g_3(\bar{\mathbf{z}}, \mathbf{x}_0) \boxminus g_4(\bar{\mathbf{z}}, \mathbf{x}_0)$
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$g_4(\bar{\mathbf{z}}, \mathbf{x}_0)$

Figure 8.3: Subtracting two concatenation functions removes any common edges between the two paths, resulting in a simple loop. The blue path is the result of subtracting the green path from the red path and does not contain the common edge $\mathbf{z}_{1/0}$.

where M is the set of loops that must fully span the set of edges, $\mathcal{L}$ is a single loop and $n_{j/i}$ is the number of times the edge $z_{j/i}$ appears in M. The process of segmenting the graph into simple loops is illustrated in Figure 8.4. The graph is split into two simply-connected loops that share some common edges. The sectioning of the graph is arbitrary, and two potential options are shown in the figure.

Starting at Eq. 8.7, we can re-construct Eq. 8.6 by selecting any subset of loops that contain every $\mathbf{x}_i$ and rewriting edge estimates in terms of the compounding functions G_i^M (i.e. $\hat{\mathbf{z}}_{j/i} = G_j^M \boxminus G_i^M$). If we expand this and regroup on nodes $\mathbf{x}_i$, we arrive back to Eq. 8.6, which we know is equivalent to Eq. 8.3.

The cost function in Eq. 8.7 provides the user with significant flexibility and can make the optimization tractable. If all loops are considered, the optimization can become computationally expensive. However, users can select a subset of loops for edge-based optimization that maintains the global minima, and is also likely to provide good structure, accurate Jacobians, and tractable computational loads to the optimization algorithm. While this this paper does not address selection of desired loops specifically, it will be demonstrated in Sec. 8.3 that even for an arbitrary selection of loops the edge-based optimization applied to Eq. 8.7 shows a dramatic increase in robustness when compared to pose-based optimization of Eq. 8.6. We now discuss the optimization of Eq. 8.7.

8.2.2 Solving REO

We wish to find the optimal update $\Delta\mathbf{z}^*$ to our initial edge estimate $\hat{\mathbf{z}}$. Letting $\hat{\mathbf{z}}^+_{j/i} = h\left(\hat{\mathbf{z}}_{j/i}, \Delta\mathbf{z}\right)$ where h properly considers the way that $\Delta\mathbf{z}$ maps into the edge update, we see

$$\begin{aligned} F\left(\hat{\mathbf{z}}^+, \bar{\mathbf{z}}\right) &= \sum_{\mathcal{L}\in M} \sum_{i,j\in\mathcal{L}} \frac{1}{n_{j/i}} \left\|\hat{\mathbf{z}}^+_{j/i} - \bar{\mathbf{z}}_{j/i}\right\|_{\Omega_{j/i}} \\ &= \sum_{\mathcal{L}\in M} \sum_{i,j\in\mathcal{L}} \frac{1}{n_{j/i}} \left\|h\left(\hat{\mathbf{z}}_{j/i}, \Delta\mathbf{z}\right) - \bar{\mathbf{z}}_{j/i}\right\|_{\Omega^{-1}_{j/i}} . \end{aligned} \tag{8.8}$$

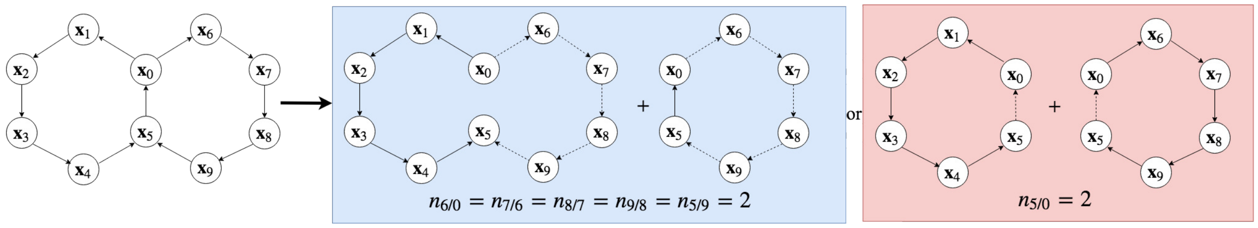
If we take the derivative of Eq. 8.8 with respect to $\Delta\mathbf{z}$ and set it to zero, we can solve for the optimal edge update $\Delta\mathbf{z}^*$.

Defining and evaluating $H = {}^{\partial h}/_{\partial\hat{\mathbf{z}}}$. is non-trivial because there is no straightforward way to find the interactions between edges $h\left(\hat{\mathbf{z}}_{j/i}, \Delta\mathbf{z}\right)$ in the loop. Previously published approaches to REO [146–148] side-step this issue by optimizing a local subset of edges during a loop closure. However, as noted in [146–148], this often results in a globally inconsistent map. In contrast, we address these interdependent constraints by first reparameterizing our cost function into the sum of loops. We approximate $h\left(\hat{\mathbf{z}}_{j/i}, \Delta\mathbf{z}\right)$ by modeling each loop as an independent loop closure with a series of *odometry* edges and a single *loop closure* edge. The odometry edges and their associated perturbations are concatenated normally and all error is lumped into the loop closure edge resulting in the following segmented definition for h:

$$\begin{aligned} h^{\text{odom}}\left(\mathbf{z}_{j/i}, \Delta\mathbf{z}\right) &= \mathbf{z}_{j/i} + \Delta\mathbf{z}_{j/i} \\ h^{\text{LC}}\left(\mathbf{z}_{j/i}, \Delta\mathbf{z}\right) &= \left(\mathbf{z}_{b/i} + \Delta\mathbf{z}_{b/i}\right) \boxplus \left(\mathbf{z}_{c/b} + \Delta\mathbf{z}_{c/b}\right) \\ &\quad \cdots \boxplus \left(\mathbf{z}_{j/z} + \Delta\mathbf{z}_{j/z}\right) . \end{aligned}$$

Finding an approximation of $H_{j/i}$ is now possible, even in non-trivial spaces for all edges in the graph, enabling a globally consistent solution. For example, in $SE(2)$, $H^{\text{odom}}_{j/i} = I$, and an example algorithm for calculating $H^{\mathrm{LC}}_{j/i}$ is given in Algorithm 2. This algorithm could be modified to accommodate other systems such as $SE(3)$

The distinction between loop closure and odometry edges in a simple loop is somewhat arbitrary. In our implementation we have chosen the *loop closure* edge as the most recently acquired measurement in the loop. This is an area of future study because, as mentioned before, the choice of this edge likely influences optimization performance.



$\mathbf{x}_0$ $\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{x}_5$ $\mathbf{x}_6$ $\mathbf{x}_7$ $\mathbf{x}_8$ $\mathbf{x}_9$
+
or
$n_{6/0} = n_{7/6} = n_{8/7} = n_{9/8} = n_{5/9} = 2$
$n_{5/0} = 2$

Figure 8.4: A pose graph, composed of two simply-connected loops that share a single edge. We model the graph as the sum of a subset of simply-connected loops. Two ways to segment this graph are shown here for reference, but there is a third perfectly valid configuration which is not shown.

Algorithm 2 Calculation of $H^{\ell}_{a/z}$ for a pose graph cycle with reversed edges in $SE(2)$

```
 1: for $\mathbf{z}_{j/i} \in \ell_{\mathrm{odom}}$ do
 2:     if $\mathrm{dir}(\mathbf{z}_{j/i}) > 0$ then
 3:         $\frac{\partial \Delta t_{a/z}}{\partial \Delta t_{j/i}} = \prod_{m,n<i,j} R^n_m$
 4:     else
 5:         $\frac{\partial \Delta t_{a/z}}{\partial \Delta t_{j/i}} = -\prod_{m,n\le i,j} R^n_m$
 6: for $\mathbf{z}_{j/i} \in \ell_{\mathrm{odom}}$ do
 7:     if $\mathrm{dir}(\mathbf{z}_{j/i}) > 0$ then
 8:         $\frac{\partial \Delta t_{a/z}}{\partial \theta_{j/i}} = \sum_{m,n>i,j} \frac{\partial \Delta t_{a/z}}{\partial \Delta t_{n/m}} \Delta t_{n/m}$
 9:     else
10:         $\frac{\partial \Delta t_{a/z}}{\partial \theta_{j/i}} = \sum_{m,n\ge i,j} -\frac{\partial \Delta t_{a/z}}{\partial \Delta t_{n/m}} \Delta t_{n/m}$
11:     $\frac{\partial \theta_{a/z}}{\partial \theta_{j/i}} = \mathrm{dir}(\mathbf{z}_{j/i})$
12:     $\frac{\partial \Delta t_{a/z}}{\partial \theta_{j/i}} = \mathbf{0}$
```

8.3 Results Comparing Pose and Edge-based Optimization

Several simulation studies and a hardware experiment were performed to illustrate some of the similarities and differences of global pose and edge-based optimization. The first study illustrates the equivalence between REO and GPO in well-conditioned problems and the second study demonstrates the improved robustness of REO compared to GPO in the presence of noisy edge measurements. The third study illustrates the robustness of REO to gross initial heading error, where GPO often fails completely. Finally, the hardware experiment demonstrates that the algorithm works on real-world data collected by an autonomous system and is well-suited for multi-agent scenarios.

