2017-12-01

Relative Navigation of Micro Air Vehicles in GPS-Degraded Environments

David Orton Wheeler
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/6609

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu.
Relative Navigation of Micro Air Vehicles
in GPS-Degraded Environments

David Orton Wheeler

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

Randal Winston Beard, Chair
Timothy Wayne McLain
Cameron K. Peterson
Marc D. Killpack
Dah Jye Lee

Department of Electrical and Computer Engineering
Brigham Young University

Copyright © 2017 David Orton Wheeler
All Rights Reserved
ABSTRACT

Relative Navigation of Micro Air Vehicles in GPS-Degraded Environments

David Orton Wheeler
Department of Electrical and Computer Engineering, BYU
Doctor of Philosophy

Most micro air vehicles rely heavily on reliable GPS measurements for proper estimation and control, and therefore struggle in GPS-degraded environments. When GPS is not available, the global position and heading of the vehicle is unobservable. This dissertation establishes the theoretical and practical advantages of a relative navigation framework for MAV navigation in GPS-degraded environments.

This dissertation explores how the consistency, accuracy, and stability of current navigation approaches degrade during prolonged GPS dropout and in the presence of heading uncertainty. Relative navigation (RN) is presented as an alternative approach that maintains observability by working with respect to a local coordinate frame. RN is compared with several current estimation approaches in a simulation environment and in hardware experiments. While still subject to global drift, RN is shown to produce consistent state estimates and stable control.

Estimating relative states requires unique modifications to current estimation approaches. This dissertation further provides a tutorial exposition of the relative multiplicative extended Kalman filter, presenting how to properly ensure observable state estimation while maintaining consistency. The filter is derived using both inertial and body-fixed state definitions and dynamics.

Finally, this dissertation presents a series of prolonged flight tests, demonstrating the effectiveness of the relative navigation approach for autonomous GPS-degraded MAV navigation in varied, unknown environments. The system is shown to utilize a variety of vision sensors, work indoors and outdoors, run in real-time with onboard processing, and not require special tuning for particular sensors or environments. Despite leveraging off-the-shelf sensors and algorithms, the flight tests demonstrate stable front-end performance with low drift. The flight tests also demonstrate the onboard generation of a globally consistent, metric, and localized map by identifying and incorporating loop-closure constraints and intermittent GPS measurements. With this map, mission objectives are shown to be autonomously completed.

Keywords: GPS degradation, GPS denied, navigation, state estimation, observability, multiplicative extended Kalman filter, error state, sensor fusion, vision-aided INS, consistency, robocentric, multirotor, micro air vehicle, indoor flight, outdoor flight, simultaneous localization and mapping, place recognition, loop closure, pose graph optimization, obstacle avoidance, visual odometry
ACKNOWLEDGMENTS

For me, this research has been a journey through an unknown, obstacle-ridden environment. At times I thought I understood where I was and had a plan moving forward, but often global measurements were sparse and my state was quite uncertain. I am indeed grateful for advisors who regularly provided high-level advice and guidance. I am grateful for friendly lab members who supplied the regular input and feedback necessary to make progress. I will always remember and never forget the support of my parents and sweet wife. Ultimately, I am completely indebted to my Redeemer, the perfect guide in whom I can place perfect faith and “who prepareth all things before he taketh [me].”
Table of Contents

List of Tables ................................................................. x

List of Figures ................................................................. xi

Chapter 1 Introduction ...................................................... 1
  1.1 Autonomy ................................................................. 1
  1.2 Micro Air Vehicles ...................................................... 3
  1.3 GPS-Degradation ....................................................... 4
  1.4 GPS-Denied Navigation ................................................ 6
  1.5 Statement of Problem .................................................. 9
  1.6 Relative Navigation .................................................... 10

Chapter 2 Contributions .................................................... 13
  2.1 Theoretical Advantages of Relative Navigation ....................... 13
  2.2 Derivation of an Observable GPS-Denied MAV Estimator ............ 14
  2.3 Effectiveness of Relative Navigation for MAVs in Practice ........... 17
  2.4 Reactive Obstacle Avoidance ......................................... 18
  2.5 Relative Navigation Overview ....................................... 19
  2.6 Summary ................................................................. 19

Chapter 3 Relative Navigation: a Keyframe-Based Approach for Observ-
able GPS-Degraded Navigation .............................................. 21
  3.1 Introduction ............................................................. 21
3.2 State-of-the-Art MAV Navigation ........................................ 24
   3.2.1 Extended Kalman Filter ........................................... 25
   3.2.2 Global Estimation Approaches ................................. 27
3.3 Relative Navigation .................................................... 36
   3.3.1 Relative Front End .............................................. 38
   3.3.2 Global Back End ................................................ 39
3.4 Simulation ............................................................... 40
3.5 Simulation Results ...................................................... 42
   3.5.1 Accuracy .......................................................... 45
   3.5.2 Consistency ....................................................... 46
   3.5.3 Eventual Global Update ....................................... 52
   3.5.4 Summary of Implications .................................... 53
3.6 Hardware Results ...................................................... 56
3.7 Conclusions ............................................................. 59
3.8 Sidebar: Connection to Simultaneous Localization and Mapping . . . 59
3.9 Sidebar: The Banana Distribution .................................. 60

Chapter 4  Relative Multiplicative Extended Kalman Filter for Observ-
able GPS-Denied Navigation .............................................. 65
4.1 Introduction ............................................................. 65
4.2 Nomenclature .......................................................... 70
4.3 Quaternion Properties ................................................ 72
   4.3.1 Quaternion Conventions ....................................... 72
   4.3.2 Vector Rotation ................................................ 74
   4.3.3 Unit Sphere Propagation ..................................... 76
   4.3.4 Time Integration ............................................... 78
   4.3.5 Attitude Kinematics .......................................... 81
   4.3.6 Error State .................................................... 82
4.3.7 Euler Decomposition ................................. 84
4.3.8 Error Quaternion Properties ....................... 86
4.4 Multiplicative Extended Kalman Filter ................ 88
  4.4.1 Propagation ......................................... 90
  4.4.2 Measurement Update ............................... 94
  4.4.3 Delayed Out-of-Order Measurements ............... 98
4.5 Relative Navigation .................................. 98
4.6 Inertial Relative Navigation (iRN) ..................... 102
  4.6.1 State Dynamics ..................................... 102
  4.6.2 Measurement Models .............................. 109
  4.6.3 Keyframe Reset .................................... 117
4.7 Body-fixed Relative Navigation (bRN) ................. 119
  4.7.1 State Dynamics .................................... 119
  4.7.2 Measurement Models .............................. 122
  4.7.3 Keyframe Reset .................................... 127
4.8 Results ................................................ 128
4.9 Conclusions .......................................... 132
4.10 Zero-Mean Error State ................................ 133
4.11 Inertial Dynamics .................................... 135
  4.11.1 Position .......................................... 136
  4.11.2 Attitude .......................................... 136
  4.11.3 Velocity ........................................... 138
4.12 Body-Fixed Dynamics .................................. 139
  4.12.1 Position .......................................... 140
  4.12.2 Attitude .......................................... 141
  4.12.3 Velocity ........................................... 143
4.13 Keyframe-Reset Derivation ............................ 143
Chapter 5  Relative Navigation of Autonomous GPS-Degraded Micro Air Vehicles

5.1 Introduction ................................................. 153
5.2 Related Work ............................................. 156
5.3 Relative Navigation Overview ...................... 158
5.4 Relative Front End ........................................ 160
   5.4.1 View-Based Odometry ............................... 160
   5.4.2 Relative State Estimation ......................... 161
   5.4.3 Relative Path Planning and Control ............ 164
5.5 Global Back End .......................................... 166
   5.5.1 Pose-Graph Map .................................... 168
   5.5.2 Place Recognition .................................. 170
   5.5.3 GPS Integration ..................................... 171
   5.5.4 Optimization ...................................... 173
   5.5.5 Global Path Planning ............................... 175
5.6 Experimental Setup ...................................... 176
   5.6.1 Flight test 1: Outdoor GPS-denied .............. 179
   5.6.2 Flight test 2: Indoor GPS-denied ................ 180
   5.6.3 Flight test 3: Indoor/outdoor intermittent GPS 181
5.7 Results ..................................................... 183
   5.7.1 Estimation Accuracy and Consistency ............ 184
   5.7.2 Map Optimization .................................. 186
   5.7.3 Autonomous Flight .................................. 190
Chapter 6  Cushioned Extended-Periphery Avoidance: a Reactive Obstacle Avoidance Plugin

6.1 Introduction .............................................. 193
6.2 ROAP Motivation ........................................... 195
6.3 CEPA Algorithm Description ................................. 198
  6.3.1 Steering Algorithm ...................................... 199
  6.3.2 Map Memory ............................................ 202
  6.3.3 Emergency Avoidance ................................. 203
6.4 Experimentation and Results ................................ 204
  6.4.1 Simulation Results ..................................... 205
  6.4.2 Hardware Results ..................................... 207
6.5 Conclusions ................................................ 210

Chapter 7  Relative Navigation in GPS-Degraded Environments  .... 211

7.1 Introduction .............................................. 211
7.2 Relative Navigation Framework .............................. 213
  7.2.1 Relative Front End Overview .......................... 214
  7.2.2 Global Back End Overview ............................ 214
  7.2.3 Motivating Scenarios ................................. 215
7.3 Relative Front End .......................................... 216
  7.3.1 Visual Odometry ....................................... 216
  7.3.2 Estimation ............................................. 217
  7.3.3 Low-Level Path Generation and Following .............. 220
  7.3.4 Control ............................................... 221
7.4 Global Back End ............................................ 222
  7.4.1 Pose Graph ........................................... 222
## List of Tables

1.1 Sources of GPS uncertainty [1] ........................................... 5

3.1 Estimation framework legend ........................................... 28
3.2 Summary comparison of the estimation approaches ...................... 45

5.1 Hardware details ............................................................. 178
5.2 Summary of flight tests ...................................................... 179

6.1 Simulation results for scenario 1 ......................................... 206

7.1 Sources of GPS uncertainty ................................................ 212
7.2 Hardware details ............................................................. 213
List of Figures

1.1 Comparison of the true MAV trajectory with reported GPS measurements .......................... 6
1.2 A relative keyframe filter, such as relative navigation, builds upon ideas presented in the SLAM literature to ensure reduced computational complexity and improved consistency .................................................... 10
1.3 Relative navigation architecture ........................................................................................... 11
1.4 2D illustration of node frame reset operation ....................................................................... 12
3.1 GPS is particularly prone to degradation or dropout when flying near the ground. ................................. 22
3.2 Comparison of estimator architectures including pseudo-global, stochastic cloning, keyframe robocentric, and relative navigation .......................................................... 28
3.3 Relative navigation architecture ........................................................................................... 37
3.4 Example simulation for a single trial showing the trajectory estimated by each approach ........................................................ 43
3.5 Average position and absolute heading error over each trial on each trajectory ......................... 46
3.6 Estimation error for one of the trajectories .............................................................................. 47
3.7 Normalized estimation error squared probability density function for each estimation approach with the ideal $\chi^2_3$ pdf overlaid ................................................................. 50
3.8 Average normalized estimation error squared (NEES) over all trials and all trajectories as a function of time .......................................................... 51
3.9 Diagram summarizing the implications that prolonged GPS dropout and heading noise have when working in a global coordinate frame ......................................................... 54
5.7 Illustration of how a pose graph is used to reconstruct the global state . . . . 168
5.8 Example loop closure detected using FAB-MAP . . . . . . . . . . . . . . . . 171
5.9 Back-end GPS integration method . . . . . . . . . . . . . . . . . . . . . . . 172
5.10 Illustrations of how incorporating global information influences vehicle control 177
5.11 The vehicle used for the flight tests . . . . . . . . . . . . . . . . . . . . . . . 177
5.12 The data flow and networking between the various system components . . . 178
5.13 Flight test 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180
5.14 Large MAV smoothly navigating through a tight, nondescript hallway . . . 181
5.15 Flight test 3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
5.16 Pose-graph map for the first 130 m of flight test 2 . . . . . . . . . . . . . . 184
5.17 Pose-graph map for flight test 1 . . . . . . . . . . . . . . . . . . . . . . . . . 185
5.18 Pose-graph map for flight test 2 . . . . . . . . . . . . . . . . . . . . . . . . . 187
5.19 Pose-graph map for flight test 3 . . . . . . . . . . . . . . . . . . . . . . . . . 188

6.1 MAV priorities in general . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
6.2 Block diagram illustrating how ROAP supplements an existing path planner
   by modifying commands . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 196
6.3 An example steering configuration . . . . . . . . . . . . . . . . . . . . . . . . . 200
6.4 A visual description of the way memory is kept in the reactive planner . . . 203
6.5 Illustration of emergency avoidance . . . . . . . . . . . . . . . . . . . . . . . . 204
6.6 Scenario 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
6.7 Scenario 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 208
6.8 Hardware validation of CEPA in a GPS-denied environment using strictly
   onboard computation and sensing . . . . . . . . . . . . . . . . . . . . . . . . . . 209

7.1 The multirotor vehicle used for the example implementation in this paper . . 213
7.2 Relative navigation framework . . . . . . . . . . . . . . . . . . . . . . . . . . . 214
7.3 2D illustration of estimated and true states relative to the current node frame 215
7.4 Performance plots for a MEKF implementation using the relative navigation framework .................................................. 219
7.5 Control system architecture for the example implementation .................................................. 221
7.6 A loop closure edge over-constrains the back-end pose graph. After optimization, the estimated node frame states will more closely resemble truth. .................. 224
7.7 A virtual zero vertex is added to the pose graph and represents the origin of a global coordinate system ........................................ 226
7.8 Map before and after optimization .......................................................... 227
7.9 Illustration of mass spring optimization example .................................................. 228
Chapter 1

Introduction

For a moment, try to imagine what the world would be like today if the personal computer was never developed. The change would be so monumental that we may only be able to picture this scenario by recalling previous decades. The advent of the personal computer directly impacts how we communicate, share information, and work. However, its indirect influence is much more widespread, affecting commerce, entertainment, politics, social opinion, and more! It is incredible to reflect how the computer has gone from a room-sized luxury to an essential commodity in the matter of a few decades.

Perhaps a similar revolution is underway for autonomous systems. Historically, many robotic applications have been restricted to factory settings, carefully isolated to avoid injuries. Robots were generally designed to perform a specific task and were expensive; however, as computing resources continue to become smaller and more powerful, and as methods for leveraging vast datasets continue to mature, many new applications of robotics become a reality. Clearly society will be directly affected in the coming decades by ideas like self-driving cars and drone delivery. These developments, together with the unforeseen, indirect effects of robot autonomy, may ultimately define the next several decades.

1.1 Autonomy

What are the overarching technical hurdles that hold back this robotic revolution? Before many autonomous systems become viable, high levels of safety and reliability must be achieved. While autonomous systems have the potential to streamline productivity and im-
prove safety, a malfunction could yield serious consequences. Imagine the consequences if a perception algorithm in a self-driving car mistakes a small child on the curb for a fire hydrant or misjudges the intent of a merging tractor trailer nearby. Autonomous systems ultimately must make correct decisions in the presence of uncertainty.

For most autonomous systems to be successful, they must accurately estimate their own state, such as position, attitude, and velocity, in addition to estimating model parameters such as sensor biases, mass, or drag coefficients. Robot autonomy also requires accurate estimation of the surrounding environment including potential obstacles, dynamic objects, or neighboring landmarks. In addition to accurate state estimates, a robot needs to understand the uncertainty associated with these estimates. In short, how certain or reliable are these states or measurements at any given time? When a state estimator properly models the underlying uncertainty of the system, being neither overly confident nor overly conservative, the filter is considered to be consistent. Proper uncertainty estimation is critical for robot path planning, vehicle coordination, and optimal sensor fusion. Yaakov Bar-Shalom, a pioneer in modern estimation techniques, describes the importance of proper uncertainty estimation for probabilistic filters such as the Kalman filter. He states using the following emphasis:

“Since the filter gain is based on the filter-calculated error covariances, it follows that consistency is necessary for filter optimality: Wrong covariances yield wrong gain. This is why consistency evaluation is vital for verifying a filter design – it amounts to evaluation of estimator optimality.” [2]

Autonomous systems must accurately understand their state and local environment while correctly acknowledging in what ways their understanding may be deficient. The issues of reliability, accuracy, and consistency are particularly relevant for micro air vehicles (MAVs).
1.2 Micro Air Vehicles

As processing, sensing, and battery technologies continue to develop, there are increased opportunities for MAVs to contribute to society. Economists anticipate that autonomous MAVs will give rise to a handful of billion-dollar markets, including infrastructure inspection, security, precision agriculture, transportation, and delivery [3]. Using MAVs to inspect bridges, dams, chemical plants, and refineries is particularly motivating as it would take the place of dangerous, time-consuming, and expensive human inspections; however, these markets are still largely speculative because autonomous MAV navigation is an active research problem with significant challenges.

Micro air vehicles have fast dynamics that are inherently unstable. In other words, if left to its own devices for even a short period of time, a MAV will likely crash; picture a broom handle precariously balanced on a finger. To reliably stabilize the system, particularly in tight environments, a MAV requires low-latency, high-bandwidth state estimates that are unbiased and free of state jumps. To accomplish this, many current autonomous MAVs require external sensing or a priori information, such as a motion capture system, Global Positioning System (GPS) measurements, or detailed maps of the environment for localization. Other systems are accompanied with strong, limiting assumptions, such as a highly-structured environment – vertical walls, flat floors, stationary scenes, and so forth.

Another prominent challenge that MAVs present is their size, weight, and power limitations. Many applications require a small vehicle footprint with a long flight time. To meet these requirements, many MAV platforms must settle on using fewer sensors and less powerful processors than is typical for ground robots. At times processing can be performed off-board and communicated to the MAV wirelessly, but such architectures introduce additional failure points that limit reliability. Further, when MAVs are operating in confined environments, strong disturbances such as ground and wall effect can degrade performance.
The majority of autonomous MAVs currently in operation use GPS with an inertial navigation system (GPS/INS) for state estimation and control. High-rate accelerometer and gyroscope measurements are integrated to estimate change in position and attitude. When properly calibrated, these measurements are remarkably accurate over small time steps and are commonly used as inputs to the estimator’s dynamics. Integrating noise, however, ultimately causes these estimates to drift. GPS measurements constrain this drift but are available at a slower rate. Commonly a probabilistic filter, such as an extended Kalman filter (EKF), fuses measurements from these two sensors using a model of the vehicle dynamics. When GPS is available and reliable, GPS/INS solutions work well for global missions requiring position hold or waypoint following. GPS/INS systems have been thoroughly researched and are widely used.

The integration of autonomous aircraft into mainstream life will depend in large part on the ability of MAVs to safely and effectively operate in cluttered, non-structured, tight, and unknown environments. While autonomous flight is currently possible in specialized circumstances, the development of real-time, onboard approaches that are robust to GPS degradation or dropout remains an open field of research.

1.3 GPS-Degradation

A MAV’s heavy reliance on GPS is a noted obstacle for navigation robustness. 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.” [4]

In the absence of global measurements, the global position and heading states of a MAV are not observable [5]. This means no combination of available inputs and measurements and
Table 1.1: Sources of GPS uncertainty [1]

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multipath</td>
<td>Signal bounces before reaching receiver (false pseudo-range)</td>
</tr>
<tr>
<td>Number of satellites</td>
<td>Few visible satellites increase sensitivity to timing errors</td>
</tr>
<tr>
<td>Dilution of precision</td>
<td>Visible satellites are poorly spaced</td>
</tr>
<tr>
<td>Spoofing</td>
<td>Signal is locally recreated with false information</td>
</tr>
<tr>
<td>Atmospheric delays</td>
<td>Signal is delayed due to ionosphere and troposphere influences</td>
</tr>
</tbody>
</table>

their time derivatives can be utilized to determine the absolute global position and heading of the vehicle. Without such feedback, these states will inevitably drift and the uncertainty estimates become inconsistent [6–8]. While these issues can be mitigated by using better sensors and models, leveraging a priori information about the environment, or limiting the duration of GPS-dropout, the core deficiencies persist. GPS not only provides global position estimates to constrain the drift introduced by noisy rate sensors like MEMS-based inertial measurement units (IMUs) but also provides a way to estimate ground speed and orientation, allowing the vehicle to estimate wind effects.

A robust navigation solution cannot naively assume GPS measurements are always accurate or always available. Table 1.1 describes several ways GPS measurements can become degraded. These issues are illustrated in Figure 1.1 where the true MAV trajectory is compared with the reported GPS measurements. Naively incorporating these GPS measurements would undoubtedly degrade the safety and stability of the system.

Beyond measurement degradation, GPS measurements may become totally unavailable during sensor failure, in the presence of GPS jammers, when shadowed by buildings or foliage, and simply cannot be applied indoors. The following story illustrates how easily GPS can be blocked. In 2013, the Newark airport was testing a new air-traffic control system, when they noticed that every morning and afternoon the system went offline as GPS became unavailable. The Federal Communications Commission ultimately identified the cause: for less than $100, a man installed a GPS-jammer on his company truck to prevent his boss from
Figure 1.1: 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. Later the GPS measurements improve, though gross outliers persist. Image first published in [9].

tracking him during the day. His daily commute along the New Jersey Turnpike took him past the airport, incapacitating the system [10, 11]. Various sources report that GPS loss for MAVs, even for a brief period, often results in catastrophic failure. As a result, GPS-denied navigation has become a strong emphasis of research over the last decade.

1.4 GPS-Denied Navigation

A vast amount of research and development has been invested into GPS-denied navigation. There are several high-level paradigms for approaching the problem, each with their own assumptions and example use cases.

Localization A common approach to GPS-denied navigation is localization. In this approach detailed maps of the environment are required a priori. The autonomous vehicle then uses its sensors to perceive its local environment and localize itself within the map. If the map is globally referenced, the vehicle can infer its global position through the map.
Self-driving cars employ this approach, where discrepancies between the current sensor measurements and the map suggest dynamic objects. This approach assumes detailed maps are available, up-to-date, and can be stored and quickly processed online. These assumptions cannot be satisfied for many MAV applications.

**Simultaneous localization and mapping** When a vehicle enters an unknown, GPS-denied environment, it must estimate both its state and the locally observed environment. The simultaneous localization and mapping (SLAM) problem solves for the most probable vehicle trajectory and landmark locations, given a time history of inputs and relative measurements to the local environment [12, 13].

Maintaining consistency during SLAM has been a prominent research focus. While traditional SLAM approaches estimate the vehicle and landmark locations with respect to a global coordinate frame, such parameterizations lose consistency as heading uncertainty increases [7]. Because consistency is important for proper sensor fusion, an increasing number of relative SLAM implementations are being published. Robocentric approaches estimate the pose of landmarks and the global origin with respect to the vehicle’s current position and attitude [14]. Relative submaps estimate the state of the vehicle and landmarks with respect to a local inertial coordinate frame [15, 16]. These submaps are subsequently fused and form a more consistent global estimate.

When adapting a SLAM approach that was originally developed for ground robots to be used for MAV navigation, additional considerations must be taken:

1. **Platform Limitations:** Size, weight, and power (SWaP) constraints restrict many MAV platforms from carrying the sensors and processors that are commonly found on sophisticated ground robots. SWaP constraints are particularly relevant when long flight times are required.
2. *Navigation vs. Estimation:* The fast, inherently-unstable MAV dynamics necessitate smooth, timely state estimates. Some SLAM formulations are blind to these navigation-specific requirements, often violating them for the sake of optimality. While a ground robot can usually pause to allow an estimator to converge, a MAV does not have that option.

**Vision-aided navigation**  Often, simplifying assumptions are made to reduce the computational burden on resource-constrained platforms. While full-SLAM solves for the most likely vehicle trajectory, online-SLAM, such as EKF-SLAM [17], solves only for the current state of the vehicle and nearby features or landmarks [18]. Keyframe filters further simplify the estimation problem by tracking a single keyframe rather than many individual landmarks. These simplified approaches are commonly referred to as vision-aided navigation or visual inertial navigation systems (V-INS) and are common for MAV systems due to their size, weight, and power constraints. Vision-aided navigation is a particular realization of SLAM that assumes the vehicle’s state evolves according to a Markov process.

Vision-aided navigation approaches typically utilize odometry computed from exteroceptive sensors such as cameras or laser scanners. Many such odometry algorithms exist for a variety of sensors, and include methods such as visual odometry [19, 20] and laser scan matching [21, 22]. Odometry can be computed either between consecutive frames (images or scans), or between the current frame and a *keyframe*. When a keyframe is used, a series of odometry measurements are computed with respect to this common, fixed reference frame. Typically, the keyframe is updated only when there is insufficient overlap to provide a reliable odometry measurement. As a result, keyframe-based odometry reduces temporal drift in the computed odometry as compared to frame-to-frame matching [23, 24].
Overall, a wide variety of techniques are employed to tackle the GPS-denied navigation problem, each with their own assumptions, strengths, and limitations. Some MAV navigation solutions are derived from the EKF-SLAM framework [25, 26], while others avoid filtering techniques altogether, electing to solve for the MAV’s pose using non-linear optimization techniques such as factor graphs [27]. Yet many others choose to use keyframe-based approaches for their reduced computational complexity [21, 23, 24, 28, 29]. A fundamental deficiency of all vision-aided navigation systems, however, is that the global position and heading of the vehicle is unobservable when global information such as GPS is not available [5, 30, 31]. State estimates eventually drift and referencing these unobserved states in a controller can destabilize the system. Further, this unobservability causes uncertainty estimates to become susceptible to modeling and linearization errors, leading to inconsistency [7]. Such issues reduce the reliability of MAVs, limiting the extent that MAVs can be leveraged for many applications.

This dissertation encourages a paradigm shift within the GPS-denied navigation literature. Rather than conceding the loss of observability, the MAV’s state can be defined with respect to its local environment such that relative measurements from vision sensors preserve observability. The focus of this dissertation is to present a relative keyframe filter that meets the practical limitations of MAVs while maintaining consistency during prolonged GPS degradation or dropout. Such a navigation solution should maintain observable state estimation, ensure that flight-critical control does not degrade as GPS degrades or drops out, place safety as the highest priority, yet be able to opportunistically incorporate GPS to fulfill global missions. This dissertation presents the relative navigation framework as a solution. Relative navigation is the logical method to improve the consistency of global keyframe approaches, as illustrated in Figure 1.2. The relative navigation framework is briefly introduced in the following section, and described in further detail in sections 3.3, 4.5, and 5.3.
Figure 1.2: A relative keyframe filter, such as relative navigation, builds upon ideas presented in the SLAM literature to ensure reduced computational complexity and improved consistency. This dissertation presents the relative navigation framework as a promising alternative to global keyframe filters that are commonly used for MAV navigation today.

1.6 Relative Navigation

Leishman et al. introduced the relative navigation framework which decouples flight-critical estimation, guidance, and control algorithms from potentially erroneous or delayed global updates such as GPS or map-based loop-closure constraints [28]. This idea parallels human navigation, where a driver can safely navigate by referencing the local environment – lane markers, pedestrians, and nearby vehicles – even when disoriented or completely lost. Incorporating global information from a map or GPS is best delegated to a separate agent. Figure 1.3 presents the relative navigation architecture, showing how these separate responsibilities are decoupled into a relative front end and global back end.

In the relative front end, a view-based odometry algorithm, such as visual odometry or laser scan matching, provides updates for how the vehicle has moved relative to a recent state when a keyframe image or scan was registered. New keyframes are declared as necessary, whenever there is insufficient overlap to provide an accurate update. For each keyframe, a node frame is declared, defined as the gravity-aligned coordinate frame positioned on the ground directly under the true position of the vehicle when the keyframe image or scan was taken. An observer estimates the position and attitude of the vehicle with respect to
Figure 1.3: The vehicle-mounted sensors feed a probabilistic filter estimating the vehicle’s state relative to its local environment. Local path planning and control stabilize the vehicle in this local frame. Meanwhile, on a distinct thread, a time history of local information can be fused with any available global information to form a global map. The back end only influences the flight-critical front end in the form of global goals represented in the current relative coordinate frame.

the current node frame, allowing the estimator to be independent of global measurements and global state estimates. In this way the front end only makes use of observable states for estimation, guidance, and control. As illustrated in Figure 1.4, when a new keyframe, and therefore node frame, is declared, the position and attitude portions of the state are reset. Because the vehicle is by definition located exactly at the keyframe, the horizontal position and heading are reset to zero and their corresponding uncertainty is removed from the front-end filter. The estimated difference between node frames and the corresponding uncertainty is then used in the global back end to reconstruct the global state. As necessary, optimization methods fuse GPS measurements or loop-closure constraints from place recognition algorithms to improve the global state. The back end only influences the flight-critical front end in the form of global goals represented in the current relative coordinate frame.

The motivation for and benefits of the relative navigation framework are discussed more thoroughly throughout the dissertation. Chapter 2 outlines the contributions of this dissertation, Chapters 3 through 7 present the research, and Chapter 8 concludes.
Figure 1.4: 2D illustration of node frame reset operation. (a) The front-end filter estimates the vehicle’s $p_x$, $p_y$, and $\psi$ with respect to the current node frame $n$ (blue). The filter’s covariance estimate (blue oval) and true vehicle trajectory (green) are also shown. (b) When a new node frame is declared, the estimated $p_x$, $p_y$, and $\psi$ (gray line) along with the associated covariance (gray oval) are saved as an edge in the global back-end pose graph. Node frame $n + 1$ is then defined at the current true state (with roll and pitch removed). As a result, the filter zeros $p_x$, $p_y$, and $\psi$ and their corresponding covariance values. The result is that the state error and uncertainty are removed from the front-end filter and delegated to the global back end.
Chapter 2

Contributions

This chapter outlines the contributions provided in this dissertation. While the relative navigation framework was proposed previously as an intuitive approach to GPS-denied MAV navigation [28], this dissertation presents the theoretical advantages that relative navigation provides. It compares relative navigation to existing approaches, first conceptually and then in simulation and hardware. The relative navigation framework is implemented on a multirotor and its effectiveness is demonstrated during several prolonged flight tests. Much of the research has been published or is currently being prepared for publication and is included as Chapters 3-7 in that form.

2.1 Theoretical Advantages of Relative Navigation

Chapter 3


Current filter-based approaches to GPS-denied navigation directly estimate the global state of the vehicle. In Chapter 3 we demonstrate for such systems that, because of unobservability, estimation and control performance can degrade significantly during periods of prolonged GPS dropout and heading uncertainty. We present as an alternative the relative navigation framework, which maintains full-state observability in spite of GPS dropout by estimating with respect to a local reference frame. While relative navigation, like all GPS-denied navigation approaches, is subject to global drift, it maintains a more accurate estimate
of global uncertainty and so provides better inputs to techniques such as map optimization using loop closures that help to reduce this drift. Relative navigation facilitates consistent state estimation and stable control, thereby improving the overall safety and reliability of MAVs.

Chapter 3 promotes a paradigm shift within the GPS-denied navigation literature. Many researchers are accustomed to working with respect to a global reference frame, and as a result concede that state observability is inevitably lost in the absence of global updates. Chapter 3 highlights the issues associated with such a concession and provides a viable alternative. In particular, we show that unobservability leads to a loss of estimator consistency. Inconsistency implies a loss of estimator optimality. By subtly restructuring the problem, relative navigation avoids this and other pitfalls that are prevalent in GPS-denied navigation systems. In terms of implementation, the modifications that need to be made to an existing keyframe-based global filter implementation to convert it to relative navigation are relatively minor and straightforward.

The main contribution of Chapter 3 is a rigorous analysis and comparison of the performance of current GPS-denied estimation approaches in simulation. We show that the relative navigation framework provides significant advantages in terms of accuracy, consistency, and its ability handle global updates after a prolonged GPS outage. To demonstrate that the findings in the simulation apply to real-world environments, we also present limited multirotor flight-test results for each of the approaches.

2.2 Derivation of an Observable GPS-Denied MAV Estimator

Chapter 4
Relative Multiplicative Extended Kalman Filter for Observable GPS-Denied Navigation.
Chapter 4 derives an estimator for a micro air vehicle that maintains observability by working with respect to a relative coordinate frame. This paper provides the following contributions:

**MEKF Tutorial.** Significant portions of the paper are tutorial in nature, clearly motivating why an indirect or error state formulation is necessary when quaternions are used to represent attitude, and providing complete explanations of each step in the derivation of the filter equations. While other publications discuss the MEKF at length, this paper provides several meaningful extensions. First, this paper derives an estimator for the full state of a MAV (position, velocity, attitude, accelerometer biases, and gyroscope biases), while most previous MEKF papers of similar scope discuss only the attitude and bias estimation. Second, this paper derives the MEKF using the Hamilton quaternion convention as opposed to the JPL convention used in some other works. While the choice of quaternion convention does not fundamentally change the problem, Hamilton quaternions are commonly used in the robotics literature and subtle but important differences arise. This paper provides a contrasting perspective to help deepen understanding of quaternions. Third, the tutorial nature of this paper provides sufficient context for the derivation of several new properties relating to quaternions, their error representations, and their Euler-angle decomposition. These properties play a key role in the derivation of the RMEKF to allow partial attitude updates.

**RMEKF Derivation.** Another purpose of this paper is to provide a thorough derivation of the RMEKF estimator successfully used in [32] for prolonged UAS navigation in GPS-degraded environments. The RMEKF presented in this paper extends the original RMEKF derivation in [33] in several important ways that have proven necessary for prolonged flight. First, this paper presents a new visual odometry measurement model and keyframe reset operation, which together ensure the state remains observable in GPS-denied environments.
Second, several novel properties of error quaternions are derived that enable partial updates to quaternion states and their covariances. Third, new terms are added to the state vector in this paper to correctly account for uncertainty in the roll, pitch, and altitude of the vehicle at the time a keyframe is declared. Finally, smaller differences include reversing the direction of the odometry measurement model to avoid unnecessarily coupling heading uncertainty into the update, and estimating the global height of the vehicle above ground rather than treating altitude as a relative state.

**Inertial and Body-Fixed Dynamics.** Another unique contribution of this paper is the derivation of the RMEKF when the state is defined with respect to either an inertial frame or a body-fixed frame. Using an inertially-fixed, gravity-aligned frame is the approach given in [28,33], and is the more traditional way of expressing the vehicle dynamics. In recent years, however, robocentric approaches, such as those in [14,26], have become popular as a method for addressing some of the inconsistency issues of traditional EKF-SLAM approaches. It is demonstrated in [8] that relative navigation obtains these same benefits without the need to invert the vehicle dynamics, but also that either robocentric or inertial dynamics can be used within the relative navigation framework. Another contribution of this paper is a presentation of the subtle differences that arise between using an inertial and body-fixed reference frame. For example, in addition to the change in dynamics, subtle changes appear in the quaternion integration, error state definition, measurement models, and keyframe reset operations. By presenting both formulations side-by-side, these differences are clearly outlined.

**Self-Contained Derivation.** The final contribution of this paper is a complete, self-contained derivation of the filter and all relevant quaternion properties. The definitions of quaternions and error states used across the current estimation literature differ in subtle ways. When these definitions are not thoroughly documented, it becomes difficult to cor-
rectly leverage properties from multiple sources. With its tutorial nature and step-by-step explanations, this paper is designed to present a complete, self-contained derivation with respect to a consistent, explicitly stated definition. This allows the reader to understand, implement, and potentially modify the RMEKF for new vehicles or applications. Note that while the keyframe reset step and several measurement models are specific to relative navigation, the propagation equations and general filter structure are equally relevant for other applications, such as GPS/INS navigation.

This work was performed in close collaboration with Daniel Koch where we both made significant contributions in theory and writing. Daniel and I jointly derived the RMEKF in the inertial frame. I extended this by deriving the RMEKF in the body frame and implemented both estimators in hardware. Overall, Daniel and I are equal contributors to this paper.

2.3 Effectiveness of Relative Navigation for MAVs in Practice

Chapter 5

Developing dependable, autonomous MAV solutions that are robust to GPS degradation is a challenging but highly relevant field of research. Chapter 5 demonstrates that the relative navigation framework offers a compelling alternative paradigm for approaching the problem. By decoupling flight-critical estimation, guidance, and control algorithms from unobservable global states that are prone to inconsistency and state jumps, relative navigation avoids many issues that plague other state-of-the-art approaches.

The contributions of Chapter 5 are twofold. First, the details necessary to implement the complete relative navigation framework are presented. Specifically, we describe the relative estimator reset operation necessary to maintain observability, and present the relative
guidance and control strategy necessary to ensure smooth, stable flight. We discuss how to
reconstruct the global state with consistent banana-shaped uncertainty distributions, and
describe how to incorporate GPS and loop-closure information to improve the global state
estimate. We explain how the high-level path planner facilitates autonomous missions and
show how to leverage off-the-shelf algorithms for visual odometry, place recognition, and
robust pose-graph optimization.

The second contribution consists of several prolonged hardware flight tests demonstrating
the effectiveness of relative navigation for autonomous GPS-degraded MAV navigation in
varied, unknown environments. We demonstrate that the relative front end successfully
fuses multiple vision sensors, works indoors and outdoors, and results in low drift with no
state jumps. We further demonstrate the onboard generation of a globally-consistent, metric,
and localized map by identifying and incorporating loop-closure constraints and intermittent
GPS measurements. Using this map, we demonstrate the fully-autonomous completion of
mission objectives, including performing a position-hold about a global position waypoint
while in a GPS-denied environment. In many ways, these flight test results meet or exceed
the current state-of-the-art.

2.4 Reactive Obstacle Avoidance

Chapter 6
Cushioned Extended-Periphery Avoidance: a Reactive Obstacle Avoidance Plugin

While collision avoidance and flight stability are generally a MAV’s highest priority, many
map-based path planning algorithms focus on path optimality, often assuming a static, known
environment. For many MAV applications a robust navigation solution requires respond-
ing quickly to obstacles in dynamic, tight environments with non-negligible disturbances.
Chapter 6 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. Often sensors have a limited field of view, making reactive obstacle avoidance difficult. CEPA introduces a fast, efficient method for providing a 360 degree lower-bound understanding of the environment by representing recent laser scans in the current body-fixed polar coordinate frame. With this extended field of view, motion assumptions common in other reactive planners can be relaxed and emergency control effort can be applied in any direction. CEPA is validated in simulation and on hardware in a GPS-denied environment using strictly onboard computation and sensing.

This work was performed in close collaboration with James Jackson where we both made equal contributions. I helped formalize the idea of the ROAP framework and made substantial contributions to the theoretical development of CEPA. I also led the effort in writing the paper.

2.5 Relative Navigation Overview

Chapter 7
Relative Navigation in GPS-Degraded Environments

Chapter 7 provides a high-level overview of the relative navigation architecture, emphasizing the intuitive nature of the approach and providing several motivating examples. The role of each component of the architecture is presented. We describe how to reconstruct the global state and opportunistically incorporate GPS and loop closure constraints for the accomplishment of global missions. Finally, we demonstrate in hardware how the complete architecture works together.

2.6 Summary

In summary, this dissertation offers the following contributions:
• Identifies that state observability is not inevitably lost in GPS-denied environments when the relative navigation framework is leveraged.

• Demonstrates that relative navigation is statistically more accurate and consistent than current state-of-the-art frameworks.

• Presents a relative state estimator for a multirotor that remains observable in GPS-denied environments. The estimator is derived for both inertial and body-fixed parameterizations.

• Provides a tutorial exposition on the MEKF using Hamilton quaternions.

• Demonstrates that the RMEKF effectively fuses relative measurements from a variety of sensors to enable prolonged flight tests.

• Presents and demonstrates the complete relative navigation framework, including the structuring of a pose-graph back end.

• Opportunistically incorporates loop-closure and GPS constraints to form a globally consistent map.

• Demonstrates fully autonomous MAV navigation in varied environments using the relative navigation architecture.

• Introduces the ROAP framework for incorporating reactive obstacle avoidance into an existing system, and presents CEPA as an effective algorithm for MAV obstacle avoidance in practice.
Chapter 3

Relative Navigation: a Keyframe-Based Approach for Observable GPS-Degraded Navigation

3.1 Introduction

As relevant technologies become smaller and less expensive, micro air vehicles (MAVs) are transitioning from predominantly military and hobbyist applications to mainstream use. Exciting new applications include delivery of medical supplies to remote areas, infrastructure inspection, environmental change detection including precision agriculture, surveillance tasks including fire or traffic monitoring, and the film and entertainment industry. However, before MAVs become fully integrated into society and the airspace, higher levels of safety and reliability must be assured.