Each simulation study focused on the optimization of the pose graph of a house-shaped trajectory in $SE(2)$ shown in Figure 8.5 that contains nine nodes and five loop closures. This trajectory was chosen because it is simple enough that properly converged graphs are easy to identify, but also complex enough to illustrate non-trivial items of discussion. The Levenberg-Marquardt optimizer implemented in GTSAM [85] was used as the GPO algorithm, while a prototype implementation of Algorithm 2 was used for REO. To evaluate performance of the optimization routines, we considered the RMS error of the final optimized position of each node in the graph

$$J = \frac{1}{N}\sum_{i=0}^{N} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|,$$

where the heading state of each node was not considered in the RMS error metric to avoid scaling ambiguity between position and heading error. The pose of the solution optimized by REO was calculated by starting at the same origin pose estimated by GPO and compounding the optimized edge estimates to put both solutions into the same frame of reference.

8.3.1 Simulation Example 1: Well-Conditioned

The first study illustrates that for a typical, well-conditioned trajectory, the optimizations perform equivalently. The house trajectory was corrupted with small amounts of random Gaussian noise on both translation and heading (see Table 8.1) and a loop closure was placed between nodes at each corner of the square in the house. One thousand of these trajectories were solved by both optimization routines and an example is shown in Figure 8.5. Every one of these optimizations produced virtually identical results between global pose and edge-based optimization. A histogram of RMS error of the solution found by both optimization algorithms is also shown in Figure 8.5. Because of the noisy inputs, neither approach perfectly resolves the initial trajectory.

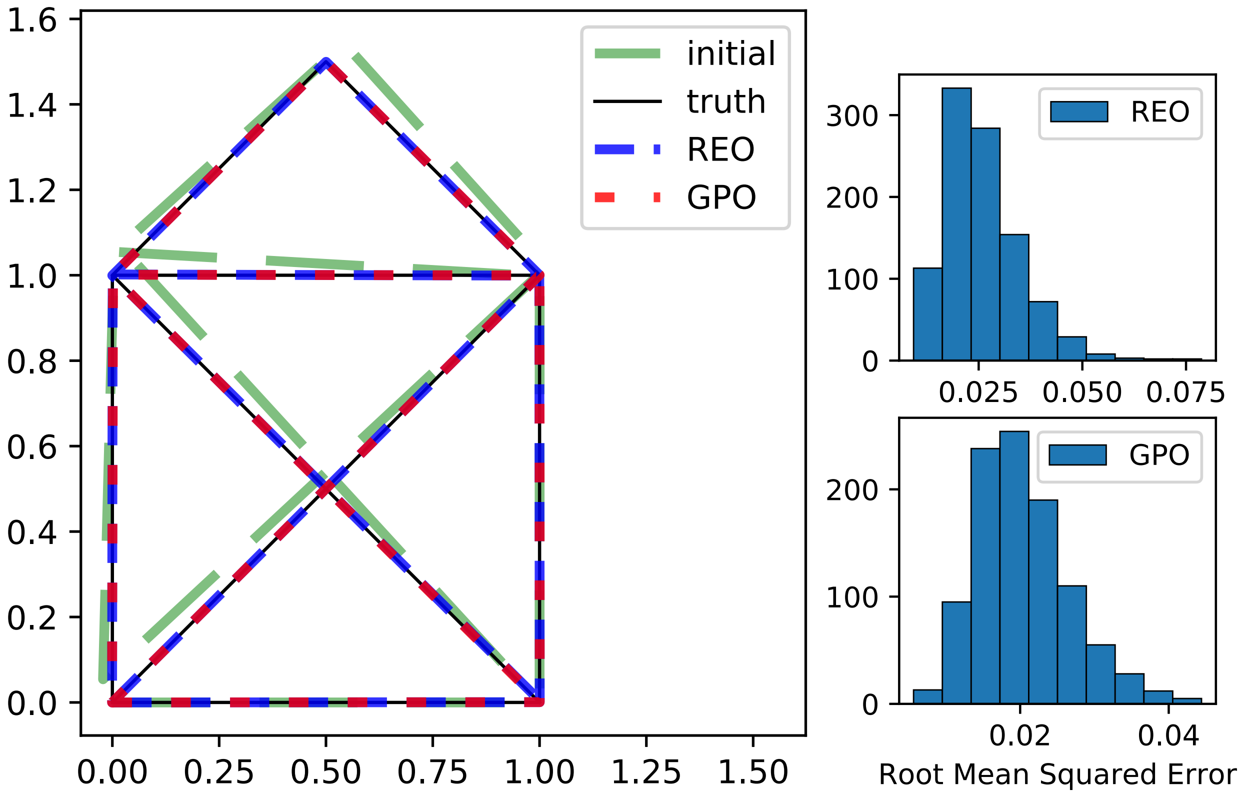


Figure 8.5: One sample and summary results of simulation example 1 where trajectories were corrupted with small errors in both translation and heading. All trials had nearly identical optimization results between relative-edge and global-pose optimization. The histogram shows the spread of the RMS error for both REO and GPO over 1000 trials.

Table 8.1: Table of noise parameters used in the simulation examples.

Sim #	σ_x^2 (m^2)	σ_y^2 (m^2)	σ_ψ^2 (rad^2)
1	1×10^{-5}	1×10^{-5}	1×10^{-5}
2	1×10^{-3}	1×10^{-3}	1×10^{-1}
3	1×10^{-5}	1×10^{-5}	1×10^{-2}

8.3.2 Simulation Example 2: Large Noise

As noted earlier, a problematic situation for global pose optimization occurs when significant heading error compounds over several edges to produce poor initial pose estimates. To model this situation, the second simulation study was to optimize random house trajectories that were generated with significant noise on translation and heading (see Table 8.1). Figure 8.6 shows an example of one of these trajectories, and a histogram of the RMS error of the two approaches.

In this situation, while GPO was able to find the correct solution much of the time (as shown by the large bin close to zero in the histogram in Figure 8.6), it sometimes failed to find the solution at all and resulted in divergent behavior. REO, however, found the appropriate minima every time. Although the fit quality is reduced compared to the low noise case, as expected, REO displays significantly more robustness than GPO in this case.