One of the factors that most limits MAV robustness is their heavy reliance on consistent and accurate measurements from satellite navigation systems such as the Global Positioning System (GPS). These measurements provide regular updates of global position, heading, and velocity, directly influencing state estimation and control. However, GPS is susceptible to degradation and dropout, as illustrated conceptually in Figure 3.1. The weak signal can be easily blocked by buildings and foliage, jammed, or spoofed. Further, the measurement quality can degrade due to multipath signals, atmospheric delays, or the number and position of visible satellites. These issues are particularly prevalent when flying near the ground, where safety and reliability are especially important.

---

1 This paper is a forthcoming publication in IEEE Control Systems Magazine and was written by David O. Wheeler, Daniel P. Koch, James S. Jackson, Timothy W. McLain, and Randal W. Beard [8].
Figure 3.1: GPS is particularly prone to degradation or dropout when flying near the ground.

To circumvent these difficulties, many GPS-denied navigation solutions have been developed that utilize relative measurements from algorithms such as visual odometry or laser scan matching [21, 23, 24, 33]. Even with these measurements, however, when GPS is unavailable the global position and heading states are not observable [30, 31], as shown in the observability analysis in [5]. This means there is no guarantee that these states can be accurately reconstructed from the available inputs and measurements. Specifically, non-observability induces three main difficulties:

1. **Global drift:** Integrating noisy inputs without correction will cause the global state to drift arbitrarily far from truth.

2. **Estimator inconsistency:** An inconsistent estimator is one where either the estimates are biased or the covariance estimate does not well represent the underlying uncertainty distribution.

3. **Potential instability:** Feedback control typically assumes some level of state observability. There is no guarantee that driving an unobservable estimated state to a desired state will actually stabilize the system.
Current filter-based approaches to GPS-denied navigation directly estimate the global state of the vehicle. In this article we demonstrate for such systems that, because of unobservability, estimation and control performance can degrade significantly during periods of prolonged GPS dropout and heading uncertainty. We present as an alternative the relative navigation framework, which maintains full-state observability in spite of GPS dropout by estimating with respect to a local reference frame. While relative navigation, like all GPS-denied navigation approaches, is subject to global drift, it maintains a more accurate estimate of global uncertainty and so provides better inputs to techniques such as map optimization using loop closures that help to reduce this drift. Relative navigation facilitates consistent state estimation and stable control, thereby improving the overall safety and reliability of MAVs.

This article promotes a paradigm shift within the GPS-denied navigation literature. Many researchers are accustomed to working with respect to a global reference frame, and as a result concede that state observability is inevitably lost in the absence of global updates [30]. This article highlights the issues associated with such a concession and provides a viable alternative. In particular, we show that unobservability leads to a loss of estimator consistency. Inconsistency implies a loss of estimator optimality [2]. By subtly restructuring the problem, relative navigation avoids this and other pitfalls that are prevalent in GPS-denied navigation systems. In terms of implementation, the modifications that need to be made to an existing keyframe-based global filter implementation to convert it to relative navigation are relatively minor and straightforward.

We begin the article with a tutorial overview of current state-of-the-art, keyframe-based approaches for GPS-denied MAV navigation. We first summarize the most common global estimation approaches in a side-by-side comparison, highlighting the differences in how they incorporate the relative odometry measurements. We then introduce the relative navigation framework, explaining how it incorporates relative measurements to produce local state
estimates, and how it uses these local estimates to produce an estimate of the global pose of the MAV. With these preliminaries established, we then present the main contribution of this article, which is a rigorous analysis and comparison of the performance of each of these estimation approaches in simulation. We show that the relative navigation framework provides advantages in terms of accuracy, consistency, and its ability handle global updates after a prolonged GPS outage. To demonstrate that the findings in the simulation apply to real-world environments, we also present limited multirotor flight-test results for each of the approaches. We then conclude with some final discussion. It should be noted at the outset that the purpose of this article is to compare the performance of different theoretical estimation frameworks, rather than to present the specific details of a new estimator. While a background in Kalman filtering and statistical analysis will be useful to the reader, relevant concepts are reviewed as needed.

3.2 State-of-the-Art MAV Navigation

The majority of autonomous MAVs currently in operation use GPS with an inertial navigation system (GPS/INS) for state estimation and control. High-rate, body-fixed accelerometers and gyroscope measurements are integrated to estimate change in position and attitude. When properly calibrated, these measurements are remarkably accurate over small time steps and are commonly used as inputs to the estimator’s dynamics. Integrating noise, however, ultimately causes these estimates to drift. In general, GPS measurements do not drift with time, but are available at a slower rate. Commonly a probabilistic filter, such as an extended Kalman filter (EKF), fuses measurements from these two sensors using a model of the vehicle dynamics. When GPS is available and reliable, GPS/INS solutions work well for global missions requiring position hold or waypoint following. GPS/INS systems have been thoroughly researched and are widely used.
When GPS is unavailable, MAV navigation approaches typically utilize odometry computed from exteroceptive sensors such as cameras or laser scanners. Many such odometry algorithms exist for a variety of sensors, and include methods such as visual odometry [19,20] and laser scan matching [21,22].

Odometry can be computed either between consecutive frames (images or scans), or between the current frame and a keyframe. When a keyframe is used, a series of odometry measurements are computed with respect to this common, fixed reference frame. Typically, the keyframe is updated only when there is insufficient overlap to provide a reliable odometry measurement. As a result, keyframe-based odometry reduces temporal drift in the computed odometry as compared to frame-to-frame matching [23,24]. In this paper we focus on global keyframe filters, which build upon GPS/INS solutions but incorporate the relative pose measurements provided by these keyframe-based odometry approaches. Many of the concepts in this paper find parallels in the simultaneous localization and mapping (SLAM) literature. “Sidebar: Connection to Simultaneous Localization and Mapping” briefly summarizes the development of SLAM, including the recent emphasis on improving consistency through relative formulations and efforts to reduce computational complexity.

In the next section we briefly review the extended Kalman filter algorithm in a general sense, and establish the notation that is used throughout this article. We then describe the three global keyframe filters to be analyzed. While the formulation presented in this article is for a discrete-time system, the principles apply to continuous or continuous-discrete formulations as well.

### 3.2.1 Extended Kalman Filter

The extended Kalman filter (EKF) is an extension of the Kalman filter to systems with nonlinear dynamics and/or nonlinear measurement models. It recursively estimates the system state $\hat{x}_t$ as a function of the state estimate at the previous time step $\hat{x}_{t-1}$, inputs $u_t$, and
and measurements $z_t$. It also maintains an estimate of its uncertainty, represented by the covariance matrix $P_t$.

Consider the discrete-time system

$$x_t = f(x_{t-1}, u_t + v_t) + \xi_t,$$

where $v_t \sim \mathcal{N}(0, Q_u)$ and $\xi_t \sim \mathcal{N}(0, Q_x)$ are zero-mean Gaussian random variables. In the prediction step, the filter propagates the state estimates forward as

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_t).$$ (3.1)

The covariance is propagated according to the linear approximation

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + G_t Q_u G_t^T + Q_x,$$

where $F_t$ and $G_t$ are the Jacobians of the system dynamics with respect to the state and input respectively.

The update step is performed after the prediction step when a new measurement $z$ is available. In practice several prediction steps may be performed between updates. A measurement is modeled as

$$z_t = h(x_t) + \zeta_t,$$ (3.2)

where $\zeta_t \sim \mathcal{N}(0, R)$ is zero-mean Gaussian noise. The Kalman gain is computed as

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R)^{-1},$$ (3.3)
where $H_k$ is the Jacobian of the measurement model with respect to the state. The state and covariance are updated as

$$
\begin{align*}
    \mathbf{x}_{t|t} &= \mathbf{x}_{t|t-1} + K_t \left( \mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}) \right), \\
    \mathbf{P}_{t|t} &= \left( \mathbf{I} - K_t H_t \right) \mathbf{P}_{t|t-1} \left( \mathbf{I} - K_t H_t \right)^T + K_t R K_t^T,
\end{align*}
$$

(3.4)

using Joseph’s form to avoid numerical instability [2].

Several of the estimation approaches examined in this article also utilize a third step that we refer to as marginalization. When the odometry algorithm declares a new keyframe, portions of the state vector are reset. We express this operation as

$$
\hat{\mathbf{x}}^+ = \mathbf{m}(\hat{\mathbf{x}}).
$$

The covariance is also updated according to the linear approximation

$$
\mathbf{P}^+ = \mathbf{M} \mathbf{P} \mathbf{M}^T,
$$

where $\mathbf{M} = \partial \mathbf{m}/\partial \mathbf{x}$ is the Jacobian of the marginalization operation with respect to the state.

### 3.2.2 Global Estimation Approaches

The three types of global keyframe filters that we examine in this article are pseudo-global, stochastic cloning, and robocentric. These methods all utilize the EKF paradigm, but differ in the way they use the relative odometry measurements to update the global state estimate. In the following sections we describe the internal states, propagation and measurement models, and marginalization steps used by each filter. This information is also
Figure 3.2: Comparison of estimator architectures. Pseudo-global (PG) compounds the keyframe state and measurement to apply a global-like update. Stochastic cloning (SC) estimates both the global state and keyframe state and applies the update as the relative difference. Keyframe robocentric (kRC) reverses the state direction and uses a delta state for applying the prediction and update steps. Relative navigation (RN) only estimates the relative state. The coordinate frames labeled $g$, $k$, and $b$ correspond to the global, keyframe, and body frame of the robot.

Table 3.1: Estimation framework legend.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>Baseline (propagation only, no vision update)</td>
</tr>
<tr>
<td>PG</td>
<td>Pseudo-global</td>
</tr>
<tr>
<td>SC</td>
<td>Stochastic cloning</td>
</tr>
<tr>
<td>kRC</td>
<td>Keyframe robocentric</td>
</tr>
<tr>
<td>kRCi</td>
<td>Keyframe robocentric (inertial error)</td>
</tr>
<tr>
<td>RN</td>
<td>Relative navigation</td>
</tr>
<tr>
<td>bRN</td>
<td>Relative navigation (body-fixed dynamics)</td>
</tr>
</tbody>
</table>

presented graphically in Figure 3.2. The acronyms used throughout the article to refer to each approach are summarized in Table 3.1.

To simplify the presentation, it is assumed that the vehicle’s dynamics with respect to some arbitrary, inertial reference frame can be described by the function $f_x$, which is a specific instantiation of (3.1). The Jacobians of these dynamics with respect to the state and input noise is notated as $F_x$ and $G_x$ respectively.
An operation that we use with some frequency is compounding the poses represented by two states. Using notation similar to [34], we denote this operation using the \( \oplus \) operator as

\[
x = x^a \oplus x^b.
\]  
for frames \( a \) and \( b \). We also commonly wish to take the Jacobian of this operation for two poses \( \hat{x}^a \) and \( \hat{x}^b \). For compactness we define the Jacobians

\[
\begin{align*}
J_{1\oplus}|_{\hat{x}^a,\hat{x}^b} &= \left. \frac{\partial (x^a \oplus x^b)}{\partial x^a} \right|_{\hat{x}^a,\hat{x}^b}, \\
J_{2\oplus}|_{\hat{x}^a,\hat{x}^b} &= \left. \frac{\partial (x^a \oplus x^b)}{\partial x^b} \right|_{\hat{x}^a,\hat{x}^b}.
\end{align*}
\]

We also commonly reverse the direction of the first pose in the compounding operation. This reversal is denoted by the \( \ominus \) operator, so that in the expression

\[
x = \ominus x^a \oplus x^b, 
\]  

the reversal of \( x^a \) is compounded with \( x^b \). The Jacobian of the reversal operation for some pose \( \hat{x} \) is defined as

\[
J_{\ominus}|_{\hat{x}} = \left. \frac{\partial (\ominus x)}{\partial x} \right|_{\hat{x}}.
\]

With these definitions, it is then the case that

\[
\left. \frac{\partial (\ominus x^a \oplus x^b)}{\partial x^a} \right|_{\hat{x}^a,\hat{x}^b} = J_{1\ominus}|_{\hat{x}^a,\hat{x}^b} J_{\ominus}|_{\hat{x}^a},
\]

and

\[
\left. \frac{\partial (\ominus x^a \oplus x^b)}{\partial x^b} \right|_{\hat{x}^a,\hat{x}^b} = J_{2\ominus}|_{\hat{x}^a,\hat{x}^b}.
\]
As an example, for the 2D planar case where a pose is represented by the three-vector \( \mathbf{x}^i = [x_i, y_i, \phi_i]^T \), these operations and Jacobians are

\[
\begin{align*}
\mathbf{x}^a \oplus \mathbf{x}^b &= \begin{bmatrix}
x_a + x_b \cos \phi_a - y_b \sin \phi_a \\
y_a + x_b \sin \phi_a + y_b \cos \phi_a \\
\phi_a + \phi_b
\end{bmatrix}, \\
\ominus \mathbf{x}^a &= \begin{bmatrix}
x_a \cos \phi_a - y_a \sin \phi_a \\
x_a \sin \phi_a - y_a \cos \phi_a \\
-\phi_a
\end{bmatrix}, \\
J_{1\oplus |_{x^a,x^b}} &= \begin{bmatrix}
1 & 0 & -x_b \sin \phi_a - y_b \cos \phi_a \\
0 & 1 & x_b \cos \phi_a - y_b \sin \phi_a \\
0 & 0 & 1
\end{bmatrix}, \\
J_{2\ominus |_{x^a,x^b}} &= \begin{bmatrix}
\cos \phi_a & -\sin \phi_a & 0 \\
\sin \phi_a & \cos \phi_a & 0 \\
0 & 0 & 1
\end{bmatrix}, \\
J_{\ominus |_{x^a}} &= \begin{bmatrix}
-\cos \phi_a & -\sin \phi_a & x_a \sin \phi_a - y_a \cos \phi_a \\
\sin \phi_a & -\cos \phi_a & x_a \cos \phi_a + y_a \sin \phi_a \\
0 & 0 & -1
\end{bmatrix}.
\end{align*}
\]

3.2.2.1 Pseudo-Global

The pseudo-global (PG) approach, illustrated in Figure 3.2a, is perhaps the simplest and most intuitive of the methods, but is technically incorrect and has significant theoretical shortcomings. While it produces surprisingly accurate estimates, it becomes grossly over-confident because it ignores important cross-correlation terms. We include it here because several prominent research groups have used this approach for MAV navigation [21, 35].
In one embodiment of this approach, the vision sensor is fed into a monocular-SLAM framework, whose output is treated as a black-box global pose update that is fused directly in the filter [35, 36]. In other implementations [21, 37], the relative measurement is applied by appending it to a saved estimate of the keyframe’s global position \( \hat{x} \) and then treating it as a global measurement. In this case the estimated state is simply the global state of the vehicle,

\[
\hat{x}_{\text{PG}} = \hat{x}^g, \quad P_{\text{PG}} = P^g,
\]

and the propagation equations are the vehicle dynamics

\[
f_{\text{PG}} (\hat{x}_{\text{PG}}, u) = f_x (\hat{x}^g, u),
\]

\[
F_{\text{PG}} = F_x|_{\hat{x}^g}, \quad G_{\text{PG}} = G_x|_{\hat{x}^g}.
\]

The pseudo-global measurement is obtained by compounding the saved estimate of the keyframe state with the relative measurement obtained from the odometry as

\[
z_{\text{PG}} = \hat{x}^k \oplus z. \quad (3.7)
\]

An estimate of the measurement covariance is also needed. The most naive approach is to simply use the covariance for the relative odometry measurement. However, this method ignores uncertainty in the saved keyframe state \( \hat{x}^k \) used to construct the measurement. A slightly better approach is to use the covariance of (3.7), which is

\[
R_{\text{PG}} = \left( J_1|_{\hat{x}^k, z} \right) P^k \left( J_1|_{\hat{x}^k, z} \right)^T + \left( J_2|_{\hat{x}^k, z} \right) R \left( J_2|_{\hat{x}^k, z} \right)^T.
\]

When the measurement is constructed in this fashion, the predicted measurement is then simply the global state of the vehicle

\[
h_{\text{PG}} (\hat{x}_{\text{PG}}) = \hat{x}^g, \quad H_{\text{PG}} = I.
\]
While the modified measurement covariance described above incorporates uncertainty in the saved estimate of the keyframe state, it ignores cross-correlation terms between the keyframe state and the current global pose. In addition, treating a relative measurement as a direct update to the global state in this fashion causes the estimator to become grossly overconfident, as will be demonstrated in the results section of this article.

### 3.2.2.2 Stochastic Cloning

The stochastic cloning (SC) approach [29] improves on the pseudo-global approach by accounting for the uncertainty in the global pose of the keyframe as well as the cross-correlations with the vehicle’s global pose. In this approach, illustrated by Figure 3.2b, the state and covariance are augmented with the global pose of the keyframe $x^k$ as

$$
\hat{x}_{SC} = \begin{bmatrix} \hat{x}^g \\ \hat{x}^k \end{bmatrix}, \quad P_{SC} = \begin{bmatrix} P^g & P^{g,k} \\ P^{k,g} & P^k \end{bmatrix},
$$

where $P^g$ is the uncertainty in the global state, $P^k$ is the uncertainty in the keyframe state, and $P^{g,k}$ is the cross-correlation. During the prediction step, the keyframe state is not expected to change since there is no new information about it. The propagation equations therefore become

$$
f_{SC} (\hat{x}_{SC}, u) = \begin{bmatrix} f_x (\hat{x}^g, u) \\ \hat{x}^k \end{bmatrix},$$

$$
F_{SC} = \begin{bmatrix} F_{x|\hat{x}^g} & 0 \\ 0 & I \end{bmatrix}, \quad G_{SC} = \begin{bmatrix} G_{x|\hat{x}^g} \\ 0 \end{bmatrix}.
$$

The predicted measurement is obtained by compounding the inverse keyframe state with the global vehicle state to estimate the relative measurement from the keyframe to the body.
The measurement model is then

\[ h_{SC}(\hat{x}_{SC}) = \ominus \hat{x}^k \oplus \hat{x}^g, \]

\[ H_{SC}(\hat{x}_{SC}) = \begin{bmatrix} J_{2\oplus |\ominus \hat{x}^k, \hat{x}^g} & J_{1\oplus |\ominus \hat{x}^k, \hat{x}^g} \end{bmatrix}. \]

Each time the odometry algorithm declares a new keyframe, the estimator’s keyframe state must be updated. The old keyframe state is marginalized out, and the new keyframe state is initialized as the current estimate of the vehicle’s global state:

\[ m_{SC}(\hat{x}_{SC}) = \begin{bmatrix} \hat{x}^g \\ \hat{x}^g \end{bmatrix}, \]

\[ M_{SC}(\hat{x}_{SC}) = \begin{bmatrix} I & 0 \\ I & 0 \end{bmatrix}. \]

The stochastic cloning approach is much more theoretically sound than PG. However, as will be demonstrated in the results section of this article, it still suffers from estimator inconsistency. One reason for this is that the linearization about unobservable, drifted global states in the measurement update artificially introduces extra information to the filter [7]. While some methods for addressing this shortcoming have been proposed [38], we demonstrate that the fundamental, underlying issue of unobservability can be avoided entirely with relative navigation approach.

3.2.2.3 Keyframe-Robocentric

Robocentric mapping [14] is a more recent approach that addresses many of the consistency issues observed in EKF-SLAM approaches that express the vehicle and feature locations with respect to a fixed frame. Robocentric mapping improves consistency by ex-
pressing feature locations with respect to a body-fixed—or robocentric—frame, thus reducing
the linearization errors that lead to inconsistency.

To the authors’ knowledge, robocentric mapping has to date been applied exclusively to
the EKF-SLAM problem, where the locations of many features are tracked as states in the
filter. To facilitate comparison with the other approaches, we adapt it to a keyframe-based
approach by treating the keyframe as the only feature being tracked in the filter. We refer to
this adaptation as keyframe-robocentric (kRC) to differentiate from the existing literature.

The state vector for the kRC filter, illustrated in Figure 3.2c, consists of the location of
the global origin expressed in the body-fixed frame \( \hat{x}_g' \), the location of the keyframe expressed
in the body-fixed frame \( \hat{x}_k' \), and the displacement state \( \hat{x}^\Delta \):

\[
\hat{x}_{\text{kRC}} = \begin{bmatrix} \hat{x}_g' \\ \hat{x}_k' \\ \hat{x}^\Delta \end{bmatrix}, \quad P_{\text{kRC}} = \begin{bmatrix} P_g' & P_{g',k'} & P_{g',\Delta} \\ P_{k',g'} & P_{k'} & P_{k',\Delta} \\ P_{\Delta,g'} & P_{\Delta,k'} & P_{\Delta} \end{bmatrix}.
\]

The displacement state represents how the vehicle has moved since the last relative mea-
surement. It is computed by aggregating the changes produced by the prediction equations.
Only \( \hat{x}^\Delta \) is updated during the prediction step, so that

\[
f_{\text{kRC}} (\hat{x}_{\text{kRC}}, u) = \begin{bmatrix} \hat{x}_g' \\ \hat{x}_k' \\ f_x (\hat{x}^\Delta, u) \end{bmatrix},
\]

\[
F_{\text{kRC}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & F_x|_{\hat{x}_g^\Delta} \end{bmatrix}, \quad G_{\text{kRC}} = \begin{bmatrix} 0 \\ 0 \\ G_x|_{\hat{x}^\Delta} \end{bmatrix}.
\]
The measurement model consists of the inverse of the keyframe state composed with the current displacement state:

\[ h_{kRC}(\hat{x}_{kRC}) = \ominus \hat{x}^k \oplus \hat{x}^\Delta, \]

\[ H_{kRC} = \begin{bmatrix} 0 & J_{1\oplus}|_{\ominus \hat{x}^k', \hat{x}^\Delta} & J_{\ominus}|_{\hat{x}^k'} & J_{2\ominus}|_{\ominus \hat{x}^k', \hat{x}^\Delta} \end{bmatrix}. \]

After the update has been applied, the displacement state is compounded onto the global and keyframe states, then reset to zero. This is accomplished with a state marginalization of the form

\[ m_{kRC,z}(\hat{x}_{kRC}) = \begin{bmatrix} \ominus \hat{x}^\Delta \oplus \hat{x}^{g'} \\ \ominus \hat{x}^\Delta \oplus \hat{x}^k' \\ 0 \end{bmatrix}, \]

\[ M_{kRC,z} = \begin{bmatrix} J_{2\ominus}|_{\ominus \hat{x}^\Delta, \hat{x}^{g'}} & 0 & J_{1\oplus}|_{\ominus \hat{x}^k', \hat{x}^\Delta} \\ 0 & J_{2\ominus}|_{\ominus \hat{x}^\Delta, \hat{x}^{g'}} & J_{1\oplus}|_{\ominus \hat{x}^k', \hat{x}^\Delta} \\ 0 & 0 & 0 \end{bmatrix}. \]

The purpose of this delayed composition is to ensure that the Jacobians associated with the composition, \( M_{kRC,z} \), are evaluated only after correction by a measurement update to reduce linearization errors.

Another state augmentation and marginalization also must be performed when a new keyframe is declared. Because the location of the keyframe is expressed in the body frame, and because the body frame was exactly at the location of the keyframe at the time it was declared, this procedure consists of setting the keyframe state to zero with no uncertainty:

\[ m_{kRC,k}(\hat{x}_{kRC}) = \begin{bmatrix} \hat{x}^{g'} \\ 0 \\ \hat{x}^\Delta \end{bmatrix}, \]

35
\[
M_{kRC,k} = \begin{bmatrix}
I & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}.
\]

As a post-processing operation, it is also common to transform the kRC global state into an inertial frame, denoted kRC\textsubscript{i}, for purposes such as plotting and path planning. This is accomplished as

\[
\hat{x}_{kRC}^g = \ominus \hat{x}_{kRC}',
\]

\[
P_{kRCi} = J_{\ominus} \hat{X}_{kRC} P_{kRC}' J_{\ominus}^T \hat{X}_{kRC}.
\]

Note that \( P_{kRCi} \) is simply a rotation of \( P_{kRC} \).

### 3.3 Relative Navigation

The relative navigation (RN) approach addresses the unobservability of the global state by not estimating it in the filter. Instead, only the relative state of the vehicle with respect to the last keyframe is estimated, as illustrated in Figure 3.2d. The result is that the odometry provides a direct measurement of the state, making the state observable by construction. The global state can be subsequently reconstructed in a back-end map by composing together the series of relative pose estimates produced by the filter. This effectively removes the uncertainty in the global pose from the filter and hands it off to the back end, and as a result the uncertainty of the relative state in the filter remains bounded and consistent. In addition, the pose-graph representation of the global state in the back end creates a more accurate representation of the global uncertainty than can be obtained when representing the global state directly (see “Sidebar: The Banana Distribution”).
Figure 3.3: The vehicle-mounted sensors feed a probabilistic filter estimating the vehicle’s state relative to its local environment. Here “view-based odometry” refers to algorithms such as visual odometry and laser scan matching. Local path planning and control stabilize the vehicle in this local frame. Meanwhile, on a distinct thread, a time history of local information can be fused with any available global information to form a global map. The back end only influences the flight-critical front end in the form of global goals represented in the current relative coordinate frame.

Another key advantage of the RN approach is the loose coupling between the relative front end and the global back end, illustrated by the architecture shown in Figure 3.3. The intuition for this can be developed by imagining a human driver navigating through a city with respect to their local environment—traffic lanes and other vehicles—while opportunistically incorporating local guidance such as “take the next right turn” from a navigator in the passenger seat. Similarly, with the relative navigation approach, a vehicle is able to maintain stable flight and avoid collisions indefinitely, even when no global information is available, by using the front-end relative state estimates for local guidance and control. When back-end estimates of the global state change dramatically due to new information, this merely results in a new relative goal being passed to the front end, avoiding the large spikes in control effort seen by systems that control using the global state directly. The system is similarly insulated from delayed or degraded global information.

It is important to note that the general framework in Figure 3.3 is agnostic to a particular platform, sensor suite, or class of estimation filter. As such, the RN approach can be readily applied to existing systems. The following sections give high-level descriptions of
the relative front end and global back end. For further details on the RN approach and its implementation, the reader is referred to [28, 32, 39].

3.3.1 Relative Front End

The central component of the front end is a filter that estimates the local state of the vehicle. The filter state is the relative pose of the vehicle with respect to the last keyframe, as illustrated in Figure 3.2d:

\[ \hat{x}_{RN} = \hat{x}^r, \quad P_{RN} = P^r. \]

During the prediction step the relative state evolves according to the vehicle dynamics

\[ f_{RN}(\hat{x}_{RN}, u) = f_x(\hat{x}^r, u), \]

\[ F_{RN} = F_x|_{\hat{x}^r}, \quad G_{RN} = G_x|_{\hat{x}^r}. \]

The measurement model is simply the current relative state,

\[ h_{RN}(\hat{x}_{RN}) = \hat{x}^r. \]

The Jacobian of this measurement model is constant, eliminating linearization errors during the update step:

\[ H_{RN} = I. \]

When a new keyframe is declared, the filter first passes its current relative pose estimate and estimated covariance to the back end, then resets its relative state. Because we know that the vehicle was by definition at the location at which the keyframe is declared, the relative state can be reset identically to zero with zero uncertainty as

\[ m_{RN}(x_{RN}) = 0, \quad M_{RN} = 0. \]
Therefore at each new keyframe, the filter starts with zero pose uncertainty, helping to maintain filter consistency [40]. This concept parallels ideas found in the relative SLAM literature [15, 16].

In addition to the estimator, the front end is also responsible for the real-time control of the vehicle. All path planning and control are carried out in the current local frame of the estimator, using goals represented in that coordinate system that come from the back-end global planner. This separation of path planning and control from the global states insulates the system from the large shifts in global state that can occur when loop closure or GPS measurements are obtained and avoids spikes in control effort.

3.3.2 Global Back End

Each time the front-end estimator resets its state at a new keyframe, it first passes its current estimate of that state and the associated covariance to the back end. These relative pose estimates and covariances from the front-end filter comprise the edges of a pose-graph map. By compounding these edges, the global pose of the vehicle can be computed at any time [41].

Each edge in the graph has a Gaussian uncertainty associated with it, parameterized by the covariance matrix. Compounding these Gaussian uncertainties results in a banana-shaped distribution for the uncertainty in the vehicle’s global pose. “Sidebar: The Banana Distribution” explains that representing the vehicle’s global pose as a series of small transforms with Gaussian uncertainty results in a better representation of the global uncertainty than can be obtained by representing the global pose directly.

The pose graph map can also be used to incorporate additional information as it becomes available, such as loop closures or intermittent GPS measurements. These measurements are represented as additional edges in the pose graph, which over-constrains the graph.
Optimization methods then adjust the edges in the graph to reduce drift [42, 43]. Global mission planning is carried out using this optimized map.

For the purposes of this paper, these additional sources of global information will not be incorporated to facilitate comparison with the other estimation approaches. The global results for relative navigation presented in this paper are produced using a limited back end that simply compounds the relative poses and uncertainties provided by the front-end estimator.

### 3.4 Simulation

A comparison of each of the estimation approaches (PG, SC, kRC, and RN) was performed in simulation. Since each of these approaches has previously been demonstrated to work on real MAV platforms, the purpose of the simulation is to compare the theoretical performance of the approaches rather than to evaluate in isolation the suitability of any one approach for MAV navigation. To accomplish this comparison, a simple simulation environment was chosen so as not to obscure the underlying trends. Specifically, simple dynamics and measurement models were used, and process and measurement noise were drawn from known normal distributions. The authors postulate that if an approach breaks down theoretically in an idealized simulation environment, it will perform no better under the complications that arise on an actual MAV platform. The extension to MAV hardware is explored later in the article.

The simulated vehicle is an idealized ground robot following the standard unicycle model. The state of the vehicle is position and heading,

\[
x = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix}
\]
A random reference trajectory of duration $t_f$ is defined by generating forward velocities and angular rates at a rate of $f_u$ and applying them to the dynamics

$$\dot{x}_t = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} V_r^r \cos \phi \\ V_r^r \sin \phi \\ \omega_r^r \end{bmatrix},$$

where

$$V_r^r \sim \mathcal{N} \left( V_{\text{nominal}}, \eta_V^2 \right),$$

$$\omega_r^r \sim \mathcal{N} \left( \omega_{r-1}^r, \eta_\omega^2 \right).$$

Identical noisy inputs $u$ are provided to each estimator to simulate wheel odometry, where

$$u \triangleq \begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} V^r \\ \omega^r \end{bmatrix} + v, \quad v \sim \mathcal{N} \left( 0, Q_u \right),$$

with $Q_u = \text{diag}(\sigma_V^2, \sigma_\omega^2)$. The estimated vehicle dynamics over a time step $\Delta t$ are defined as

$$f_x (\hat{x}, u) = \begin{bmatrix} \hat{x} + V \Delta t \cos \hat{\phi} \\ \hat{y} + V \Delta t \sin \hat{\phi} \\ \hat{\phi} + \omega \Delta t \end{bmatrix},$$

with Jacobians

$$F_x = \begin{bmatrix} 1 & 0 & -V \Delta t \sin \hat{\phi} \\ 0 & 1 & V \Delta t \cos \hat{\phi} \\ 0 & 0 & 1 \end{bmatrix}, \quad G_x = \begin{bmatrix} \Delta t \cos \hat{\phi} & 0 \\ \Delta t \sin \hat{\phi} & 0 \\ 0 & \Delta t \end{bmatrix}.$$
Keyframes are established at a rate of $f_k$ as the vehicle moves through the environment. Relative measurements are then simulated at a rate of $f_z$ by differencing the keyframe’s true pose from the vehicle’s current true pose, and then applying Gaussian white noise with covariance $R = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_\phi^2)$.

The estimators use the true values of $Q_u$, $Q_x$, and $R$. In all, $N_1$ trajectories were generated, and for each trajectory $N_2$ realizations of $u$ and $z$ were simulated and provided to each of the estimators; these realizations are subsequently referred to as trials. The values used for this simulation are $N_1 = 32$, $N_2 = 1000$, $t_f = 600$ s, $f_u = 100$ Hz, $f_k = 1$ Hz, $f_z = 10$ Hz, $V_{\text{nominal}} = 1$ m/s, $\eta_V = 0.3$ m/s, $\eta_\omega = 0.5$ rad/s, $\sigma_V = 0.3$ m/s, $\sigma_\omega = 0.35$ rad/s, $\sigma_x = \sigma_y = 0.03$ m, and $\sigma_\phi = 0.052$ rad.

A baseline estimator (BL) is also established by propagating the input $u$ while ignoring the relative measurements $z$. While a propagation-only approach is typically not practical, it is useful for observing how the various measurement models influence accuracy and consistency.

### 3.5 Simulation Results

Figure 3.4 shows an example result from a single simulation trial for the pseudo-global (PG), stochastic cloning (SC), keyframe-robocentric (kRC/kRCi), and relative navigation (RN) approaches. In addition to the maximum-likelihood trajectory, the level curve corresponding to the 90% confidence bound for each estimation approach, derived from the estimator’s covariance estimate, is shown. For a given trial any of the estimation approaches could be the most accurate, but the shape and size of the level curves in this figure are representative of the trends observed across all trials.

The following sections present an analysis of the performance of each estimator based on the Monte-Carlo simulation results. The accuracy and consistency of each estimator
Figure 3.4: Example simulation for a single trial showing the trajectory estimated by each approach. 90 percent confidence bounds are drawn for each using the global uncertainty covariance. Only the banana-like distribution available with RN contained the true position. The uncertainty bounds for PG are too small to see. The inset image shows the jaggedness of the SC estimates.
are examined. In addition, the way in which each estimator handles a global measurement update, if one were to become available, is examined. These results are summarized in Table 3.2.

Many of the performance metrics deal with estimation error. Global error for each of the estimation approaches listed in Table 3.1 is defined as

\[
\begin{align*}
e_{BL} &= x - \hat{x}_{BLg}, \\
e_{PG} &= x - \hat{x}_{PGg}, \\
e_{SC} &= x - \hat{x}_{SCg}, \\
e_{kRC} &= \bigodot x - \hat{x}_{kRCg}, \\
e_{kRCI} &= x - \bigodot \hat{x}_{kRCIg}, \\
e_{RN} &= \log( \bigodot x \oplus \hat{x}_{RNg}),
\end{align*}
\]

where \( \hat{x}_{*g} \) is the global component of the estimated state. Error for a robocentric estimator can be computed either in the body frame (\( e_{kRC} \)), or in the inertial frame (\( e_{kRCI} \)). The former is relevant when control is computed directly from the body-fixed state estimates. The latter introduces heading error into the inertial position states, but is relevant whenever the estimated states are ultimately utilized in an inertial frame. Error for RN uses the matrix logarithm defined in “Sidebar: The Banana Distribution”.

It is important to note that when better input odometry is available, the differences in estimator performance is not as prevalent. The following simulation results should be interpreted in the context of identifying underlying issues and trends, and not a universal judgment on the quality or effectiveness of each estimation approach for a given scenario.
Table 3.2: Summary comparison of the estimation approaches. RN exhibits improved performance in terms of average position error, average absolute heading error, estimation bias, and average normalized estimation error squared (NEES). When a global measurement $z_g$ is eventually applied at $t_f$, the global estimation approaches either experience a large state jump or the measurement is rejected.

<table>
<thead>
<tr>
<th></th>
<th>BL</th>
<th>PG</th>
<th>SC</th>
<th>kRC</th>
<th>kRCi</th>
<th>RN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average position error at $t_f$ (m)</td>
<td>104.9</td>
<td>67.4</td>
<td>89.3</td>
<td>70.1</td>
<td>69.9</td>
<td>66.6</td>
</tr>
<tr>
<td>Average absolute heading error at $t_f$ (deg)</td>
<td>39.3</td>
<td>24.5</td>
<td>32.6</td>
<td>23.7</td>
<td>23.7</td>
<td>23.7</td>
</tr>
<tr>
<td>Number of biased trajectories at 99% level</td>
<td>31/32</td>
<td>31/32</td>
<td>31/32</td>
<td>4/32</td>
<td>32/32</td>
<td>0/32</td>
</tr>
<tr>
<td>Average NEES at $t_f$</td>
<td>10.2</td>
<td>3.52e7</td>
<td>175</td>
<td>7.77</td>
<td>7.89</td>
<td>3.09</td>
</tr>
<tr>
<td>Average position update given $z_g$ (m)</td>
<td>104.2</td>
<td>7.2e-4</td>
<td>70.3</td>
<td>69.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Average absolute heading update after $z_g$ (deg)</td>
<td>31.4</td>
<td>0.02</td>
<td>10.5</td>
<td>19.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

3.5.1 Accuracy

The accuracy of each estimator is evaluated by averaging the position and absolute heading errors over each trial of each trajectory. Figure 3.5 shows that for each estimator, as expected, the average error grows unbounded when global measurements are unavailable. However, each estimator loses accuracy at a different rate. By 600 seconds, each estimator has a statistically significant difference in average position and heading error given a 99% confidence interval, except kRC and kRCi position and kRC, kRCi, and RN heading. The final error is listed in Table 3.2. Whether these differences are practically important is application specific.

RN, kRC, and PG all have comparatively low error. In their own way, they each apply the relative measurement to a state with a bounded covariance estimate. In contrast, despite using the same relative measurements, SC performs notably worse than the other estimators. One potential explanation for this difference in accuracy is illustrated by the image inset in Figure 3.4, which shows the jagged nature of the SC trajectory. This jaggedness occurs because the update step for SC sometimes produces unnaturally large corrections, often much larger than the measurement innovation term itself. This phenomenon, first described in [7], occurs when heading uncertainty is allowed to grow unchecked.
Figure 3.5: Average position and absolute heading error over each trial on each trajectory. While the estimation error grows unbounded for each approach without global measurements, the error for RN grows at the slowest rate. kRC, kRCi, and RN have identical heading error.

3.5.2 Consistency

A consistent estimator is one whose estimates are unbiased and whose covariance estimate represents well the true underlying uncertainty distribution [2]. Each of these properties are explored in the following sections.

3.5.2.1 Estimator Bias

Figure 3.6 shows, for each estimator, the distribution of errors across all realizations of input and measurement noise for one of the generated trajectories. For all of the estimators except RN, the errors form a banana-like distribution. The expected values of $e_{BL}$, $e_{PG}$, $e_{SC}$, and $e_{kRCi}$, marked with crosses, are non-zero, indicating a bias that is a direct result of the banana-like distribution. The distribution of $e_{RN}$ in Figure 3.6f, on the other hand, is not banana shaped because the matrix logarithm unwarps the distribution. As a result, the expected value of $e_{RN}$ is approximately zero. As an addendum to the discussion in “Sidebar: The Banana Distribution”, the fact that Figure 3.6f appears approximately elliptical suggests
Figure 3.6: Estimation error for one of the trajectories. The cross indicates the mean error. BL, PG, SC, and kRCi are biased, while kRC and RN are not. Units are in meters.
that a Gaussian covariance parameterization in exponential coordinates is appropriate for
the RN pose graph representation.

To investigate bias more precisely, the Hotelling $T^2$ statistic is computed for each esti-
mator and each trajectory. The statistic is defined as

$$T^2 = N_2 \bar{e}^T S^{-1} \bar{e},$$

where $\bar{e}$ and $S$ are the sample mean and covariance, and is distributed according to

$$T^2 \sim p\frac{(N_2 - 1)}{N_2 - p} F_{p, N_2 - p},$$

where $p = 3$ is the dimensionality of the state vector and $F$ is the $F$-distribution with
the indicated degrees of freedom. Table 3.2 indicates the number of trajectories for each
estimator that are statistically biased at a 99 percent confidence level. RN is the only
estimation approach that can be considered unbiased for every trajectory at this level. kRC
is usually not biased while working in the body-centered frame, but is always biased when
represented in an inertial frame (kRCi).

### 3.5.2.2 Estimator Uncertainty

Having an accurate covariance estimate is important for proper sensor fusion [2]. In
addition, many higher-level algorithms, such as path planning, rely on accurate uncertainty
estimates.

Figure 3.4 shows the typical sizes and shapes of the uncertainty estimates for each
approach. The covariance of the PG approach remains on the same order as the relative
measurement uncertainty $R$, which is too small to be seen in Figure 3.4. The covariance
of the SC approach does not grow sufficiently because the measurement model artificially
introduces extra information. Additional steps can reduce this effect [38], but the root cause—unobservability—is avoided altogether by the RN approach. Because kRC predominately works in an observable, local frame, the uncertainty estimate grows at the appropriate rate [14], but like PG, SC, and BL the covariance representation only supports an elliptical confidence bound. The banana-like confidence bound of RN properly represents the true uncertainty distribution.