8.3.3 Simulation Example 3: Global Heading Misalignment

Many global pose optimization methods struggle when the initial heading is inaccurate. This sort of situation often occurs in GPS-denied navigation or multi-agent problems when, for lack of better information, an agent is initialized in a nominal direction that is out of alignment with its true global heading. To illustrate this, a simulation study was performed where the house trajectory was initialized with 90 deg of error in global orientation. Small amounts of translation noise and heading noise were applied to each edge (see Table 8.1) and loop closures between the corners of the house were replaced with loop closures to a *virtual node* co-located at $\mathbf{x}_0$ and connected to $\mathbf{x}_0$ with a zero-information edge as described in [9, 142, 158]. Figure 8.7 shows a sample trajectory

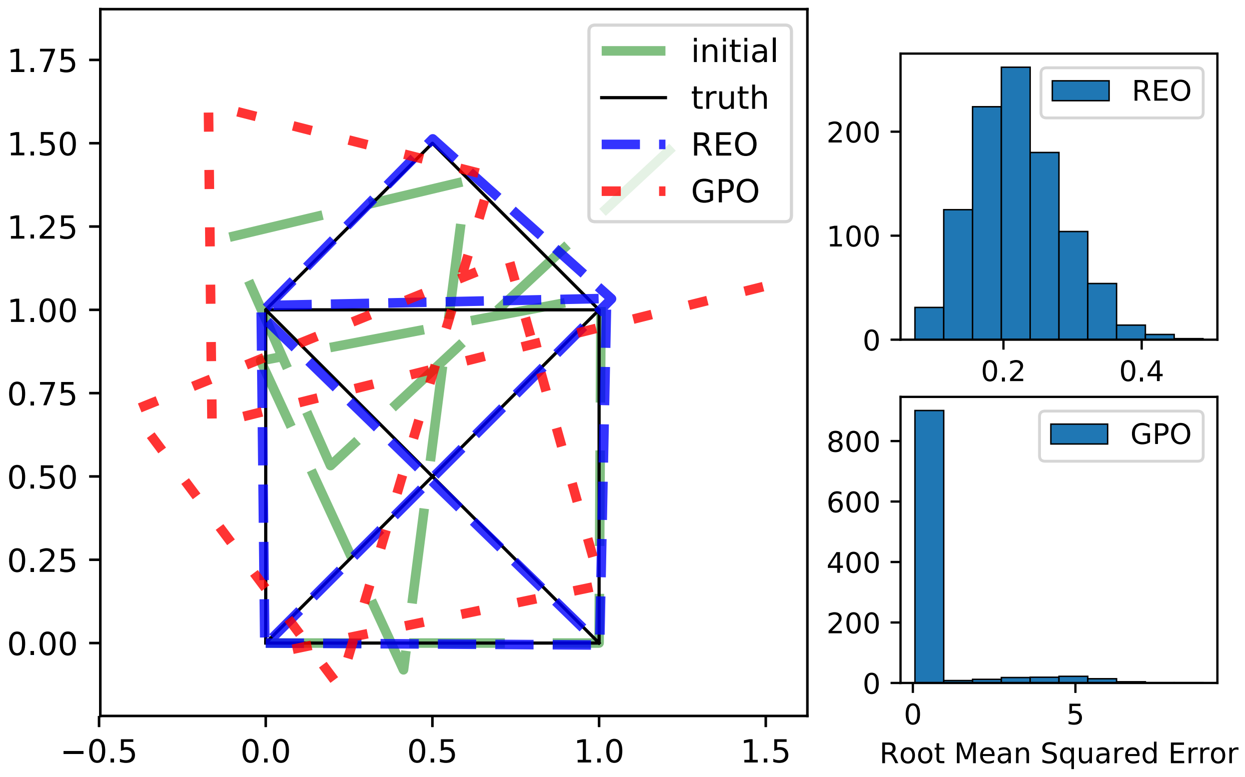


Figure 8.6: One sample and summary results of simulation example 2 where 1000 trajectories were corrupted with large errors in both translation and heading. GPO was often able to find the solution, however, it diverged in approximately 10% of cases (as shown in the RMSE histograms). REO found the optimal solution each time.

and summary results from this study. The results of this study illustrate how GPO struggles in this situation and may end up stalling in a local, and incorrect, minima.

The fundamental problem illustrated in Sec. 8.3.2 and Sec. 8.3.3 is the compounding of errors and uncertainty that occurs in the pose parameterization. As a result of these large errors and uncertainty, pose-based Jacobians must be evaluated far from the solution and incur significant linearization error (See Figure 8.1). In contrast, the error and uncertainty about any relative constraint is independent of any other edges. This means that a relative parameterization is much more linear in nature, and easily deals with problems such as arbitrarily large heading misalignment.

This idea is illustrated by a loop containing four edges with heading error in Figure 8.8. Note the size of $\Delta\mathbf{x}_1$ and associated uncertainty in the global update (upper right) compared the incremental

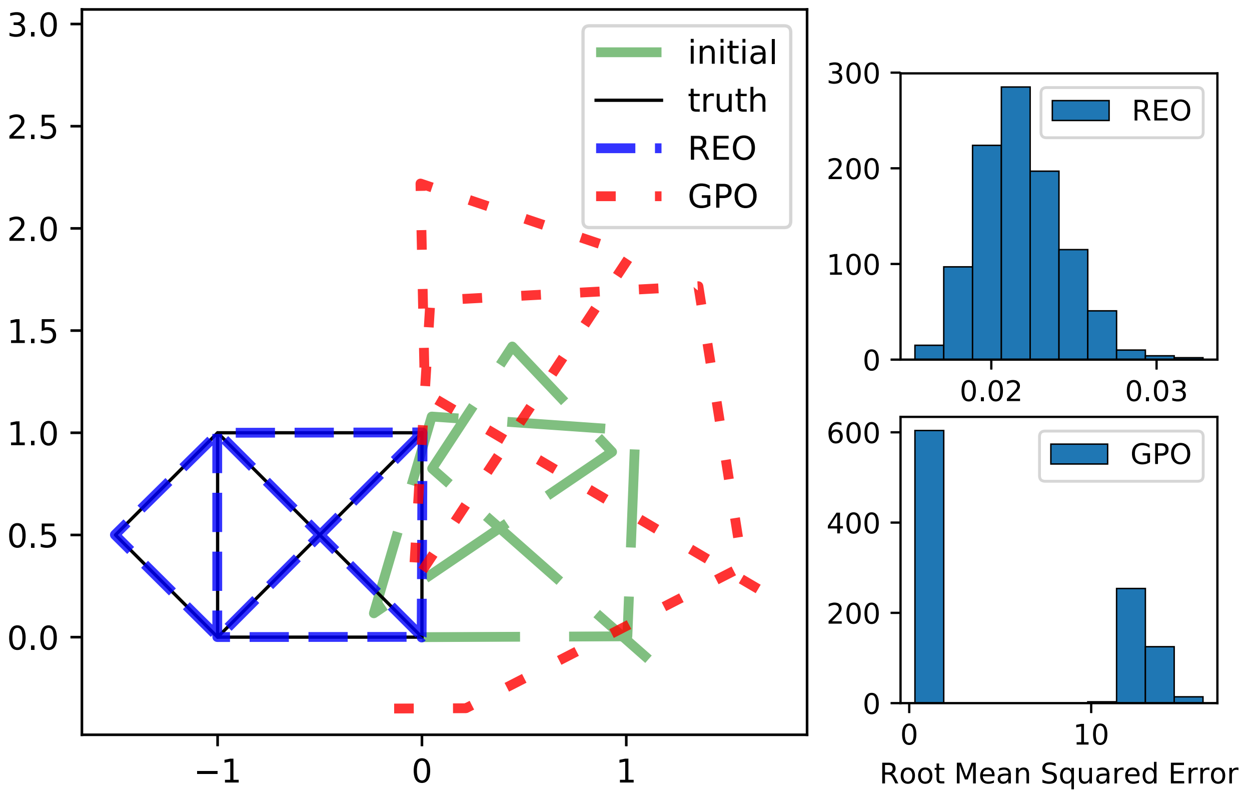


Figure 8.7: One sample and summary results of simulation example 3: the house trajectory simulation study with 90° initial heading error. GPO was often able to find the solution, however in approximately 40% of cases, it diverged (as shown in the RMSE histogram). REO found the optimal solution for each of the 1000 trajectories.

edge updates $\Delta\mathbf{z}_{j/i}$ (bottom). Because the uncertainty estimates for edges do not compound like they do for poses, both the error and uncertainty for a relative edge constraint are typically lower than its connected poses, and therefore, linearization errors are smaller. For this reason we believe that the recent globally guaranteed methods in global pose optimization [155, 156] could be applied to our relative parameterization of the cost function. Such an approach would combine the benefits of guaranteed global convergence with the smaller initial error in the relative parameterization and potentially improve the speed of convergence.