The degree to which the underlying uncertainty is well-modeled is evaluated using the normalized estimation error squared (NEES), defined at each time step $t$ as

$$
\epsilon_t = e_t^T P_t^{-1} e_t .
$$

This metric, commonly referred to as the Mahalanobis distance, weights the deviations in each state based on the inverse of the associated uncertainty. When $P$ is the true covariance, NEES has the property

$$
\epsilon \sim \chi_p^2 ; \quad E[\epsilon] = p ,
$$

where $p = 3$ is dimensionality of the state vector.

Figure 3.7 presents the observed probability distribution function (pdf) of $\epsilon$ for each approach. $\epsilon$ was calculated once per second over each trial of each trajectory. The ideal $\chi_3^2$ pdf is overlaid and the expected value is labeled. When the observed distribution is more heavily weighted on the left than the ideal $\chi_3^2$ distribution, the filter is said to be conservative, indicating the covariance is too large. Conversely, when the observed distribution has a longer or fatter tail to the right than the ideal, the filter is optimistic [7] or overconfident. It is usually better for an estimator to be conservative.

From Figure 3.7 we see RN appears to be globally consistent in spite of prolonged GPS-dropout and substantial heading uncertainty. PG does not provide an accurate global uncer-
Figure 3.7: Normalized estimation error squared (NEES) probability density function (pdf) for each estimation approach with the ideal $\chi^2_3$ pdf overlaid. $\epsilon$ is calculated once per second over each trial of each trajectory. RN appears globally consistent while the other approaches are overconfident. The distribution mean is labeled, where the ideal mean is 3.
Figure 3.8: Average normalized estimation error squared (NEES) over all trials and all trajectories as a function of time. The expected value for a consistent estimator should be 3. RN begins and stays near the ideal while the other approaches lose consistency from the start.

tainty estimate where the smallest $\epsilon_{\text{PG}}$ is 320. Only occasionally does SC provide a reasonable global covariance estimate given these circumstances. kRC, kRCi, and BL follow the general trend but experience an abnormally high number of large NEES estimates. For example, given the particularly erroneous inputs illustrated in Figure 3.4, $\epsilon_{\text{BL}} = 279$, $\epsilon_{\text{PG}} = 2.7 \times 10^8$, $\epsilon_{\text{SC}} = 1275$, $\epsilon_{\text{kRC}} = 39.8$, $\epsilon_{\text{kRCi}} = 67.9$, and $\epsilon_{\text{RN}} = 6.03$ at time $t_f$.

Because we are exploring how estimation approaches break down, it is insightful to see how the average NEES changes over time. In Figure 3.8 we see RN begins and stays approximately equal to the ideal value of three. PG almost immediately loses consistency. Interestingly, the consistency of BL, SC, kRC, and kRCi all degrade at the same rate for the first 50 seconds. At this point, SC continues to degrade while the others approximately settle out. Plots similar to Figure 3.8 are common in the SLAM literature, though typically only 20-50 trials are averaged [6, 7, 44]. They similarly show that without global information such as loop closures consistency is lost, even when using an iterative EKF, unscented KF, or ideal Jacobians evaluated at the true unknown state. RN avoids this issue completely by working in an observable frame.
3.5.3 Eventual Global Update

If a global measurement such as GPS becomes available, it is useful to incorporate that information into the state estimate. After a prolonged period of GPS dropout, the discrepancy between the true and estimated global position is likely to be large due to drift in the estimates. When an estimator’s uncertainty is also large it will readily accept the measurement, causing a large state jump that could potentially produce a large spike in control effort. This is particularly troubling when GPS is degraded; even when the global information is accurate, however, directly fusing the information using a Kalman filter update is known to degrade both the estimation and control [23, 45]. The extent to which each approach is affected by a delayed global update is examined in the following extension to the simulation.

For each trial a global position measurement \( z_g \) is generated, where \( z_g \sim \mathcal{N}(x_{tf}, R_{z_g}) \) and \( R_{z_g} = \text{diag}(25, 25) \). Using (3.2)–(3.4), the state update is computed as

\[
\Delta = K(z_g - h(\hat{x})) .
\]

While the measurement only includes position information, the heading estimate is updated through the cross-correlation covariance terms. Table 3.2 lists the average magnitude of the position and heading updates. It is informative to note the magnitude of the average update as well as its relative size to the average error. From (3.3) it is clear that if \( P \) is substantially smaller than \( R \), the Kalman gain approaches zero and the measurement innovation is largely rejected, as demonstrated with PG. Conversely, when \( P \) is substantially larger than \( R \), the Kalman gain approaches identity and the measurement innovation is readily accepted, causing a large state jump as demonstrated by BL and kRC. SC only partially accepts the global update. None of these scenarios are conducive to robust MAV navigation and control.

Several approaches have been presented to work around these issues, such as simultaneously tracking a GPS-corrected and odometry-only global trajectory [23, 45], or using a
series of measurement gates [46]. Other approaches refrain from incorporating GPS into the filter at all, opting instead to incorporate GPS exclusively using a pose graph [47].

Because RN only estimates a relative state, directly fusing an eventual global measurement is not an option. Rather, these measurements are incorporated using a pose-graph map as described in [32]. This formulation completely avoids the problem of large jumps in the filter states that are used for control. In addition, it makes it possible to identify erroneous measurements and completely eliminate their effect on the global state estimate at any time as more information is received.

3.5.4 Summary of Implications

Implications of the simulation results are summarized in Figures 3.9 and 3.10. Rather than showing data flow, these diagrams illustrate consequences, where implications are represented as connecting arrows. The blue boxes, comprising prolonged GPS-dropout, heading uncertainty, intermittent and erroneous global measurements, as well as loop-closure constraints, represent the non-ideal yet realistic inputs to the system. Green boxes highlight the deficiencies that inevitably result.

The simulation results are summarized by discussing paths through Figure 3.9, proceeding from top to bottom and from left to right. During prolonged GPS dropout and in the presence of heading uncertainty, the global position and heading state will inevitably drift. This drift will induce an underlying probability distribution that is banana-shaped, which is not well modeled by a Gaussian in Cartesian coordinates. Further, some approaches tend to introduce undue information into the filter, artificially constraining the uncertainty magnitude. Eventually, this leads to inconsistent uncertainty estimates, causing measurements to be fused sub-optimally. When filters properly acknowledge that the state is unobservable, the uncertainty will grow without bound. A large uncertainty also causes the system to strongly trust eventual global measurements, whether they be degraded or not, often
Figure 3.9: Diagram summarizing the implications that prolonged GPS dropout and heading noise have when working in a global coordinate frame. Arrows indicate implications, blue boxes highlight non-ideal yet realistic inputs to the system, and green boxes highlight inevitable deficiencies. Ultimately global drift will induce an inconsistent and/or unbounded state uncertainty estimate $P$ which leads to non-robust navigation. For each approach, the relevant consequences are labeled according to the legend.
Figure 3.10: Diagram summarizing the implications that prolonged GPS dropout and heading noise has when global and relative frames are decoupled (i.e. relative navigation). Note that unlike the global estimation techniques described in Figure 3.9, the non-ideal inputs (blue) and inevitable deficiencies (green) are specific to the global frame and do not affect the robustness of relative navigation.
leading to large state updates. Sudden jumps in state lead to degraded position control, impacting MAV reliability. Finally, many approaches attempt to reduce drift by identifying when the vehicle returns to a previously visited location. Incorporating loop closure constraints through non-linear optimization techniques also results in large state updates and either presents non-negligible delays, stresses the platform’s size, weight and power (SWaP) constraints, or assumes an uninterrupted network connection.

A wide variety of techniques are used in practice to address the issues presented in Figure 3.9, particularly for handling large state updates. It should be noted, however, that the root of each issue stems from working with respect to an unobservable coordinate frame. Figure 3.10 illustrates how decoupling the system into relative and global frames allows for optimal navigation within the local frame itself, avoiding many of these issues entirely. Because relative measurements directly update the relative state, the state covariance represents the underlying uncertainty well, leading to optimal sensor fusion. While the non-ideal inputs and inevitable deficiencies may hamper the completion of a global mission, they need not degrade the stability of the system. For example, in the worst case, a large, erroneous global update results in an incorrect relative goal.

3.6 Hardware Results

The simulation results constitute the primary contributions of this article, showing that global keyframe filters break down more readily than relative navigation during prolonged GPS dropout. To supplement these contributions, this section presents the estimation performance of each method when implemented on MAV hardware. This section is not intended to thoroughly describe all MAV implementation details, but rather to demonstrate that the performance of full-state estimators running on actual hardware parallels the results seen in simulation. Complete details on the relative navigation implementation used for these results, along with a presentation of more extensive flight-test results, are given in [32].
Figure 3.11: Estimation results from a multirotor flight. 90 percent confidence bounds are drawn. A body-fixed relative navigation (bRN) approach was also included. While kRCi had the lowest error, the overall accuracy of an approach should only be evaluated after many trials, as done in Figure 3.5. However, the confidence bounds are typical; only RN and bRN include the true state. BL quickly degraded due to significant IMU noise and was excluded from the figure. (Background image courtesy of U.S. Geological Survey)

Reference [39] includes the details necessary to implement a relative state estimator for a MAV, including the vehicle dynamics and measurement models. The estimator implementations of each approach for the results in this article are all based on the error-state, multiplicative extended Kalman filter described in [39]. The changes needed to adapt this filter to each of the different estimation approaches were minimal, requiring modifications to less than ten lines of code for each approach. The vehicle’s state includes position, velocity, attitude parameterized with a quaternion, and gyroscope and accelerometer biases. Inputs were body-fixed accelerometer and gyroscope measurements at 100 Hz from a calibrated MicroStrain 3DM-GX3-15 IMU. A MaxBotix MB1242 ultrasonic altimeter provided height-above-ground measurements, while visual odometry updates were obtained at 15 Hz using the algorithm from [48] with an ASUS Xtion Pro Live RGB-D camera.

A user provided velocity commands to navigate the multirotor around the perimeter of the building shown in Figure 3.11. The flight lasted nine minutes and traversed 320 meters. The RN estimator provided the necessary onboard state estimates for feedback control.
Afterwards, time-stamped measurements were provided to each estimator resulting in the trajectory and covariance estimates shown in Figure 3.11.

As shown in Figure 3.11, the flight-test produced similar results to those seen in simulation. While a single trial cannot be used to make claims about the relative accuracy of the different approaches, all of the approaches produced reasonably accurate results with errors on a similar order of magnitude. Like in the simulation results, PG produced an extremely overconfident uncertainty bound, while SC produced a somewhat better, but still overconfident, uncertainty estimate. The RN approach produced a banana-shaped uncertainty bound that includes the true pose of the vehicle.

Because conventional multirotor dynamics assume an inertial reference frame, the robocentric displacement vector $\hat{x}^\Delta$ in Figure 3.2c cannot be propagated directly. Instead, following [26] kRC was implemented using vehicle dynamics expressed with respect to the body, also described in [39]. Because these position dynamics do not depend on the current attitude, the EKF has no mechanism to properly increase position uncertainty due to heading uncertainty. As a result, the kRC confidence bound remained small.

To provide a more direct comparison to the body-fixed implementation of kRC, relative navigation was additionally implemented using the body-fixed dynamics (bRN) presented in [39]. In contrast to kRC, the coupling between position and heading uncertainty at the keyframe level is negligible. As a result, when the global state and uncertainty are reconstructed using a sequence of many relative states, as described in “Sidebar: The Banana Distribution”, a reasonable banana-shaped confidence bound results. A more thorough investigation into the consistency of bRN remains as future work.
3.7 Conclusions

Global drift is inevitable when MAVs fly in GPS-denied environments because the global position and heading states are unobservable when only relative measurements are available. While most navigation frameworks estimate the global state directly despite this unobservability, relative navigation maintains local observability by estimating the vehicle states with respect to a local frame. As demonstrated in this article, estimating and controlling with respect to a local frame produces more consistent global position estimates and avoids many of the consistency and stability issues common to existing global approaches. Relative navigation also provides a framework for robustly incorporating intermittent global information. Moving from a global to a relative estimation approach is a fairly small adjustment conceptually and in terms of implementation, but yields significant advantages that can benefit systems that currently use a global estimation approach.

3.8 Sidebar: Connection to Simultaneous Localization and Mapping

When a vehicle enters an unknown, GPS-denied environment, it must estimate both its state and the locally observed environment. The simultaneous localization and mapping (SLAM) problem solves for the most probable vehicle trajectory and landmark locations, given a time history of inputs and relative measurements to the local environment [12, 13]. Often, simplifying assumptions are made to reduce the computational burden on resource-constrained platforms. While full-SLAM solves for the most likely vehicle trajectory, online-SLAM, such as EKF-SLAM [17], solves only for the current state of the vehicle and nearby features or landmarks [18]. Keyframe filters further simplify the estimation problem by tracking a single keyframe rather than many individual landmarks.

MAV state estimators are generally limited to solving some simplified form of the SLAM problem due to size, weight, and power (SWaP) limitations specific to each platform. Some MAV navigation solutions are derived from the EKF-SLAM framework [25, 26], while others
avoid filtering techniques altogether, electing to solve for the MAV’s pose using non-linear optimization techniques such as factor graphs [27]. Yet many others choose to use keyframe-based approaches for their reduced computational complexity [21, 23, 24, 28, 29]. Each approach works towards solving similar problems but with different assumptions and different formulations.

Another prominent research emphasis for SLAM is maintaining consistency. While traditional SLAM approaches estimate the vehicle and landmark locations with respect to a global coordinate frame, it has been shown that such parameterizations lose consistency as heading uncertainty increases [7]. As such, an increasing number of relative SLAM implementations are being published. Robocentric approaches estimate the pose of landmarks and the global origin with respect to the vehicle’s current position and attitude [14]. Relative submaps estimate the state of the vehicle and landmarks with respect to a local inertial coordinate frame [15,16]. These submaps are subsequently fused and form a more consistent global estimate.

In summary, keyframe-based filters are common for MAV state estimation because of the reduced computational burden, while relative formulations are common within the SLAM literature to improve consistency. This article presents a relative keyframe filter, which we call relative navigation, as the logical method to improve consistency of global keyframe approaches. The purpose of this article is not to compare the performance of keyframe-based MAV navigation approaches to other SLAM techniques, but rather to rigorously compare relative navigation to state-of-the-art global keyframe filters. These ideas are illustrated in Figure 3.12.

3.9 Sidebar: The Banana Distribution

The “banana distribution,” first described in [49], is the sickle-shaped distribution of position uncertainty that arises due to heading uncertainty in robot localization problems.
Figure 3.12: A relative keyframe filter, such as relative navigation, builds upon ideas presented in the SLAM literature to ensure reduced computational complexity and improved consistency. This article compares the relative navigation framework to popular global keyframe filters.

Intuition about why the banana distribution occurs can be developed by considering the arc-like distribution that arises for a robot that travels in a straight line for a known distance, but with an uncertain initial heading or growing heading uncertainty.

A similar distribution arises in the pose-graph representation of the relative navigation approach, where several short transforms are concatenated as in the example in Figure 3.13. Because the length of each individual transform is small and the heading uncertainty is low, the uncertainties on the transforms are well approximated by Gaussian normal distributions. However, when several transforms are concatenated, the resulting global uncertainty becomes distinctly banana shaped. This is illustrated in the distribution shown in Figure 3.13, created by sampling from the individual transform uncertainties in a Monte-Carlo fashion.

The most common parameterization of uncertainty, inherent to the Kalman filter and its variants, is a Gaussian normal distribution. For filters that estimate the global state directly, the covariance is computed directly in a Cartesian coordinate system. However, this parameterization, illustrated with ellipses in Figure 3.13, results in a poor fit for the true underlying distribution.
Figure 3.13: Banana-shaped distribution arising from the concatenation of small, uncertain transforms. The black line shows the nominal path created by concatenating several small transforms with Gaussian uncertainties shown by the purple ellipses. The gray lines and dots show a Monte-Carlo sampling from the transform distributions. The level contours of a Gaussian distribution are shown parameterized with Cartesian coordinates in red, and parameterized with exponential coordinates in blue.

The distribution captured by the pose-graph representation can be better parameterized using a Gaussian normal distribution in exponential coordinates [50]. While methods exist for approximating the final distribution directly from the covariance matrices of the concatenated transforms [40], in this sidebar we discuss computing an approximate distribution from the sample covariance of the Monte-Carlo points, used in [50] and as the baseline method in [40]. To accomplish this, we need to lay some theoretical groundwork.

Rigid body rotations are represented by members of the special orthogonal group,

\[ SO(2) \triangleq \left\{ C \in \mathbb{R}^{2 \times 2} \mid C^T = I, \det C = 1 \right\}, \]
for planar motion, or similarly \( SO(3) \) for six-degree-of-freedom (6DOF) motion. The special Euclidean group,

\[
SE(2) \triangleq \left\{ T = \begin{bmatrix} C & r \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \mid \{ C, r \} \in SO(2) \times \mathbb{R}^2 \right\},
\]

(or \( SE(3) \) for 6DOF) represents transformations parameterized by a translation \( r \) and rotation \( C \). The pose compounding operations of equations (3.5) and (3.6) are equivalent to representing the poses as transforms in \( SE(2) \) and multiplying them (or their inverses) as appropriate.

\( SE(2) \) is an example of a \textit{Lie group}, and associated with that group is the \textit{Lie algebra} \( \mathfrak{se}(2) \). The Lie algebra is the tangent space to the Lie group around the identity element, and unlike the group is a vector space. Members of the Lie algebra can be mapped to elements of the group via the exponential mapping, while the matrix logarithm provides the inverse mapping.

Expressing a difference in pose in exponential coordinates is equivalent to mapping the error transform into the Lie algebra. For each Monte-Carlo point \( x_i \) in Figure 3.13, we therefore express the distance from the true pose \( x \) in exponential coordinates as

\[
\delta \xi_i = \log (\oplus x \oplus x_i) ,
\]

where for \( SE(2) \)

\[
\log (x) = \begin{bmatrix} \phi \\
\cot \frac{\phi}{2} \\ -1 \cot \frac{\phi}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.
\]

Level curves of the resulting distribution are shown by the blue lines in Figure 3.13, and are a much better fit for the distribution represented by the pose graph. This is numerically
verified using a log-likelihood ratio test in [50]. Unlike the native pose-graph representation, however, the distribution is parameterized entirely by the mean and covariance, and so additional statistics can easily be computed. This idea is used throughout the article to define the global uncertainty estimate, determine bias, and compute NEES estimates for the relative navigation approach.
Chapter 4

Relative Multiplicative Extended Kalman Filter for Observable GPS-Denied Navigation

4.1 Introduction

GPS-denied navigation for small unmanned aircraft systems (UAS) is an active and rich field of research with significant practical applications such as infrastructure inspection and security. Most UAS fuse GPS with accelerometer and gyro data to provide accurate global state estimates suitable for feedback control. When GPS is not available, however, additional sensors such as cameras or lidars are required. Because of the size, weight, and power constraints and fast vehicle dynamics associated with small UAS, many such systems incorporate these additional sensors using filter-based estimation techniques rather than traditional full simultaneous localization and mapping (SLAM) algorithms. Filter-based approaches are computationally efficient and ensure smooth, timely state estimates for control.

In the absence of GPS updates, many filtering methods utilize incremental odometry measurements from either visual odometry or laser scan matching. These odometry measurements can be computed frame-to-frame, or several measurements can be computed with respect to the same keyframe image or scan. The keyframe image or scan is updated when there is insufficient overlap with current images or scans to compute reliable odometry measurements. Keyframe-based approaches have the advantage of reducing temporal drift in the odometry measurements [24].

\footnote{This paper was written by Daniel P. Koch, David O. Wheeler, Randal W. Beard, Timothy W. McLain, and Kevin M. Brink, and is available at [39].}
Figure 4.1: Block diagram of the general relative navigation system architecture. Flight critical estimation and control is performed with respect to a local frame. The framework is described in more detail in [32].