8.3.4 Hardware Experiment

To demonstrate REO on a non-trivial data set, the algorithm was used to optimize data collected by an experimental autonomous navigation system flying two loops through an indoor/outdoor

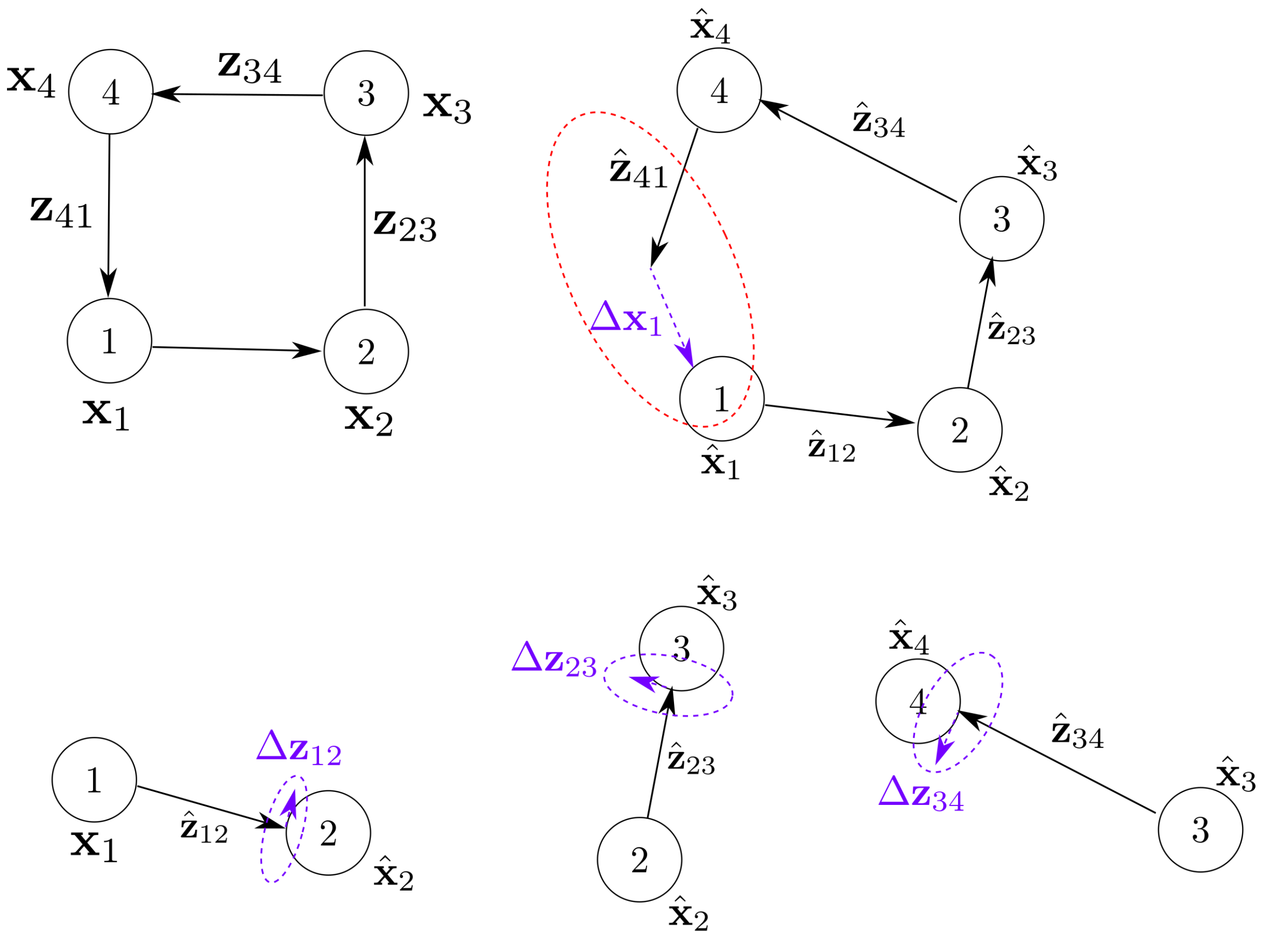


Figure 8.8: A simple loop of four nodes and the associated updates to pose (upper right) and edges (bottom).

environment with an RGBD camera using a multirotor aircraft, as described in [51] and shown in Figure 8.9. The data includes 891 nodes and 30 loop closures calculated using the methods described in [51] and is shown in Figure 8.10. Each loop is approximately 100 meters around and the noise parameters for the experiment are given in Table 8.2.

Table 8.2: Table of noise parameters in the hardware experiment.

Edge Type	$\sigma_x^2(\mathrm{m}^2)$	$\sigma_y^2(\mathrm{m}^2)$	$\sigma_\psi^2(\mathrm{rad}^2)$
Odometry	8.2×10^{-4}	8.2×10^{-4}	2.2×10^{-7}
Loop Closure	1×10^{-5}	1×10^{-5}	1×10^{-3}

Both algorithms produced similar results to the ones found in [51]. Because the results were not identical they were both evaluated using the original cost function without global information, given in Eq. 8.4. The the value of the cost function for REO was 0.00303 and for GPO was 0.115.

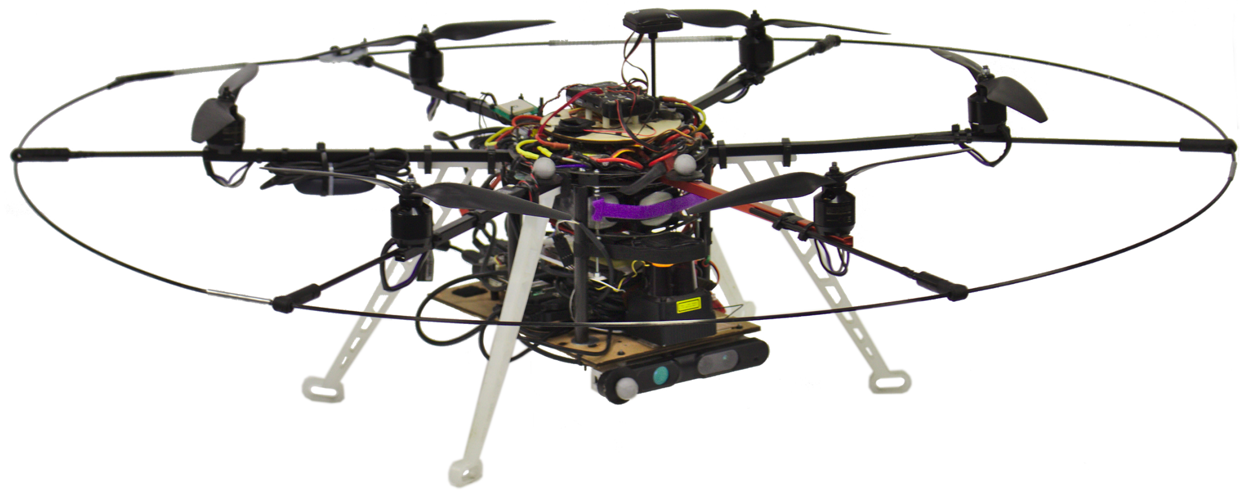


Figure 8.9: The hexacopter used to collect the data in the hardware experiment

The specific reason for the difference between the result given by two algorithms is unclear but may be due to GPO reaching a local minima close to the global solution.

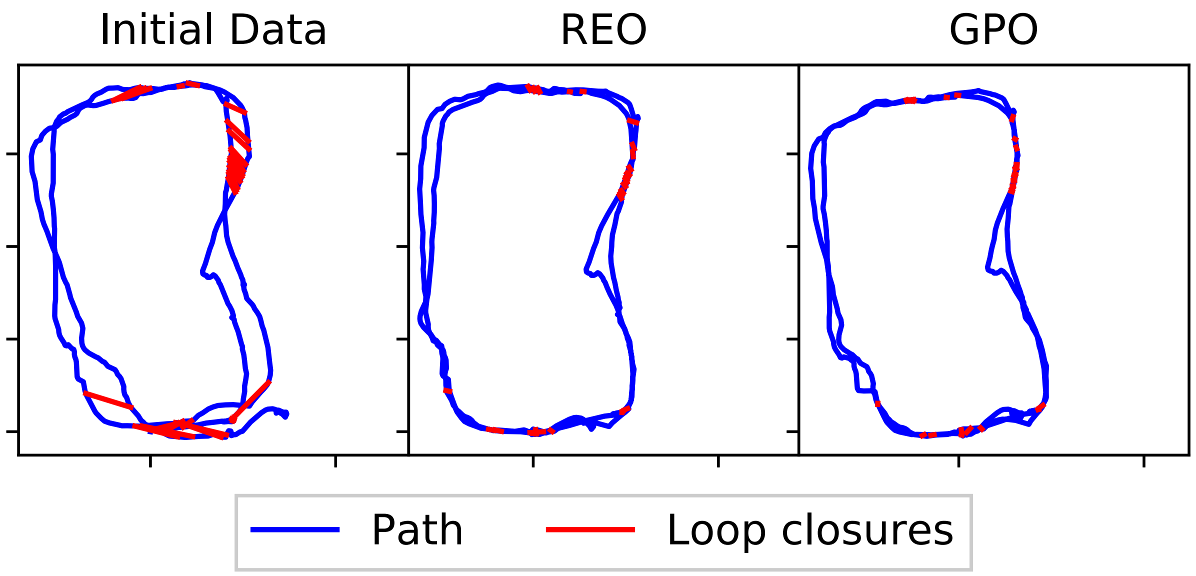


Figure 8.10: The results of the optimization when one agent is collecting data.

In a second experiment, to illustrate a collaborative mapping problem, the same data set was used but the data was split in two as if it had been collected by independent agents. The initial position of each agent was assumed to be unknown and therefore one agent was initialized with a heading error of approximately 180 deg. The associated paths and loop closures for each agent can be seen in Figure 8.11.

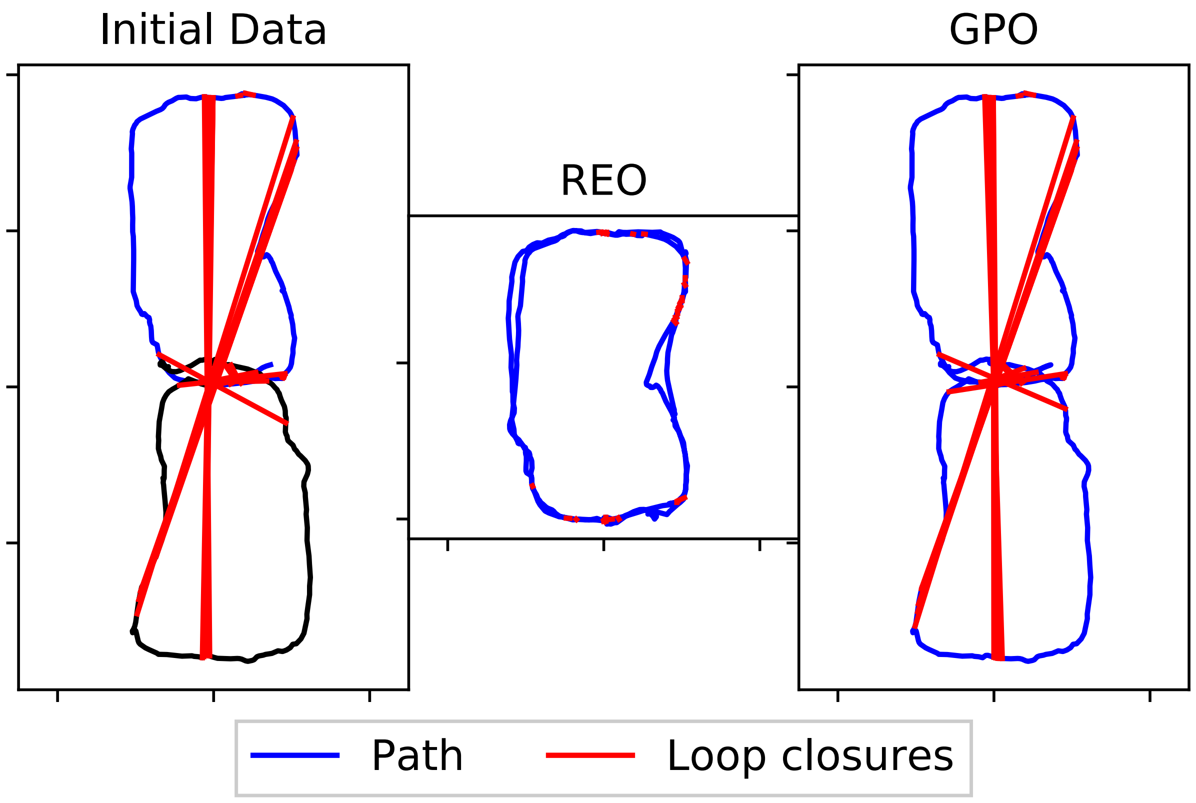


Figure 8.11: The results of the optimization when multiple agents are collecting data. In the plot of the initial data the blue and black lines are the data collected by the first and second agent respectively.

In the multi-agent case, GPO failed to converge to any solution other than the initial guess. Additional testing revealed that GPO only converged if the initial heading error was less than 100 deg. This indicates that the jacobians of the cost function were so ill-conditioned that the cost function appeared locally flat and the stopping condition was satisfied in GPO after the first iteration. REO, on the other hand, converged for the given initial heading error between agents and closely matches the results in Figure 8.10 and [51].

In REO the majority of error in the graph at initialization exists only in the loop closure edges and there is little incentive for the optimization to modify odometry edges until the maps are first aligned. GPO does not have this property, so previous approaches deliberately inform the optimization of the alignment error through the addition of extra anchor nodes and modification of the loop closure edges [150]. We require no such modification, and all constraints can be considered homogeneously just by nature of the parameterization.

In a real-time application, it is likely that an optimized implementation of REO would perform slower than the GPO implementation in GTSAM because of the dense Jacobian REO produces. However, given the demonstrated robustness of REO over traditional methods, optimization of the REO algorithm is of interest and we believe that REO can be refined to run in real time on problems of practical importance.

8.4 Conclusion

In summary, we have shown that optimizing with respect to relative-edge constraints is robust to large initial and propagated heading errors. We have extended the relative optimization technique presented in [146] to avoid map tears that occur when optimizing only a subset of edges. With these improvements, we conclude that REO should be considered for use in place of, or in tandem with GPO, for solving graph optimization problems that exhibit large initial and propagated heading errors that have proven problematic to global approaches.

CHAPTER 9: CONCLUSIONS AND FUTURE WORK

Before we can capture the exciting opportunities provided by autonomous MAVs, we must first improve robustness to autonomous operations in GNSS-degraded areas. This dissertation treaded a broad range of problems, starting with the lowest levels of flight control and working up to the multi-agent mapping problem, all with a focus on improving robustness in these areas. Chapter 3 discussed improving the low-level flight control architecture to enable the flexible experimentation and rapid prototyping that enabled the rest of the work and has been subsequently used in many projects both at BYU and in other universities around the world.

Chapter 4 presented a method to improve the execution of commanded trajectories by performing optimal control directly on the rotation manifold. This work showed that performing optimization directly on the manifold results in a significant improvement in the computational efficiency of optimal control for a MAV, exceeding the state-of-the-art. This work was built upon in [13] in which hardware demonstrations of this method showed improved computational efficiencies when compared with state-of-the-art, and validated the theoretical contributions of Chapter 4. Chapter 5 described a novel method to rapidly adapt to unknown and changing environments. This method was demonstrated in several hardware experiments to validate the practical usefulness of this method.

Given fast, accurate and save control command execution provided by the new flight control architecture, optimal control strategies and obstacle avoidance, the next main area of work presented targeted the problem of robust state estimation in GPS denied and GPS-degraded environments. Chapter 6 showed how improving the dynamic model, adding partial updates and a keyframe step to monocular visual-inertial filtering significantly improved filter accuracy and consistency. These improvements are fairly simple to implement, and incur almost no computational cost, but yield significant benefits.

Unfortunately, a visual-inertial system is still unable to avoid drift, as visual information is inherently relative. GNSS is a clear candidate to provide global information, but including GNSS information in a GNSS-degraded area is not trivial. Chapter 7 presents GV-INS as an elegant way to incorporate this information in a MHE framework. The use of MHE not only improves the visual-inertial component of the approach, but it allows for flexible measurement models, such as the switching parameter that was shown to effectively reject multipath. Chapter 7 also showed that the proper fusion of visual, inertial and GNSS measurements results in a synergistic relationship between the different sensors. Visual information improves multipath rejection, and properly fused GNSS measurements improves depth and velocity estimation that are sometimes difficult in a purely visual-inertial system.