Despite having only incremental measurements available, the majority of GPS-denied navigation approaches directly estimate the vehicle’s global pose with respect to some fixed origin. Without global position measurements, however, the vehicle’s global pose and heading are unobservable [5,30,31]. As a result, these global filters often suffer from inconsistency and performance issues [8,13,51,52]. In contrast, the relative navigation approach estimates only the relative state of the vehicle with respect to the location of the most recent odometry keyframe [28]. As a result, the odometry provides direct measurements of the position and heading states, making them observable by construction. Each time a new keyframe is declared, the current state and covariance estimates are passed to a back-end map that concatenates them as edges in a pose graph to reconstruct the global path of the vehicle. The position and heading states in the filter are then reset to zero and estimation continues. The relative navigation architecture is illustrated graphically in Figure 4.1, and parallels ideas found in the SLAM literature [15,16]. Simulation results show that using the relative navigation framework to ensure observability provides significant advantages in terms of consistency of the estimated relative and global states, as well some improvement in accuracy [8]. Multirotor hardware flight test results also demonstrate the practicality of relative navigation for small UAS [32].
The core component of the relative navigation approach is the relative state-estimation filter. While a variety of filter types could be utilized, the flight tests presented in Ref. [32] successfully leveraged the relative multiplicative extended Kalman filter (RMEKF) presented in this paper. While Ref. [32] demonstrated the effectiveness of the relative navigation framework and gave an overview of the various components, it did not describe the RMEKF in detail. The RMEKF builds upon the multiplicative Kalman filter (MEKF), which uses a quaternion to represent attitude and quaternion multiplication to define attitude error. The RMEKF extends the MEKF by defining the UAS state to be with respect to a local coordinate frame associated with the current keyframe image. To accommodate this relative state, the RMEKF introduces an additional keyframe reset step that is applied each time a new keyframe is declared.

This paper contributes to the literature in three ways. First, the paper provides a tutorial derivation of the MEKF for UAS state estimation given Hamilton quaternions. Second, the paper presents a complete derivation of the RMEKF for a multirotor UAS, including several important extensions to the original presentation of [33]. Third, the paper provides a thorough derivation of the RMEKF for both inertial and body-fixed (robocentric) state representations, highlighting the subtle but important differences that exist between the two methods. The following paragraphs describe these contributions in further detail and relate how they compare to the existing literature.

MEKF Tutorial. Significant portions of the paper are tutorial in nature, clearly motivating why an indirect or error state formulation is necessary when quaternions are used to represent attitude, and providing complete explanations of each step in the derivation of the filter equations. The MEKF was first introduced in [53], and several in-depth discussions and derivations of MEKF implementations have been published [54–56]. While these publications are extremely useful and several are of similar scope to the current work, this paper provides several meaningful extensions. First, this paper derives an estimator for the full state of a
UAS (position, velocity, attitude, accelerometer biases, and gyroscope biases), while most previous MEKF papers of similar scope discuss only the attitude and bias estimation. Second, this paper derives the MEKF using the Hamilton quaternion convention as opposed to the JPL convention used in some other works. While the choice of quaternion convention does not fundamentally change the problem, Hamilton quaternions are commonly used in the robotics literature and subtle but important differences arise. This paper provides a contrasting perspective to help deepen understanding of quaternions. Third, the tutorial nature of this paper provides sufficient context for the derivation of several new properties relating to quaternions, their error representations, and their Euler-angle decomposition. These properties play a key role in the derivation of the RMEKF to allow partial attitude updates.

**RMEKF Derivation.** Another purpose of this paper is to provide a thorough derivation of the RMEKF estimator successfully used in [32] for prolonged UAS navigation in GPS-degraded environments. The RMEKF presented in this paper extends the original RMEKF derivation in [33] in several important ways that have proven necessary for prolonged flight. First, this paper presents a new visual odometry measurement model and keyframe reset operation, which together ensure the state remains observable in GPS-denied environments. Second, several novel properties of error quaternions are derived that enable partial updates to quaternion states and their covariances. Third, new terms are added to the state vector in this paper to correctly account for uncertainty in the roll, pitch, and altitude of the vehicle at the time a keyframe is declared. Finally, smaller differences include reversing the direction of the odometry measurement model to avoid unnecessarily coupling heading uncertainty into the update, and estimating the global height of the vehicle above ground rather than treating altitude as a relative state.
Inertial and Body-Fixed Dynamics. Another unique contribution of this paper is the derivation of the RMEKF when the state is defined with respect to either an inertial frame or a body-fixed frame. Using an inertially-fixed, gravity-aligned frame is the approach given in [28,33], and is the more traditional way of expressing the vehicle dynamics. In recent years, however, robocentric approaches, such as those in [14,26], have become popular as a method for addressing some of the inconsistency issues of traditional EKF-SLAM approaches. It is demonstrated in [8] that relative navigation obtains these same benefits without the need to invert the vehicle dynamics, but also that either robocentric or inertial dynamics can be used within the relative navigation framework. Another contribution of this paper is a presentation of the subtle differences that arise between using an inertial and body-fixed reference frame. For example, in addition to the change in dynamics, subtle changes appear in the quaternion integration, error state definition, measurement models, and keyframe reset operations. By presenting both formulations side-by-side, these differences are clearly outlined.

The final contribution of this paper is a complete, self-contained derivation of the filter and all relevant quaternion properties. The definitions of quaternions and error states used across the current estimation literature differ in subtle ways. When these definitions are not thoroughly documented, it becomes difficult to correctly leverage properties from multiple sources. With its tutorial nature and step-by-step explanations, this paper is designed to present a complete, self-contained derivation with respect to a consistent, explicitly stated definition. This allows the reader to understand, implement, and potentially modify the RMEKF for new vehicles or applications. Note that while the keyframe reset step and several measurement models are specific to relative navigation, the propagation equations and general filter structure are equally relevant for other applications, such as GPS/INS navigation.
Section 4.2 summarizes the notation used throughout the paper. Section 4.3 provides an overview of the quaternion definitions used in this paper. Specifically, Section 4.3.8 derives several relevant, new properties of error quaternions. Section 4.4 outlines the structure of the MEKF. The keyframe reset step is described in Section 4.5, and an overview of the complete RMEKF algorithm is given. Section 4.6 derives the specific filter equations for inertial relative navigation (iRN), and Section 4.7 derives the equations for body-fixed relative navigation (bRN). Finally, Sections 4.8 and 4.9 present simulation results and conclusions respectively.

4.2 Nomenclature

The following variables, operators, and notation are defined and motivated throughout the paper and are summarized here for convenience. Let \( B \) denote the vehicle’s body frame and \( I \) denote an inertial frame.

**State variables**
- \( \mathbf{x} \quad \text{state} \)
- \( \mathbf{x}_v \quad \text{vector component of state} \)
- \( \mathbf{x}_q \quad \text{quaternion component of state} \)
- \( c_p^a \quad \text{position of } b \text{ with respect to } a, \text{ expressed in } c \)
- \( p^a_b \quad \text{position of } b \text{ with respect to } a, \text{ expressed in } a \)
- \( q^a_b \quad \text{quaternion that rotates from } a \text{ to } b \)
- \( \mathbf{v} \quad \text{velocity of } B \text{ with respect to } I, \text{ expressed in } B \)
  \( (\mathbf{v} \triangleq \mathcal{B}\mathbf{v}_I) \)
- \( \mathbf{\omega} \quad \text{angular velocity of } B \text{ with respect to } I, \text{ expressed in } B \)
  \( (\mathbf{\omega} \triangleq \mathcal{B}\mathbf{\omega}_I) \)
- \( \mathbf{a} \quad \text{acceleration of } B \text{ with respect to } I, \text{ expressed in } B \)
  \( (\mathbf{a} \triangleq \mathcal{B}\mathbf{a}_I) \)
- \( \beta_\omega \quad \text{rate gyro biases, expressed in } B \)
- \( \beta_a \quad \text{accelerometer biases, expressed in } B \)
- \( \mu \quad \text{linear drag coefficient} \)
- \( \eta \quad \text{zero-mean Gaussian process noise} \)
- \( \nu \quad \text{zero-mean Gaussian input noise} \)
Error state variables
\( \delta x \) error state
\( \delta x_v \) vector component of error state
\( \delta x_\theta \) attitude component of error state
\( \delta p \) position error state
\( \delta q \) quaternion error state
\( \delta \theta \) attitude error state (minimal representation)

Filter variables
\( u \) input
\( z \) measurement
\( r \) measurement residual
\( P \) state covariance
\( Q \) process noise covariance
\( R \) measurement noise covariance
\( S \) residual covariance
\( K \) Kalman gain
\( F, G \) propagation Jacobians
\( H \) measurement Jacobian
\( N \) keyframe reset Jacobian
\( N_p \) position reset Jacobian
\( N_\theta \) attitude reset Jacobian

Operators
\( \otimes \) quaternion multiplication (Hamilton)
\( [\cdot] \) skew-symmetric matrix
\( (\cdot)^\wedge \) mapping from vector to quaternion
\( (\cdot)^\vee \) mapping from quaternion to vector
\( E[\cdot] \) expected value
\( R(q) \) rotation matrix associated with \( q \)

Other
\( \hat{y} \) estimate (or expected value) of \( y \)
\( \tilde{y} \) measurement of \( y \)
\( \dot{y} \) time derivative of \( y \)
\( y^+ \) a posteriori value of \( y \)
\( k \) unit vector \( [0 \ 0 \ 1]^T \)
\( I_{a \times b} \) identity matrix in \( \mathbb{R}^{a \times b} \)
\( M \) mass matrix \( \frac{1}{m}(I_{3 \times 3} - kk^T) \)
\( g \) gravity vector \( g_k \)
4.3 Quaternion Properties

Quaternions are a common method for representing attitude due to their improved computational complexity and accuracy compared to alternative approaches [57]. A variety of definitions exist for quaternions and their associated operations, leading to subtle discrepancies and potential confusion. The various approaches, described in more detail in [56], include left-handed vs. right-handed quaternion multiplication, active vs. passive representations, local-to-global vs. global-to-local attitude direction, and quaternion ordering. This section explicitly establishes the definitions and notation used throughout this paper and additionally derives several properties required for the filter’s derivation. This section is not intended as a complete introduction to quaternions, but rather as a summary of relevant points.

4.3.1 Quaternion Conventions

A quaternion \( q \in \mathbb{H} \) is a hyper-complex number of rank four consisting of a scalar and vector portion as
\[
q = q_0 + q_x i + q_y j + q_z k .
\]

We use the Hamilton definition of the quaternion, with
\[
\begin{align*}
ij &= -ji = k , \\
jk &= -kj = i , \\
ki &= -ik = j , \\
i^2 &= j^2 = k^2 = ijk = -1 .
\end{align*}
\]

For notational convenience we define the vector portion of the quaternion as
\[
\bar{q} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^T ,
\]
and write the quaternion as

\[
q = \begin{bmatrix}
\bar{q} \\
q_0
\end{bmatrix}.
\] (4.2)

Quaternion multiplication is denoted with the \( \otimes \) operator, and is carried out according to the rules in (4.1) and standard algebraic multiplication. Using the notation in (4.2), quaternion multiplication can be written as a matrix multiplication according to

\[
p \otimes q = \begin{pmatrix}
p_0 \mathbf{I} + [\bar{p}] & \bar{p} \\
-\bar{p}^T & p_0
\end{pmatrix}
\begin{pmatrix}
\bar{q} \\
q_0
\end{pmatrix},
\] (4.3a)

\[
= \begin{pmatrix}
q_0 \mathbf{I} - [\bar{q}] & \bar{q} \\
-\bar{q}^T & q_0
\end{pmatrix}
\begin{pmatrix}
\bar{p} \\
p_0
\end{pmatrix},
\] (4.3b)

where the operator \([\cdot]\) is the skew-symmetric operator

\[
[a] = \begin{bmatrix}
0 & -a_z & a_y \\
a_z & 0 & -a_x \\
-a_y & a_x & 0
\end{bmatrix}
\]

so that \(a \times b = [a] b\). The skew-symmetric operator has the property that

\[
[a] b = -[b] a.
\] (4.4)

The conjugate of a quaternion \(q\) is denoted by \(q^*\), and is equal to \(q\) but with the elements of the vector portion negated. The inverse of a quaternion is given by

\[
q^{-1} = \frac{q^*}{\|q\|}.
\]
The quaternions used in this paper all represent rotations and so are unit quaternions, meaning that their norm is 1. Therefore, for unit quaternions we have

\[ q^{-1} = q^* = \begin{bmatrix} -\bar{q} \\ q_0 \end{bmatrix}. \] (4.5)

Inverting the product of two quaternions results in the product of the inverse of each quaternion in the opposite order, as

\[ (p \otimes q)^{-1} = q^{-1} \otimes p^{-1}. \]

### 4.3.2 Vector Rotation

In this paper, quaternions are denoted passively, meaning that they represent the rotation necessary to express a vector in a different frame. Let quaternion \( q^b_a \) represent the rotation from frame \( a \) to frame \( b \) and let \( ^a y \) represent a vector expressed in frame \( a \). As described in [58], \( ^a y \) can be expressed in frame \( b \) as

\[ \begin{bmatrix} b y \\ 0 \end{bmatrix} = (q^b_a)^{-1} \otimes \begin{bmatrix} ^a y \\ 0 \end{bmatrix} \otimes q^b_a. \] (4.6)

The term \( \begin{bmatrix} ^a y^T \\ 0 \end{bmatrix}^T \) is referred to as the virtual quaternion constructed from \( ^a y \).

It is convenient to define an equivalent rotation matrix \( R(q) \) such that

\[ \begin{bmatrix} R(q)y \\ 0 \end{bmatrix} = q^{-1} \otimes \begin{bmatrix} y \\ 0 \end{bmatrix} \otimes q. \] (4.7)
An expression for $R(q)$ can be derived by expanding the right-hand side of (4.7) according to (4.5), (4.3a), and (4.3b) as

$$
\begin{bmatrix}
R(q)y \\
0
\end{bmatrix} = \begin{bmatrix}
q_0 I - |q| - \bar{q} \\
\bar{q}^T \\
q_0
\end{bmatrix} \begin{bmatrix}
y \\
0
\end{bmatrix} \otimes q
$$

$$
= q_0 y - |q| y
$$

$$
= \begin{bmatrix}
q_0 I - |q| & q \\
\bar{q}^T & q_0
\end{bmatrix} \begin{bmatrix}
q_0 y - |q| y \\
\bar{q}^T y
\end{bmatrix}
$$

$$
= \begin{bmatrix}
(q_0^2 I - 2q_0 |q| + \bar{q}q^T + |q|^2) y \\
0
\end{bmatrix},
$$

which implies that

$$
R(q) = q_0^2 I - 2q_0 |q| + \bar{q}q^T + |q|^2.
$$

It can be shown, however, that

$$
|q|^2 = \bar{q}q^T - (1 - q_0^2) I,
$$

so that

$$
R(q) = \left(2q_0^2 - 1\right) I - 2q_0 |q| + 2\bar{q}q^T.
$$

(4.9)

Rotation matrices exhibit the following properties:

$$
R^{-1}(q_b) = R^T(q_b) = R(q_a) = R(q_c) = \bar{q}q^T.
$$

$$
R(q_a) = R(q_c)R(q_b)
$$

$$
det(R(q)) = 1.
$$

(4.10)
The formula for vector rotation in equation (4.6) can be used to derive the manner in which two rotations are compounded together. If $q^b_a$ defines the rotation from frame $a$ to frame $b$, and $q^c_b$ the rotation from frame $b$ to frame $c$, then to take a vector expressed in frame $a$ and express it in frame $c$ we have

\[
\begin{bmatrix}
^c y \\
0
\end{bmatrix} = (q^c_b)^{-1} \otimes \begin{bmatrix}
^b y \\
0
\end{bmatrix} \otimes q^c_b \\
= (q^c_b)^{-1} \otimes \left((q^b_a)^{-1} \otimes \begin{bmatrix}
^a y \\
0
\end{bmatrix} \otimes q^b_a\right) \otimes q^c_b \\
= (q^b_a \otimes q^c_b)^{-1} \otimes \begin{bmatrix}
^a y \\
0
\end{bmatrix} \otimes (q^b_a \otimes q^c_b).
\]

We therefore conclude that

\[
q^c_a = q^b_a \otimes q^c_b. \tag{4.11}
\]

Comparing (4.10) and (4.11), we see that rotation matrices and quaternions compound in the opposite order:

\[
R(q^b_a \otimes q^c_b) = R(q^c_b)R(q^b_a). \tag{4.12}
\]

### 4.3.3 Unit Sphere Propagation

Attitude is represented using quaternions of unit length. Unit quaternions do not form a vector space, but rather form a group on the unit sphere $S^3 \subset \mathbb{H}$. The group operator is quaternion multiplication and the group of unit quaternions gives a double cover parameterization of the group of rotations $SO(3)$.
Multiplying a unit quaternion by a non-unit quaternion will cause the product to leave the unit sphere. Normalizing the resulting quaternion according to

\[ \mathbf{q} \leftarrow \frac{\mathbf{q}}{\|\mathbf{q}\|} \]

returns the quaternion to the unit sphere, but linearization errors are introduced.

To properly rotate a quaternion along the manifold, it is necessary to represent the rotation in terms of a unit quaternion. A rotation can be represented using a quaternion as

\[ \mathbf{q} = \begin{bmatrix} \hat{\mathbf{e}} \sin \frac{\theta}{2} \\ \cos \frac{\theta}{2} \end{bmatrix}, \quad (4.13) \]

where \( \hat{\mathbf{e}} \) is a unit vector defining the axis of rotation and \( \theta \) is the angle of rotation about that axis. Let \( \theta \triangleq \hat{\mathbf{e}} \in \mathbb{R}^3 \) define the magnitude and direction of rotation. The mapping from this three-vector rotation parameterization to a quaternion is denoted by the operator \( \wedge : \mathbb{R}^3 \rightarrow \mathbb{H} \) and the inverse mapping by the operator \( \triangledown : \mathbb{H} \rightarrow \mathbb{R}^3 \). The \( \wedge \) operator is defined using (4.13) as

\[ \theta \wedge \triangleq \begin{bmatrix} \frac{\theta}{\|\theta\|} \sin \frac{\|\theta\|}{2} \\ \cos \frac{\|\theta\|}{2} \end{bmatrix}. \quad (4.14) \]

Rotating a quaternion \( \mathbf{q} \) along the unit sphere by the rotation \( \theta \) is accomplished as \( \mathbf{q} \otimes \theta \wedge \). This is analogous to the notation \( \mathbf{q} \boxplus \theta \) and \( \mathbf{q} \otimes \exp \left( \frac{\theta}{2} \right) \) found in the estimation literature [59]. The \( \triangledown \) operator is defined as

\[ \mathbf{q} \triangledown \triangleq 2 \text{atan2} \left( \|\mathbf{q}\|, \mathbf{q}_0 \right) \frac{\mathbf{q}}{\|\mathbf{q}\|}. \quad (4.15) \]

Extracting the underlying rotation between \( \mathbf{q}_a \) and \( \mathbf{q}_b \) is accomplished as \( (\mathbf{q}_a \otimes (\mathbf{q}_b)^{-1}) \triangledown \). This is analogous to the notation \( \log(\mathbf{q}_a \boxdot \mathbf{q}_b) \) found in the estimation literature [59]. As inverse mappings, it can be shown that \( \theta = (\theta \wedge) \triangledown \) and \( \mathbf{q} = (\mathbf{q} \triangledown) \wedge \).
Figure 4.2: Quaternion definitions used in the derivation of the quaternion time derivative and integration. The same $q_{\Delta t}$ is used for both iRN and bRN because it corresponds to the body-fixed angular velocity $\omega$ measured by the rate gyros.