Finally, some potentially rewarding applications of MAVs includes collaborative operation of multiple agents. The other work in this dissertation provides a foundation for robust operation of a single agent, but fusing information from multiple agents can be difficult, and is not always possible with traditional methods. Relative edge optimization (REO) is a potential alternative to the traditional global pose optimization (GPO) framework which is able to overcome many of GPO's limitations. REO has been shown to outperform GPO in situations with significant error in relative pose estimates, or initial heading error. These conditions are very common in multi-agent situations and REO is a strong candidate to solving these kinds of problems.

9.1 Recommendations for Future Work

Limitations of Monocular SLAM approaches This dissertation focused on improving the robustness of only monocular SLAM. It is probable that a stereo or multi-camera setup could provide additional robustness. Existing approaches to stereo-inertial odometry suffer from issues when data mis-association cause significant estimation errors. The use of MHE with a switching parameter or direct pixel intensity methods may improve robustness to these issues.

Carrier-Phase Based Relative State Estimation When performing GNSS fusion in a MHE context, we discovered an interesting analog between the GNSS carrier-phase measurement and the feature projection measurement model. The carrier-phase measurement is cleverly used in real-time-kinematic (RTK) methods by differencing multiple measurements between two receivers. It is not obvious how to incorporate this measurement from a single receiver in a filtering state estimation approach, however, in a MHE scheme, it appears that we can difference measurements to the same satellite using the same receiver at different time instances to achieve many of the same advantages enjoyed by a two-receiver RTK approach. Granted, we will only be able to observe relative information between subsequent states, but this information could be just as useful as visual information in improving real-time performance. Pursuing this avenue of research is an interesting opportunity that could potentially yield significant improvements in accuracy and robustness.

Limitations of REO While REO is significantly more robust to significant errors and initial alignment problems, it is computationally inefficient, and does not scale well with increasing numbers of loop closures. Making REO computationally scalable would be a significant effort, but could yield impressive robustness to detecting false loop closures, poor graph initialization and other real-world conditions which affect current pose-based solutions.

Multi-Agent Experimentation and Validation of REO All experiments in this work were performed with a single agent, although many of the initialization issues addressed by Chapter 8 primarily occur in multi-agent scenarios. All multi-agent experimental validation in Chapter 8 was done by splitting the trajectory of a single agent into two. Future work could include demonstrating

REO with several agents, and solving the related communications and consensus issues faced in collaborative situations.

Autonomous Missions While this work did focus on a number of issues related to performing real-world autonomous operations, it did not focus on the high-level path planning and situational awareness required for useful tasks. Closing the loop on the improved state estimation techniques presented in this dissertation with robust path-planning in real-life operations would be an interesting future direction and might yield additional problems that need to be solved.

REFERENCES

[1] "Top countries/markets by smartphone penetration & users," 2019. [Online]. Available: https://newzoo.com/insights/rankings/top-50-countries-by-smartphone-penetration-and-users/

[2] PricewaterhouseCoopers, "Clarity from above: PwC global report on commercial applications of drone technology," https://www.pwc.pl/pl/pdf/clarity-from-above-pwc.pdf, May 2016, accessed: 2017-03-30.

[3] N. Schwartz, "The United States as an aerospace nation: Challenges and opportunities," in *IFPA Fletcher Conference on National Security Strategy and Policy*, Medford, MA, Jan. 2010.

[4] P. W. Nyholm, "Globally consistent map generation in GPS-degraded environments," Master's thesis, Department of Mechanical Engineering, Brigham Young University, Provo, Utah, 2015.

[5] D. Wheeler, D. Koch, J. Jackson, T. McLain, and R. Beard, "Relative navigation: A keyframe-based approach for observable GPS-degraded navigation," *IEEE Control Systems Magazine*, vol. 38, no. 4, pp. 30–48, Aug 2018.

[6] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2014.

[7] J. Solà, "Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 1, pp. 3513–3518, 2010.

[8] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," May 2011, pp. 3607–3613.

[9] D. O. Wheeler, P. W. Nyholm, D. P. Koch, G. J. Ellingson, T. W. McLain, and R. W. Beard, "Relative navigation in GPS-degraded environments," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, Ltd, May 2016, pp. 1–10.

[10] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," p. arXiv:1812.04755, Dec 2018.

[11] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3.

[12] J. Greaves, M. Robinson, N. Walton, M. Mortensen, R. Pottorff, C. Christopherson, D. Hancock, and J. M. D. Wingate, “Holodeck: A high fidelity simulator,” 2018.

[13] M. Farrell, J. Jackson, J. Nielsen, C. Bidstrup, and T. McLain, “Error-state LQR control of a multirotor UAV,” *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2019.

[14] P. Foehn and D. Scaramuzza, “Onboard state dependent LQR for agile quadrotors,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6566–6572.

[15] J. Jackson, G. Ellingson, and T. McLain, “ROSflight: A lightweight, inexpensive MAV research and development tool,” *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 758–762, June 2016.

[16] D. O. Wheeler, D. P. Koch, J. S. Jackson, T. W. McLain, and R. W. Beard, “Relative navigation: A keyframe-based approach for GPS-degraded navigation,” *All Faculty Publications*, 2019.

[17] D. Wheeler, D. Koch, J. Jackson, G. Ellingson, P. Nyholm, T. Mclain, and R. Beard, “Relative navigation of autonomous gps-degraded micro air vehicles,” *BYU Scholars Archive, All Faculty Publications. 1962*, vol. PP, pp. 1–1, Aug 2017.

[18] DJI Technology Co., Ltd., “N3,” 2019. [Online]. Available: https://www.dji.com/n3?site=brandsite&from=nav

[19] Cloud Cap Technology, “Piccolo autopilots,” 2016. [Online]. Available: http://www.cloudcaptech.com/products/auto-pilots

[20] GmbH, HiSystems, “Mikrokopter,” 2016. [Online]. Available: http://www.mikrokopter.de/en/home

[21] Lockheed Martin, “Kestrel Flight Systems and Autopilot,” 2016. [Online]. Available: http://www.lockheedmartin.com/us/products/procerus/kestrel-autopilot.html

[22] Parrot.com, “Parrot USA,” 2016. [Online]. Available: http://www.parrot.com/usa/

[23] Ascending Technologies GmbH, “Ascending Technologies,” 2016. [Online]. Available: http://www.asctec.de/en/

[24] Cleanflight.com, “Cleanflight flight controller - 32bit multiwii,” 2016. [Online]. Available: http://cleanflight.com/

[25] L. Meier, D. Honegger, and M. Pollefeys, “PX4 : A Node-Based Multithreaded Open Source Robotics Framework for Deeply Embedded Platforms,” *International Conference on Robotics and Automation*, pp. 6235–6240, 2015.

[26] LibrePilot, “Librepilot,” 2019. [Online]. Available: https://www.librepilot.org/site/index.html

[27] Ardupilot.com, “Ardupilot | open source autopilot,” 2016. [Online]. Available: http://ardupilot.com/

[28] J. Jackson, J. Nielsen, R. Beard, and T. McLain, “Improving the robustness of visual-inertial extended kalman filtering,” *2019 IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[29] S. Atoev, K. Kwon, S. Lee, and K. Moon, “Data analysis of the mavlink communication protocol,” in *2017 International Conference on Information Science and Communications Technologies (ICISCT)*, Nov 2017, pp. 1–3.

[30] R. E. Mahony, T. Hamel, and J.-M. Pflimlin, “Complementary filter design on the special orthogonal group so(3),” *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1477–1484, 2005.

[31] R. T. Casey, M. Karpenko, R. Curry, and G. Elkaim, “Attitude Representation and Kinematic Propagation for Low-Cost UAVs,” *AIAA Guidance, Navigation, and Control (GNC) Conference*, pp. 1–25, 2013.

[32] LibrePilot, “Openpilot revolution,” 2019. [Online]. Available: https://opwiki.readthedocs.io/en/latest/user_manual/revo/revo.html

[33] P. Ru and K. Subbarao, “Nonlinear Model Predictive Control for Unmanned Aerial Vehicles,” *Aerospace*, vol. 4, no. 2, p. 31, 2017.

[34] P. N. Chikasha and C. Dube, “Adaptive Model Predictive Control of a Quadrotor,” *IFAC-PapersOnLine*, vol. 50, no. 2, pp. 157–162, 2017.