Eqn. (4.14) is undefined when $\|\theta\|$ equals zero, and in practice becomes numerically unstable as $\|\theta\|$ approaches zero. There are a number of common approximations of (4.14) for a small angle $\delta\theta$, such as the second-order Gibbs vector parameterization [55]

$$
\delta \theta^\wedge \approx \frac{1}{\sqrt{4 + \delta \theta^T \delta \theta}} \begin{bmatrix}
\delta \theta \\
2
\end{bmatrix}.
$$

(4.16)

The first-order approximation of both (4.14) and (4.16) is

$$
\delta \theta^\wedge \approx \begin{bmatrix}
\frac{1}{2} \delta \theta \\
1
\end{bmatrix}
$$

(4.17)

which is useful when deriving first-order Jacobians. Eqn. (4.15) can similarly be approximated for a small quaternion $\delta q$ as

$$
\delta q^\vee \approx 2 \text{sign}(\delta q_0) \delta \bar{q}.
$$

(4.18)

4.3.4 Time Integration

Several methods can be used to numerically integrate a quaternion that represents the attitude of a rigid body. Numerical integration is carried out over a finite time step $\Delta t$, 

78
and is governed by the angular velocity of the body with respect to an inertial frame as expressed in the body frame, $\omega \triangleq B^T \omega_I^B$. Let $q_{\Delta t}$ be an incremental quaternion and $\omega_0$ be the nominal angular velocity. Zero-order quaternion integration assumes that the angular velocity is constant over the timestep, $\omega_0 = \omega_t$, while first-order integration uses linear interpolation, $\omega_0 = \frac{1}{2}(\omega_t + \omega_{t-1})$. From (4.14), we write $q_{\Delta t}$ as

$$q_{\Delta t} = (\omega_0 \Delta t)^\wedge$$

$$= \begin{bmatrix}
\frac{\omega_0}{\|\omega_0\|} \sin \left( \frac{\|\omega_0\| \Delta t}{2} \right) \\
\cos \left( \frac{\|\omega_0\| \Delta t}{2} \right)
\end{bmatrix}. \tag{4.19}$$

The value of the quaternion $q_{t+\Delta t}$ at time $t + \Delta t$ can be expressed as the combination of the quaternion $q_t$ at time $t$ and the incremental quaternion $q_{\Delta t}$. The manner in which these quaternions is combined depends on whether the attitude quaternion represents the attitude of the body with respect to an inertial frame (iRN), or the attitude of an inertial frame with respect to the body (bRN). As illustrated in Figure 4.2, the attitude at time $t + \Delta t$ for these cases is

$$\text{iRN: } q_{t+\Delta t} = q_t \otimes q_{\Delta t}, \tag{4.20}$$

$$\text{bRN: } q_{t+\Delta t} = (q_{\Delta t})^{-1} \otimes q_t,$$

where the order of compounding follows equation (4.11).

Substituting (4.19) into (4.20) gives

$$\text{iRN: } q_{t+\Delta t} = q_t \otimes \begin{bmatrix}
\frac{\omega_0}{\|\omega_0\|} \sin \left( \frac{\|\omega_0\| \Delta t}{2} \right) \\
\cos \left( \frac{\|\omega_0\| \Delta t}{2} \right)
\end{bmatrix}, \tag{4.21a}$$

$$\text{bRN: } q_{t+\Delta t} = \begin{bmatrix}
-\frac{\omega_0}{\|\omega_0\|} \sin \left( \frac{\|\omega_0\| \Delta t}{2} \right) \\
\cos \left( \frac{\|\omega_0\| \Delta t}{2} \right)
\end{bmatrix} \otimes q_t. \tag{4.21b}$$
Integrating according to (4.21) maintains unit norm, allowing the attitude to propagate on the unit sphere $S^3 \subset \mathbb{H}$. In practice, however, this definition becomes numerically unstable as $\|\omega_0\|$ approaches zero. As described in [54], applying L'Hospital’s rule to (4.21a) for iRN shows that

$$\lim_{\|\omega_0\| \to 0} q_{t+\Delta t} = q_t + \Delta t \left( \frac{1}{2} \Omega(\omega_0) q_t \right),$$

(4.22)

where

$$\Omega(\omega) = \begin{bmatrix} -|\omega| & \omega \\ -\omega^T & 0 \end{bmatrix}.$$ 

(4.23)

Comparing (4.23) to (4.3b) shows that (4.22) can be written as

$$q_{t+\Delta t} = q_t + \Delta t \left( \frac{1}{2} q_t \otimes \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix} \right).$$

For bRN, a similar analysis can be applied to (4.21b) to show

$$q_{t+\Delta t} \approx q_t + \Delta t \left( \frac{1}{2} \begin{bmatrix} -\omega_0 \\ 0 \end{bmatrix} \otimes q_t \right).$$

Note that for bRN both the order of the quaternion multiplication and the sign of $\omega_0$ have been reversed.

In summary, the attitude quaternion is integrated according to (4.21) when $\|\omega_0\|$ is sufficiently large to avoid numerical issues. When $\|\omega_0\|$ is small, the integration is approximated for iRN as

$$\text{iRN: } q_{t+\Delta t} = q_t + \Delta t \left( \frac{1}{2} q_t \otimes \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix} \right),$$

(4.24a)
and for bRN as

\[ bRN: \quad q_{t+\Delta t} = q_t + \Delta t \left( \frac{1}{2} \begin{bmatrix} -\omega_0 \\ 0 \end{bmatrix} \otimes q_t \right). \]  

(4.24b)

Integrating according to (4.24) causes the quaternion to depart from the unit sphere \( S^3 \); when this method is used a normalization step therefore follows.

### 4.3.5 Attitude Kinematics

The attitude kinematics of a rigid body are described by the time derivative of the attitude quaternion. Some authors [59, 60] emphasize the manifold structure of attitude dynamics by defining \( \dot{q} \in \mathbb{R}^3 \) as a member of the associated Lie algebra, while other authors [54, 55] assume a first-order approximation such that \( \dot{q} \in \mathbb{R}^4 \). In this paper, while we use (4.21) to propagate along the manifold \( S^3 \), we use the first-order approximation of the quaternion dynamics over a finite timestep \( \Delta t \) for computing the first-order Jacobian matrices required by the extended Kalman filter.

The first-order Taylor series approximation of (4.21), which assumes \( \frac{1}{2} \|\omega_0\| \Delta t \) is small\(^2\), yields the same result as (4.24). Using (4.24a), the quaternion time derivative for iRN is computed as

\[
\dot{q} = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left( q_{t+\Delta t} - q_t \right)
\]

\[
\dot{q} = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left( q_t + \Delta t \left( \frac{1}{2} q_t \otimes \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix} \right) - q_t \right)
\]

\[
= \frac{1}{2} q_t \otimes \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix} .
\]

\(^2\)Even for a large \( \|\omega\| = 2\pi \) rad/s and moderate \( \Delta t = 0.01 \) s, the error introduced by linearizing the integration is only on the order of \( 10^{-6} \) rad.
Figure 4.3: Error state definitions for inertial and body-fixed relative navigation. To keep the same intermediate nominal frame, and as a result have similar error dynamics, the error state definitions are defined differently.

A similar analysis follows for bRN using (4.24b). In summary, attitude quaternion kinematics are represented by

\[
\text{iRN: } \dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} \omega_0 \\ 0 \end{bmatrix} ,
\]

\[
\text{bRN: } \dot{q} = \frac{1}{2} \begin{bmatrix} -\omega_0 \\ 0 \end{bmatrix} \otimes q .
\]

4.3.6 Error State

Because unit quaternions do not form a vector space, quaternion error cannot be computed using vector subtraction. Rather, a true quaternion state \( q \) is represented as the quaternion multiplication of an estimated quaternion \( \hat{q} \) and quaternion attitude error \( \delta q \). By varying the order and direction of the quaternion multiplication, there are four possible methods to define attitude error:

Method 1: \( q \triangleq \hat{q} \otimes \delta q \),

Method 2: \( q \triangleq \delta q \otimes \hat{q} \),
Method 3: \[ q \triangleq \hat{q} \otimes \delta q^{-1} \, , \]

Method 4: \[ q \triangleq \delta q^{-1} \otimes \hat{q} \, . \]

In this paper, we use (4.27a) for inertial relative navigation (iRN) described in Section 4.6 and use (4.27b) for body-fixed relative navigation (bRN) described in Section 4.7:

\[
\text{iRN: } q \triangleq \hat{q} \otimes \delta q, \quad (4.28a) \\
\text{bRN: } q \triangleq \delta q \otimes \hat{q}. \quad (4.28b)
\]

While using different definitions requires additional care when deriving the filter, ultimately this minimizes differences between the dynamics, measurement models, and keyframe reset steps of iRN and bRN. Figure 4.3 illustrates the choice of error state definitions and shows how this selection allows both approaches to keep the same intermediate nominal body frame \( \hat{b} \). Rearranging (4.28), the quaternion error state is defined as

\[
\text{iRN: } \delta q = \hat{q}^{-1} \otimes q, \quad (4.29a) \\
\text{bRN: } \delta q = q \otimes \hat{q}^{-1}. \quad (4.29b)
\]

Using (4.12), we can express (4.28) as

\[
\text{iRN: } R(q) = R(\delta q)R(\hat{q}), \quad (4.30a) \\
\text{bRN: } R(q) = R(\hat{q})R(\delta q). \quad (4.30b)
\]

When representing the attitude uncertainty associated with a quaternion error, a minimal representation is required. A quaternion is parameterized with four numbers, but only three are required to fully parameterize an orientation since orientations are associated with
unit quaternions, elements of the three dimensional group $S^3$. Because the group is three dimensional, the tangent space at the identity element, or Lie algebra, will be isomorphic to $\mathbb{R}^3$, and error covariances can be defined in this 3-dimensional vector space. Accordingly, we represent the uncertainty in $\delta q$ as the covariance of the vector $\delta \theta \in \mathbb{R}^3$. The $\wedge$ and $\vee$ operators of (4.17) and (4.18) define the mapping between the error quaternion $\delta q$ and its minimal representation $\delta \theta$ as

$$
\delta \theta = \delta q^\vee \approx 2 \text{sign}(q_0)q,
$$

$$
\delta q = \delta \theta^\wedge \approx \begin{bmatrix}
\frac{1}{2}\delta \theta \\
1
\end{bmatrix}.
$$

By substituting (4.31) into (4.9) and ignoring second-order terms, it is shown that

$$
R(\delta q) \approx I - [\delta \theta]
$$

and

$$
R(\delta q^{-1}) = R^T(\delta q) \approx I + [\delta \theta].
$$

### 4.3.7 Euler Decomposition

Aircraft attitude is commonly represented using three angles: roll $\phi$, pitch $\theta$, and yaw $\psi$. Yaw represents the rotation about the inertial $z$-axis (down). Pitch represents the rotation about the resulting $y$-axis. Roll represents the rotation about the $x$-axis formed after pitching and yawing. This sequence of rotations, known as 3-2-1 Euler angles, relates the vehicle’s body frame to an inertial frame, and can be represented as the multiplication of three rotation matrices

$$
R(q) = R_\phi R_\theta R_\psi,
$$

84
where

\[
\begin{align*}
R_\phi & \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix}, \\
R_\theta & \triangleq \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix}, \\
R_\psi & \triangleq \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{align*}
\]

Because quaternions are generally less intuitive, they are often mapped into roll, pitch, and yaw angles for plotting, analysis, and control. Expanding (4.34) using (4.9) and comparing terms, we obtain

\[
\begin{align*}
\phi &= \text{atan} \left( \frac{2q_0q_x + 2q_yq_z}{q_x^2 - q_y^2 - q_z^2 + q_0^2} \right), \\
\theta &= \text{asin} \left( 2q_0q_y - 2q_xq_z \right), \\
\psi &= \text{atan} \left( \frac{2q_0q_z + 2q_xq_y}{q_x^2 - q_y^2 - q_z^2 + q_0^2} \right).
\end{align*}
\]  

(4.35)

To map Euler angles into a quaternion, we first decompose the attitude quaternion into its roll, pitch, and yaw components using the order derived in (4.11) such that

\[
q = q_\psi \otimes q_\theta \otimes q_\phi.
\]  

(4.36)

Note the order of decomposition is opposite of (4.34) as described by (4.12). From (4.13), we get

\[
q_\phi \triangleq \begin{bmatrix} \sin \frac{\phi}{2} \\ 0 \\ 0 \\ \cos \frac{\phi}{2} \end{bmatrix}, \quad q_\theta \triangleq \begin{bmatrix} 0 \\ \sin \frac{\theta}{2} \\ 0 \\ \cos \frac{\theta}{2} \end{bmatrix}, \quad q_\psi \triangleq \begin{bmatrix} 0 \\ 0 \\ \sin \frac{\psi}{2} \\ \cos \frac{\psi}{2} \end{bmatrix}.
\]  

(4.37)
To show that (4.36) and (4.34) are consistent using the definitions in (4.37), we need only apply (4.9) and double-angle trigonometry identities to (4.37). For example, letting $\mathbf{k} \triangleq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, we see that

$$
R(q_\psi) = \left(2 \cos^2\left(\frac{\psi}{2}\right) - 1\right) \mathbf{I} - 2 \cos\left(\frac{\psi}{2}\right) \sin\left(\frac{\psi}{2}\right) [\mathbf{k}]
+ 2 \sin^2\left(\frac{\psi}{2}\right) \mathbf{k} \mathbf{k}^T
= \cos \psi \mathbf{I} - \sin \psi [\mathbf{k}] + (1 - \cos \psi) \mathbf{k} \mathbf{k}^T
= R_\psi.
$$

By substituting (4.37) into (4.36), a unit quaternion can be constructed from roll, pitch, and yaw angles as

$$
q_x = \cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2},
q_y = \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2},
q_z = \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2},
q_0 = \cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2}.
$$

### 4.3.8 Error Quaternion Properties

This section presents several properties of error quaternions that are needed in the derivation of the Jacobians used in the RMEKF. The first of these properties relates to the product of two error quaternions. Using (4.31) and (4.3a), and by dropping second-order terms, we obtain

$$
\delta \mathbf{q}_c = \delta \mathbf{q}_a \otimes \delta \mathbf{q}_b
$$
\[
\begin{bmatrix}
\frac{1}{2} \delta \theta_c \\
1
\end{bmatrix}
= \begin{bmatrix}
I + \frac{1}{2} |\delta \theta_a| \frac{1}{2} \delta \theta_a \\
-\frac{1}{2} \delta \theta_a^T \\
\end{bmatrix}
\begin{bmatrix}
\frac{1}{2} \delta \theta_b \\
1
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{2} \delta \theta_a + \frac{1}{4} |\delta \theta_a| \delta \theta_b + \frac{1}{2} \delta \theta_a \\
-\frac{1}{2} \delta \theta_a^T \delta \theta_b + 1
\end{bmatrix}
\approx
\begin{bmatrix}
\frac{1}{2} (\delta \theta_a + \delta \theta_b) \\
1
\end{bmatrix},
\]
from which we can conclude that

\[
\delta q_c = \delta q_a \otimes \delta q_b
\]

\[
\implies \delta \theta_c \approx \delta \theta_a + \delta \theta_b.
\] (4.38)

It can be similarly shown that

\[
\delta q_c = (\delta q_b)^{-1} \otimes \delta q_b \implies \delta \theta_c \approx -\delta \theta_a + \delta \theta_b,
\]

\[
\delta q_c = \delta q_a \otimes (\delta q_b)^{-1} \implies \delta \theta_c \approx \delta \theta_a - \delta \theta_b.
\]

For the next property, we first revisit the rotation of a vector by a quaternion as discussed in Section 4.3.2. Equation (4.7) is the canonical way of accomplishing this rotation, where a pure quaternion with a scalar component of value zero is constructed from a vector. It can be shown, however, that the same rotation can also be accomplished by constructing the quaternion from the vector with a scalar component of value one instead of zero. This is demonstrated using the same procedure outlined in (4.8), which shows that

\[
\begin{bmatrix}
R(q)y \\
1
\end{bmatrix}
= q^{-1} \otimes \begin{bmatrix}
y \\
1
\end{bmatrix} \otimes q
\] (4.39)
for the same $R(q)$ given by (4.9). Using equation (4.39) with (4.31), we see that

$$
\delta q_b = q^{-1} \otimes \delta q_a \otimes q
$$

$$
\Rightarrow \delta \theta_b \approx R(q) \delta \theta_a .
$$

(4.40)

In addition, (4.38) and (4.40) can be combined to verify that

$$
\delta q_c = \delta q_a \otimes q^{-1} \otimes \delta q_b \otimes q
$$

$$
\Rightarrow \delta \theta_c \approx \delta \theta_a + R(q) \delta \theta_b .
$$

(4.41)

4.4 Multiplicative Extended Kalman Filter

With the quaternion notation and properties established, we are prepared to outline the proposed estimation framework: a continuous-discrete, indirect, multiplicative extended Kalman filter. A Kalman filter provides the optimal, maximum-likelihood state estimate for a linear system under Gaussian noise. It recursively estimates the evolution of the system state $x$ as a function of the current state estimate $\hat{x} = E[x]$, input $u$, and measurement $z$. A Kalman filter also maintains an estimate of the state uncertainty, represented by the covariance matrix $P$, typically defined as

$$
P = E \left[ (x - E[x]) (x - E[x])^T \right] .
$$

(4.42)

The extended Kalman filter (EKF) is an extension of the Kalman filter for systems with nonlinear dynamics and/or nonlinear measurement models. The EKF linearizes the system about the current maximum-likelihood state estimate. While optimality and convergence are no longer guaranteed as opposed to a linear Kalman filter, EKFs are widely used in practice
for their robust performance and straightforward implementation. If the state \( \mathbf{x} \) includes a quaternion, however, (4.42) is fundamentally flawed. First, quaternion subtraction is not well-defined as described in Section 4.3.6, and second, (4.42) is never full rank because quaternions are not a minimal representation. These issues are addressed using an error-state, or indirect, formulation of the Kalman filter.

The indirect Kalman filter tracks the error state \( \delta \mathbf{x} \) and its uncertainty. Unlike the state \( \mathbf{x} \), the error state is defined as an element of a vector space by using a minimal attitude representation. The error state is a measure of the discrepancy between the true state \( \mathbf{x} \) and a nominal state \( \mathbf{x}_{\text{nom}} \), where \( \mathbf{x}_{\text{nom}} \) can be defined in a number of different ways, as described in [61]. When the system dynamics are especially well-modeled, such as for a spacecraft in orbit, \( \mathbf{x}_{\text{nom}} \) may be a predetermined feedforward state estimate. More commonly, the nominal state is the maximum likelihood state estimate \( \mathbf{x}_{\text{nom}} = E[\mathbf{x}] \). In this case, measurements provide feedback to update the nominal state, forcing the expected value of the error state to zero. Some indirect Kalman filter implementations differentiate between fast and slow measurements, and only update the nominal state for the slow measurements [61, 62]. For such systems, \( \mathbf{x}_{\text{nom}} \neq E[\mathbf{x}] \) at the fast rate, such that the expected error state is non-zero and must be propagated. For the derivation in this paper, the nominal state is updated equivalently for every measurement, ensuring that \( \mathbf{x}_{\text{nom}} = \hat{\mathbf{x}} = E[\mathbf{x}] \) at any given time.

Let \( \mathbf{x}_v \) and \( \mathbf{x}_q \) represent vector and quaternion portions of the state, and \( \delta \mathbf{x}_v \) and \( \delta \mathbf{x}_\theta \) be the corresponding error states. We define the error state as

\[
\delta \mathbf{x}_v \triangleq \mathbf{x}_v - \hat{\mathbf{x}}_v, \quad \tag{4.43a}
\]

\[
\delta \mathbf{x}_\theta \triangleq \delta \mathbf{q}^\prime, \quad \tag{4.43b}
\]
where $\delta \mathbf{q}$ is defined by (4.29) using $\mathbf{x}_q$ and $\dot{\mathbf{x}}_q$. This results in the following properties, which are derived in Appendix 4.10:

\begin{equation}
E[\delta \mathbf{x}] = 0,
\end{equation}

\begin{equation}
E[\delta \dot{\mathbf{x}}] = 0.
\end{equation}

Because the error state is part of a vector space, and as a result of (4.44), the indirect Kalman filter represents state uncertainty with the well-defined covariance

\begin{equation}
\mathbf{P} = E\left[ (\delta \mathbf{x} - E[\delta \mathbf{x}]) (\delta \mathbf{x} - E[\delta \mathbf{x}])^T \right] = E[\delta \mathbf{x} \delta \mathbf{x}^T].
\end{equation}

Kalman filters are decomposed into two steps: the *propagation* step and the *update* step, described in Sections 4.4.1 and 4.4.2. Section 4.4.3 discusses a method to handle delayed or out-of-order measurements. In general, the MEKF can use either an inertial or body-fixed coordinate frame. Several nuanced differences exist, however, and are highlighted as iRN- or bRN-specific.