[35] G. V. Raffo, M. G. Ortega, and F. R. Rubio, *MPC with nonlinear H-infinity control for path tracking of a quad-rotor helicopter*. IFAC, 2008, vol. 17, no. 1 PART 1.

[36] M. Kamel, M. Burri, and R. Siegwart, “Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[37] C. Liu, H. Lu, and W. H. Chen, “An explicit MPC for quadrotor trajectory tracking,” *Chinese Control Conference, CCC*, vol. 2015-Septe, pp. 4055–4060, 2015.

[38] G. Ganga and M. M. Dharmana, “MPC Controller for Trajectory Tracking Control of Quadcopter,” in *MPC Controller for Trajectory Tracking Control of Quadcopter*, 2017.

[39] M. Bangura and R. Mahony, *Real-time model predictive control for quadrotors*. IFAC, 2014, vol. 19, no. 3.

[40] M. Greeff and A. P. Schoellig, “Model Predictive Path-Following for Constrained Differentially Flat Systems,” 2017.

[41] A. Aswani, P. Bouffard, and C. Tomlin, “Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter,” *Proc American Control Conference ACC to appear*, pp. 4661–4666, 2012.

[42] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 4001–4008.

[43] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," 2006.

[44] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[45] J. Solà, "Quaternion kinematics for the error-state Kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[46] D. P. Koch, D. O. Wheeler, R. Beard, T. McLain, and K. M. Brink, "Relative multiplicative extended Kalman filter for observable GPS-denied navigation," 2017.

[47] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li, "High performance full attitude control of a quadrotor on SO (3)," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1698–1703.

[48] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[49] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, "LQR control for a quadrotor using unit quaternions: Modeling and simulation," in *Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on*. IEEE, 2013, pp. 172–178.

[50] R. C. Leishman, J. C. MacDonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, 2014.

[51] D. O. Wheeler, D. P. Koch, J. S. Jackson, G. J. Ellingson, P. W. Nyholm, T. W. McLain, and R. W. Beard, "Relative navigation of autonomous GPS-degraded micro air vehicles," *All Faculty Publications*, 2017.

[52] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[53] M. Posa and R. Tedrake, "Direct Trajectory Optimization of Rigid Body Dynamical Systems Through Contact," *Algorithmic Foundations of Robotics X*, vol. 86, pp. 527–542, 2013.

[54] J. Pajarinen, V. Kyrki, M. Koval, S. Srinivasa, J. Peters, and G. Neumann, "Hybrid control trajectory optimization under uncertainty," 2017.

[55] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 2688–2693.

[56] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, “Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds,” *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[57] A. Laub, “A Schur method for solving algebraic Riccati equations,” *IEEE Transactions on automatic control*, vol. 24, no. 6, pp. 913–921, 1979.

[58] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2018.

[59] J. Mattingley and S. Boyd, “Cvxgen: A code generator for embedded convex optimization,” *Optimization and Engineering*, vol. 13, 03 2012.

[60] J. Jackson, D. Wheeler, and T. McLain, “Cushioned extended-periphery avoidance: A reactive obstacle avoidance plugin,” *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 399–405, June 2016.

[61] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao-Blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.

[62] J. Barraquand, B. Langlois, and J.-C. Latombe, “Numerical potential field techniques for robot path planning,” pp. 1012–1017 vol.2, 1991.

[63] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566 – 580, 1996.

[64] S. Koenig and M. Likhachev, “D* Lite,” *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 476–483, 2002.

[65] C. McCarthy and N. Barnes, “Performance of optical flow techniques for indoor navigation with a mobile robot,” *IEEE International Conference on Robotics and Automation*, vol. 2, no. April, pp. 5093–5098, 2004.

[66] J. R. Deming and S. Bruder, “Obstacle avoidance using image flow in an RT-Linux environment in a PC-104 platform,” *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pp. 215–219, 2004.

[67] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, “Learning monocular reactive UAV control in cluttered natural environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1765–1772, 2013.

[68] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, “Flying fast and low among obstacles,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2023–2029, 2007.

[69] H. Oleynikova, D. Honegger, and M. Pollefeys, “Reactive avoidance using embedded stereo vision for mav flight,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 50–56.

[70] J. Saunders, R. Beard, and J. Byrne, “Vision-based Reactive Multiple Obstacle Avoidance for Micro Air Vehicles,” *2009 American Control Conference, Vols 1-9*, pp. 5253–5258, 2009.

[71] S. Hrabar, “Reactive obstacle avoidance for rotorcraft UAVs,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 4967–4974, 2011.

[72] S. Schopferer and F. M. Adolf, “Rapid trajectory time reduction for unmanned rotorcraft navigating in unknown terrain,” in *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, 2014, pp. 305–316.

[73] S. Koenig and M. Likhachev, “Fast replanning for navigation in unknown terrain,” *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[74] S. Scherer, D. Ferguson, and S. Singh, “Efficient C-space and cost function updates in 3D for unmanned aerial vehicles,” *2009 IEEE International Conference on Robotics and Automation*, pp. 2049–2054, 2009.

[75] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,” *ICRA*, vol. 3, no. Figure 1, p. 5, 2009.

[76] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[77] R. C. Leishman and T. W. McLain, “Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation,” *Journal of Aerospace Information Systems*, pp. 1–17, 2014.

[78] J. Zhang, M. Kaess, and S. Singh, “Real-time depth enhanced monocular odometry,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 4973–4980, 2014.

[79] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, “Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback,” *International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[80] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust Visual Inertial Odometry Using a Direct EKF-Based Approach,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany: IEEE, 2015.

[81] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[82] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “Semi-Direct Visual Odometry for Monocular , Wide-angle, and Muti-Camera Systems,” *IEEE Transactions on Robotics*, vol. 33, pp. 249–265, 2017.

[83] C. Forster, M. Pizzoli, D. Scaramuzza, and A. Motivation, “Fast Semi-Direct Monocular Visual Odometry,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 15–22, 2014.

[84] S. Shen, N. Michael, and V. Kumar, “Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 5303–5310, 2015.

[85] F. Dellaert, “Factor graphs and GTSAM : A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012.

[86] T. Qin, P. Li, and S. Shen, “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator,” *arXiv preprint arXiv:1708.03852*, 2017.

[87] Z. Yang and S. Shen, “Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 39–51, 2017.

[88] M. Burri, M. DÃďtwiler, M. Achtelik, and R. Siegwart, “Robust state estimation for micro aerial vehicles based on system dynamics,” vol. 2015, 06 2015, pp. 5278–5283.

[89] K. M. Brink, “Partial-Update Schmidt–Kalman Filter,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2214–2228, 2017.

[90] D. P. Koch, D. O. Wheeler, R. W. Beard, T. W. McLain, and K. M. Brink, “Relative multiplicative extended Kalman filter for observable GPS-denied navigation,” 2017, available at http://scholarsarchive.byu.edu/facpub/1963/.

[91] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, “Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds,” *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[92] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, “Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback,” *International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[93] J. M. M. Montiel, J. Civera, and A. J. Davison, “Unified Inverse Depth Parametrization for Monocular SLAM.” *Proceedings of Robotics Science & Systems*, vol. 24, no. 5, pp. 16–19, 2006.

[94] M. P. Parsley and S. J. Julier, “Avoiding negative depth in inverse depth bearing-only SLAM,” *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, no. 1, pp. 2066–2071, 2008.

[95] M. George and S. Sukkarieh, “Tightly Coupled INS/GPS with Bias Estimation for UAV Applications,” *Components*, p. 7, 2004.

[96] D. B. Kingston and R. W. Beard, “Real-time attitude and gyro-bias estimation for small UAVs using low-cost sensors,” in *AIAA 3rd Unmanned Unlimited Systems Conference and Workshop*, 2004, pp. 3289–3294.

[97] J. Hall, N. Knoebel, and T. McLain, “Quaternion attitude estimation for miniature air vehicles using a multiplicative extended Kalman filter,” *Record - IEEE PLANS, Position Location and Navigation Symposium*, pp. 1230–1237, 2008.

[98] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, “An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter,” *Aerospace Science and Technology*, vol. 10, no. 6, pp. 527–533, 2006.

[99] J. N. Gross, Y. Gu, M. B. Rhudy, S. Gururajan, and M. R. Napolitano, “Flight-test evaluation of sensor fusion algorithms for attitude estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2128–2139, 2012.

[100] H. Chao, C. Coopmans, L. Di, and Y. Q. Chen, “A comparative evaluation of low-cost IMUs for unmanned autonomous systems,” *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 211–216, 2010.

[101] C. V. Angelino, V. R. Baraniello, and L. Cicala, “UAV position and attitude estimation using IMU, GNSS and camera,” *15th International Conference on Information Fusion*, pp. 735–742, 2012.

[102] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi, “A nonlinear observer for integration of GNSS and IMU measurements with gyro bias estimation,” *2012 American Control Conference (ACC)*, pp. 4607–4612, 2012.

[103] T. Konrad, M. Breuer, T. Engelhardt, and D. Abel, “State Estimation for a Multirotor using Tight-Coupling of GNSS and Inertial Navigation,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 683–11 688, 2017.

[104] G. Hu, S. Gao, and Y. Zhong, “A derivative UKF for tightly coupled INS/GPS integrated navigation,” *ISA Transactions*, vol. 56, pp. 135–144, 2015.

[105] Y. Tawk, P. Tomé, C. Botteron, Y. Stebler, and P.-A. Farine, “Implementation and Performance of a GPS/INS Tightly Coupled Assisted PLL Architecture Using MEMS Inertial Sensors,” *Sensors*, vol. 14, no. 2, pp. 3768–3796, 2014.

[106] T. Chu, N. Guo, S. Backén, and D. Akos, “Monocular camera/IMU/GNSS integration for ground vehicle navigation in challenging GNSS environments,” *Sensors*, vol. 12, no. 3, pp. 3162–3185, 2012.

[107] G. Falco, M. Pini, and G. Marucco, “Loose and Tight GNSS/INS Integrations: Comparison of Performance Assessed in Real Urban Scenarios,” *Sensors (Switzerland)*, vol. 17, no. 2, 2017.

[108] N. Sünderhauf, M. Obst, G. Wanielik, and P. Protzel, “Multipath mitigation in GNSS-based localization using robust optimization,” in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 784–789.

[109] R. M. Watson and J. N. Gross, “Robust navigation in GNSS degraded environment using graph optimization,” *CoRR*, vol. abs/1806.08899, 2018.

[110] R. Eustice, O. Pizarro, and H. Singh, “Visually augmented navigation in an unstructured environment using a delayed state history,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, vol. 1, April 2004, pp. 25–32 Vol.1.

[111] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, “Vision-based state estimation for autonomous rotorcraft MAVs in complex environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1758–1764, 2013.

[112] F. Fraundorfer and D. Scaramuzza, “Visual odometry: Part II: Matching, robustness, optimization, and applications,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[113] J. Engel, J. Sturm, and D. Cremers, “Camera-based navigation of a low-cost quadrocopter,” *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2815–2821, 2012.

[114] J. Engel, J. Sturm, and D. Cremers, “Accurate figure flying with a quadrocopter using onboard visual and inertial sensing,” *Workshop on Visual Control of Mobile Robots (ViCoMoR)*, 01 2012.

[115] Bachrach, A., S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D.Maturana, D. Fox, and N. Roy, “Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments,” *International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[116] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation,” *Robotics: Science and Systems XI*, 2015.

[117] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual-Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[118] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual-inertial odometry using nonlinear optimization,” *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[119] M. Li and A. I. Mourikis, “Improving the Accuracy of EKF-Based Visual-Inertial Odometry,” in *Robotics & Automation (ICRA)*, 2012.

[120] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Incremental smoothing and mapping,” vol. 24, no. 6, pp. 1365–1378, 2008.

[121] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “iSAM2: incremental smoothing and mapping using the Bayes tree,” vol. 31, no. 2, pp. 216–235, 2012.

[122] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, “Generic node removal for factor-graph SLAM,” *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[123] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Real-time monocular slam: Why filter?” in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2657–2664.

[124] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2502–2509.

[125] M. George and S. Sukkarieh, "Camera aided inertial navigation in poor GPS environments," *IEEE Aerospace Conference Proceedings*, 2007.

[126] J. Wang, M. Garratt, A. Lambert, J. J. Wang, S. Han, and D. Sinclair, "Integration of Gps/Ins/Vision Sensors To Navigate Unmanned Aerial Vehicles," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVII, no. Part B1, pp. 963–970, 2008.

[127] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, "GPS-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *Journal of Field Robotics*, vol. 30, no. 3, pp. 415–438, may 2013.

[128] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," 2019.

[129] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[130] "MATLAB optimization toolbox," 2019.

[131] P. G. Savage, "Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 1, pp. 19–28, jan 1998.

[132] Jianbo Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, pp. 593–600.

[133] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," *OpenCV Document, Intel, Microprocessor Research Labs*, vol. 1, 01 2000.

[134] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, June 2004.

[135] J. A. Klobuchar, "Ionospheric time-delay algorithm for single-frequency GPS users," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-23, no. 3, pp. 325–331, May 1987.

[136] J. Saastamoinen, "Atmospheric correction for the troposphere and stratosphere in radio ranging satellites," *The use of artificial satellites for geodesy*, pp. 247–251, 1972.

[137] J. Jackson, K. Brink, B. Forsgren, D. Wheeler, and T. McLain, "Direct relative edge optimization, a robust alternative for pose graph optimization," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Jan 2019.

[138] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2010.

[139] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, “An Atlas framework for scalable mapping,” *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, pp. 1899–1906, 2003.

[140] M. Bosse, P. Newman, J. Leonard, and S. Teller, “Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework,” *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[141] F. Lu and E. Milios, “Globally consistent scan matching for environment mapping,” *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[142] K. Konolige, “Large-scale map making,” in *National Conference on Artificial Intelligence*. AAAI Press, 2004, pp. 457–463.

[143] U. Frese, P. Larsson, and T. Duckett, “A multilevel relaxation algorithm for simultaneous localization and mapping,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, 2005.

[144] G. Grisetti, “Nonlinear constraint network optimization for efficient map learning,” *Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[145] E. Olson, J. Leonard, and S. Teller, “Fast iterative optimization of pose graphs with poor initial estimates,” *ICRA*, no. May, pp. 2262–2269, 2006.

[146] D. Sibley and C. Mei, “Adaptive relative bundle adjustment,” *Robotics: Science and Systems*, vol. 220, no. 3, pp. 1–8, 2009.

[147] S. Anderson, K. MacTavish, and T. D. Barfoot, “Relative continuous-time SLAM,” *International Journal of Robotics Research*, vol. 34, no. 12, pp. 1453–1479, 2015.

[148] S. Anderson and T. D. Barfoot, “Towards Relative Continuous-Time SLAM,” *ICRA*, vol. 34, no. 12, pp. 1033–1040, 2013.

[149] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, “A fast and accurate approximation for planar pose graph optimization,” *International Journal of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014.

[150] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, “Multiple relative pose graphs for robust cooperative mapping,” *IEEE International Conference on Robotics and Automation*, pp. 3185–3192, 2010.

[151] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, “Pairwise-consistent measurement set maximization for robust multi-robot map merging,” 2018.

[152] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, “Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization,” vol. 2015-June, no. June, pp. 4597–4604, 2015.

[153] P. Agarwal, G. Grisetti, G. Diego Tipaldi, L. Spinello, W. Burgard, and C. Stachniss, "Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates," *IEEE International Conference on Robotics and Automation*, pp. 3626–3631, 2014.

[154] J. Wang and E. Olson, "Robust pose graph optimization using stochastic gradient descent," *ICRA*, pp. 4284–4289, 2014.

[155] J. G. Mangelson, J. Liu, R. M. Eustice, and R. Vasudevan, "Guaranteed Globally Optimal Planar Pose Graph and Landmark SLAM via Sparse-Bounded Sums-of-Squares Programming," *arXiv e-prints*, p. arXiv:1809.07744, Sep 2018.

[156] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. MIT-CSAIL-TR-2017-002, Feb. 2017.

[157] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[158] J. Rehder, K. Gupta, S. Nuske, and S. Singh, "Global pose estimation with limited GPS and long range visual odometry," *IEEE International Conference on Robotics and Automation*, pp. 627–633, 2012.