### 4.4.1 Propagation

Consider the continuous-time system

\begin{equation}
\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u} + \mathbf{v}) + \eta,
\end{equation}

where $\mathbf{v} \sim \mathcal{N}(0, \mathbf{Q}_u)$ and $\eta \sim \mathcal{N}(0, \mathbf{Q}_\eta)$ are zero-mean Gaussian random variables. More specifically, we assume that $\eta$ and $\mathbf{v}$ are uncorrelated,

\begin{equation}
E[\eta \mathbf{v}^T] = 0,
\end{equation}

where $\delta \mathbf{q}$ is defined by (4.29) using $\mathbf{x}_q$ and $\dot{\mathbf{x}}_q$. This results in the following properties, which are derived in Appendix 4.10:

\begin{equation}
E[\delta \mathbf{x}] = 0,
\end{equation}

\begin{equation}
E[\delta \dot{\mathbf{x}}] = 0.
\end{equation}

Because the error state is part of a vector space, and as a result of (4.44), the indirect Kalman filter represents state uncertainty with the well-defined covariance

\begin{equation}
\mathbf{P} = E\left[ (\delta \mathbf{x} - E[\delta \mathbf{x}]) (\delta \mathbf{x} - E[\delta \mathbf{x}])^T \right] = E[\delta \mathbf{x} \delta \mathbf{x}^T].
\end{equation}

Kalman filters are decomposed into two steps: the *propagation* step and the *update* step, described in Sections 4.4.1 and 4.4.2. Section 4.4.3 discusses a method to handle delayed or out-of-order measurements. In general, the MEKF can use either an inertial or body-fixed coordinate frame. Several nuanced differences exist, however, and are highlighted as iRN- or bRN-specific.

### 4.4.1 Propagation

Consider the continuous-time system

\begin{equation}
\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u} + \mathbf{v}) + \eta,
\end{equation}

where $\mathbf{v} \sim \mathcal{N}(0, \mathbf{Q}_u)$ and $\eta \sim \mathcal{N}(0, \mathbf{Q}_\eta)$ are zero-mean Gaussian random variables. More specifically, we assume that $\eta$ and $\mathbf{v}$ are uncorrelated,
and that the input and process noise are not correlated in time:

\[ E[\eta(t)\eta(\tau)^T] = Q_x \delta(t - \tau) , \]
\[ E[\nu(t)\nu(\tau)^T] = Q_u \delta(t - \tau) , \] (4.48)

where \( \delta(t) \) is the Dirac delta function.

In the propagation step, the filter propagates the state estimates forward according to the expected value of the dynamics as

\[ \dot{x} = f(\hat{x}, u) \] (4.49)
\[ = E[\dot{x}] . \]

When \( x \) is part of a vector space, the Jacobians needed for propagating the covariance are obtained from the first-order Taylor series expansion of the error-state dynamics as

\[ \delta \dot{x} = f(x, u + \nu) + \eta - f(\hat{x}, u) \] (4.50a)
\[ \approx f(\hat{x}, u) + \left. \frac{\partial f}{\partial x} \right|_{\hat{x}, u} \delta x + \left. \frac{\partial f}{\partial u} \right|_{\hat{x}, u} \nu + \eta - f(\hat{x}, u) \]
\[ = \left. \frac{\partial f}{\partial x} \right|_{\hat{x}, u} \delta x + \left. \frac{\partial f}{\partial u} \right|_{\hat{x}, u} \nu + \eta . \] (4.50b)

When the state \( x \) includes quaternion terms, however, the error-state dynamics cannot be formed by simple subtraction as in (4.50a). In fact, as described in Section 4.3.6, the error state \( \delta x \) is of lower dimensionality than \( x \). Noting that \( x \) is a function of \( \hat{x} \) and \( \delta x \) according to (4.43), the dynamics of the error state can be expressed generally as a single function

\[ \delta \dot{x} = \tilde{f}(\delta x, \nu, \hat{x}, u) + \eta , \] (4.51)
where only $\delta x$, $\nu$, and $\eta$ are stochastic variables\textsuperscript{3}. The function $\bar{f}$ then handles the quaternion portions of the state appropriately. Note also that from (4.44) and $E[\eta] = 0$, we have that $E \left[ \bar{f}(\delta x, \nu, \hat{x}, u) \right] = \bar{f}(E[\delta x], E[\nu], \hat{x}, u) = 0$. The Jacobians for the covariance propagation are computed from the first-order Taylor series expansion of (4.51) about $(E[\delta x], E[\nu], \hat{x}, u)$ as

\[
\delta \dot{x} \approx \bar{f}(E[\delta x], E[\nu], \hat{x}, u) + \eta \\
+ F(\delta x - E[\delta x]) + G(\nu - E[\nu]) \\
= F\delta x + G\nu + \eta,
\]

(4.52)

where

\[
F = \left. \frac{\partial \bar{f}(\delta x, \nu, \hat{x}, u)}{\partial \delta x} \right|_{E[\delta x], E[\nu], \hat{x}, u}
\]

and

\[
G = \left. \frac{\partial \bar{f}(\delta x, \nu, \hat{x}, u)}{\partial \nu} \right|_{E[\delta x], E[\nu], \hat{x}, u},
\]

with $E[\delta x] = E[\nu] = 0$. Note that (4.52) has the form of (4.50b), but the Jacobian terms differentiate the error-state dynamics with respect to the error state and input noise, rather than with respect to the state and input.

The error-state covariance propagation is given by differentiating (4.45) with respect to time and utilizing the linearized error dynamics from (4.52) as

\[
\dot{P} = E \left[ \delta \dot{x} \delta x^\top + \delta x \delta \dot{x}^\top \right]
\]

\textsuperscript{3}Note our slight abuse of notation in that $\eta$ in (4.51) is of the same dimensionality as the error vector, while in (4.46) $\eta$ is of the same dimensionality as the state vector.
\[
= E \left[ F \delta x \delta x^T + G \nu \delta x^T + \eta \delta x^T \right] \\
+ E \left[ \delta x \delta x^T F + \delta x \nu^T G + \delta x \eta^T \right] \\
= FP + PF^T + E \left[ G \nu \delta x^T + \delta x \nu^T G \right] \\
+ E \left[ \eta \delta x^T + \delta x \eta^T \right].
\] (4.53)

To simplify terms, we solve the differential equation in (4.52) with initial conditions \( \delta x_0 \) to obtain

\[
\delta x(t) = e^{Ft} \delta x_0 + \int_0^t e^{F(t-\tau)} G \nu(\tau) d\tau + \int_0^t e^{F(t-\tau)} \eta(\tau) d\tau.
\] (4.54)

Using (4.54) and the properties (4.47) and (4.48), we see that

\[
E \left[ \delta x \nu^T G^T \right] = E \left[ e^{Ft} \delta x_0 \nu^T G^T \right] \\
+ E \left[ \int_0^t e^{F(t-\tau)} G \nu(\tau) \nu^T(t) G^T d\tau \right] \\
+ E \left[ \int_0^t e^{F(t-\tau)} \eta(\tau) \nu^T(t) G^T d\tau \right] \\
= E \left[ \int_0^t e^{F(t-\tau)} G Q_u \delta(t-\tau) G^T d\tau \right] \\
= \frac{1}{2} G Q_u G^T
\] (4.55)

where the \( \frac{1}{2} \) is because the bounds of integration only use half of the area inside of the delta function. Similarly,

\[
E \left[ \delta x \eta^T \right] = \frac{1}{2} Q_x.
\] (4.56)

Because \( Q_u \) and \( Q_x \) are symmetric, combining (4.53), (4.55), and (4.56) we have that \( P \) evolves between measurements as

\[
\dot{P} = FP + PF^T + G Q_u G^T + Q_x.
\] (4.57)
In summary, during the propagation step, $\hat{x}$ is propagated forward using (4.49) and the error covariance is propagated forward using (4.57). Also, since the evolution of $\delta x$ is given by (4.52) and $E[\delta x](0) = 0$, we have that $E[\delta x](t) = 0$ over the propagation window.

4.4.2 Measurement Update

For the update step, consider the measurement

$$z = h(x, \eta_z),$$

where $\eta_z \sim \mathcal{N}(0, R)$ represents measurement noise. This measurement noise is usually additive if the measurement is a vector quantity, but if the measured quantity is a quaternion the noise is applied through quaternion multiplication.

The residual $r$ is the discrepancy between the true measurement and the expected measurement

$$E[z] = h(\hat{x}, 0).$$

Conventionally a vector space measurement is assumed, such that (4.58) is simplified as

$$z = h(x) + \eta_z.$$

In this case, the residual is found by subtraction,

$$r = z - h(\hat{x}) = h(x) + \eta_z - h(\hat{x}),$$

94
and Jacobians are computed using the first-order Taylor-series expansion as

\[
\mathbf{r} \approx \mathbf{h}(\hat{\mathbf{x}}) + \left. \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}} \delta \mathbf{x} + \eta_z - \mathbf{h}(\hat{\mathbf{x}})
\]

\[
= \left. \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}} \delta \mathbf{x} + \eta_z .
\]  

(4.59)

These measurement and residual models share similar shortcomings with the conventional error-state dynamics presented in (4.50), namely they do not hold for quaternion measurements and (4.59) assumes the state \( \mathbf{x} \) is of the same dimensionality as \( \delta \mathbf{x} \).

These issues can be addressed by expressing the residual as a single function, paralleling the approach taken for the propagation step in Section 4.4.1. Again noting that \( \mathbf{x} \) is a function of \( \hat{\mathbf{x}} \) and \( \delta \mathbf{x} \), the residual is expressed as a function

\[
\mathbf{r} = \tilde{\mathbf{h}}(\delta \mathbf{x}, \eta_z, \hat{\mathbf{x}}) ,
\]

where only the first two variables are stochastic. For measurements of quaternion values, the residual is expressed as the three-vector minimal representation of the error between the observed and expected quaternions. The residual has the property that \( E[\mathbf{r}] = \tilde{\mathbf{h}}(E[\delta \mathbf{x}], E[\eta_z], \hat{\mathbf{x}}) = 0 \). The measurement models in this paper are chosen so that the noise is additive in the residual space, implying that \( \frac{\partial \mathbf{r}}{\partial \eta_z} = \mathbf{I} \). The measurement Jacobians are computed using the first-order Taylor-series expansion of \( \mathbf{h} \) about \( (E[\delta \mathbf{x}], E[\eta_z], \hat{\mathbf{x}}) \) as

\[
\mathbf{r} \approx \tilde{\mathbf{h}}(E[\delta \mathbf{x}], E[\eta_z], \hat{\mathbf{x}}) + \mathbf{H} (\delta \mathbf{x} - E[\delta \mathbf{x}]) + \mathbf{I}(\eta_z - E[\eta_z])
\]

\[
= \mathbf{H} \delta \mathbf{x} + \eta_z ,
\]

where

\[
\mathbf{H} = \left. \frac{\partial \tilde{\mathbf{h}}(\delta \mathbf{x}, \eta_z, \hat{\mathbf{x}})}{\partial \delta \mathbf{x}} \right|_{E[\delta \mathbf{x}], E[\eta_z], \hat{\mathbf{x}}}
\]

and \( E[\delta \mathbf{x}] = E[\eta_z] = 0 \).
For a measurement $z_v$ of a vector value, the residual is computed as

$$r = z_v - h(\hat{x}, u, 0) \quad (4.60)$$

When a quaternion measurement $z_q$ is available, the relevant error-state definition (4.28) is used so that

\begin{align*}
iRN: \quad r &= \left(h(\hat{x}, u, 0)^{-1} \otimes z_q\right)^\vee \\
bRN: \quad r &= \left(z_q \otimes h(\hat{x}, u, 0)^{-1}\right)^\vee.
\end{align*}

(4.61)

Assuming the measurement noise, error state, and input noise are uncorrelated, the residual uncertainty is

$$S \triangleq E[rr^T] = E[H\delta x \delta x^T H^T + \eta_z \eta_z^T] = HPH^T + R \quad (4.62)$$

The Kalman gain uses the residual and state uncertainty to find the extent to which the residual should be trusted and applied. Using the residual covariance, the Kalman gain is

$$K = PH^T S^{-1} \quad (4.63)$$

In fusing the information provided by the measurement, the \textit{a posteriori} estimate of the error state, denoted with a $+$, is

$$\delta x^+ = \delta x + Kr,$$

implying that

$$E[\delta x^+] = Kr.$$
With the additional information provided by the measurement update, the error state is no longer zero-mean, violating the properties in (4.44). Let \( \mathbf{x}_v \) and \( \mathbf{x}_q \) again be vector and quaternion states within \( \mathbf{x} \), and let \( \Delta \mathbf{v} \) and \( \Delta \mathbf{\theta} \) be the corresponding portions of the Kalman update \( \mathbf{K} \). To ensure the error state remains zero mean, the Kalman update \( \mathbf{K} \) is used to adjust \( \hat{\mathbf{x}} \) as

\[
\hat{x}_v^+ = \hat{x}_v + \Delta \mathbf{v} \quad (4.64a)
\]

iRN: \( \hat{x}_q^+ = \hat{x}_q \otimes \Delta \mathbf{\theta}^\wedge \quad (4.64b) \)

bRN: \( \hat{x}_q^+ = \Delta \mathbf{\theta}^\wedge \otimes \hat{x}_q \quad (4.64c) \)

where (4.64b) and (4.64c) are specific to the quaternion error-state definition used. Appendix 4.10 presents the derivation of (4.64).

Finally, the covariance is updated conventionally as

\[
\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{H}) \mathbf{P} .
\]

In practice we use the Joseph form Kalman update,

\[
\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{H}) \mathbf{P} (\mathbf{I} - \mathbf{K}\mathbf{H})^\top + \mathbf{K}\mathbf{R}\mathbf{K}^\top, \quad (4.65)
\]

because it improves numerical stability and ensures that the covariance matrix remains symmetric [2].

In summary, during the update step the measurement residual (4.60) or (4.61) provides additional information causing a non-zero \( E[\delta \mathbf{x}] \). Using the Kalman gain (4.63), \( \hat{\mathbf{x}} \) is updated according to (4.64) to ensure \( E[\delta \mathbf{x}] \) remains zero mean. The error-state covariance is updated according to (4.65).
4.4.3 Delayed Out-of-Order Measurements

The Kalman filter assumes the state evolves according to a Markov process. As such, state estimates for all previous time steps are marginalized. This makes it difficult to compute a residual for delayed or out-of-order measurements when they finally arrive. In practice, delayed measurements are common. For example, visual odometry algorithms may require hundreds of milliseconds to perform the computer vision operations necessary to compute a measurement. Ideally this measurement is used to constrain the state of the vehicle when the image was taken, not to constrain the state of the vehicle when the measurement arrives.

To address delayed measurements, we use priority queues to save the last $T$ seconds of inputs, measurements, states, and error-state covariances ordered by time. If a delayed measurement arrives with a time stamp more than $T$ seconds old, we discard the measurement. Otherwise, we discard all saved states and error-state covariances that have a time stamp later than the time stamp of the incoming delayed measurement. At this point, we are left with the state and covariance estimate at the instant the delayed measurement should have arrived. We apply the measurement normally, and then use the input and measurement queues to re-propagate the MEKF to the current time instance. This approach provides the same state estimate as if all measurements had arrived at the correct time. Handling delayed messages in this way may not be practical for all processors. Similar methods are described in [23, 63].

4.5 Relative Navigation

Section 4.4 provided a general overview of the indirect MEKF. This section describes how the MEKF is adapted to the relative navigation framework.

Conventional filtering approaches directly estimate the vehicle’s global state with respect to some inertially-fixed origin, such as the GPS origin or the vehicle’s starting location; how-
ever, when only relative position measurements such as those obtained from visual odometry or laser scan matching are available, the vehicle’s global position and heading are unobservable [5, 30, 31]. Over time, directly estimating these unobservable states leads to inconsistency and performance issues in the filter [13, 51, 52]. Methods for mitigating these issues have been proposed [13, 14, 52], but the core underlying issue of unobservable states can be avoided entirely by reformulating the problem in terms of relative states [8].

The relative navigation approach maintains observability of the filter states by estimating the pose of the vehicle with respect to a local coordinate frame referred to as the node frame. This node frame is positioned at zero altitude directly below the most recent odometry keyframe, but is gravity-aligned (i.e. the heading is aligned with the vehicle’s heading when the keyframe was declared, but there is no pitch or roll). As a result, the odometry provides nearly-direct measurements of the position and heading of the vehicle with respect to the current node frame, making those states observable by construction.

Because of the way the node frame is defined, the roll and pitch components of the vehicle’s attitude ($\phi, \theta$), as well as the vehicle’s altitude $p_z$, are estimated no differently than if defined with respect to a global origin. These states are not affected when transitioning from one node frame to another and so are, in effect, independent of the current node frame. On the other hand, the horizontal position and heading states ($p_x, p_y, \psi$) define how the vehicle has moved since the last node frame, and are termed relative states. Each time a new keyframe is declared, a new node frame is also declared and the relative states ($p_x, p_y, \psi$) are reset to zero. The covariances associated with the relative states are also reset to zero, since the vehicle is, by definition, at the location of the node frame and so there is no uncertainty in these states. This is illustrated in Figure 4.4. The non-relative states (roll, pitch, altitude, body-fixed velocities, and body-fixed IMU biases) and their covariances are unchanged by the keyframe reset operation. Note that resetting the heading component of the state and covariance is non-trivial when attitude is parameterized with a quaternion.
Prior to the reset, the vehicle’s current relative pose estimate and covariance are passed to a back-end pose-graph map that concatenates the relative poses into an estimate of the vehicle’s global path and current global pose [8].

We describe the keyframe reset operation mathematically as follows. Let \( n(x) \) define the keyframe reset operation. The estimated state is reset as

\[
\hat{x}^+ = n(\hat{x}). \tag{4.66}
\]

The error state after the reset, \( \delta x^+ \), is the difference between \( x^+ \) and \( \hat{x}^+ \) as defined by (4.43). Again recalling that since \( x \) is a function of \( \delta x \) and \( \hat{x} \), we can express the error state after the reset as a single function

\[
\delta x^+ = n(\delta x, \hat{x}).
\]

This allows the covariance to be updated as

\[
P^+ = NPN^T, \tag{4.67}
\]

where

\[
N = \left. \frac{\partial n(\delta x, \hat{x})}{\partial \delta x} \right|_{E[\delta x], \hat{x}}.
\]

The details of the reset operation \( n(x) \) and the Jacobian \( N \) are presented in Section 4.6.3 for iRN and in Section 4.7.3 for bRN.

This formulation provides several advantages in terms of estimator performance. One of these advantages is that when the covariance associated with the relative states is reset to zero, uncertainty is in essence removed from the filter and delegated to the back-end map, which helps to maintain filter consistency [40]. As a result, the covariance in the filter also remains bounded. In addition, since the distances between keyframes are relatively
Figure 4.4: Illustration of the keyframe reset operation as viewed from above. Before the reset (4.4a), there is some error between the estimated pose (blue) and the true pose (green), with the estimated uncertainty represented by the blue covariance ellipse. When the reset occurs, the vehicle is known to be by definition at the location of the keyframe. As a result, after the reset (4.4b), both the relative states and the corresponding elements of the covariance matrix are set identically to zero and there is no error.

small, the state error remains small, avoiding significant linearization errors that can cause inconsistency in a global estimator [14].

While reconstructing the global pose estimate requires implementing a back-end pose-graph map, this architecture has also been shown to improve consistency and accuracy of the final global pose estimate over global filter approaches, even in the absence of additional pose-graph constraints such as loop closures [8]. It also has the advantage of avoiding large, potentially destabilizing state jumps when new global information becomes available. The requirement to implement this pose-graph map is also not a particularly onerous one, seeing as some global filter approaches already use a back-end batch-processed map to provide updates to their global filter [23, 36].

The various steps for implementing an RMEKF are summarized in Algorithm 1, along with references to the key equations. The specific implementation equations are derived in Section 4.6 for inertial relative navigation and in Section 4.7 for body-fixed relative navigation.
Algorithm 1 Relative multiplicative extended Kalman filter (RMEKF)

1: Initialize: $\hat{x} = x_0$
2: Initialize: $P = P_0$
3: for Each new available input $u$ do
4: Propagate nominal state $\hat{x}$ using (4.49)
5: Propagate error-state covariance $P$ using (4.57)
6: for $i$ in sensors do
7: if Measurement is available from sensor $i$ then
8: Compute residual $r$ using (4.60) or (4.61)
9: Compute residual uncertainty $S$ using (4.62)
10: Compute Kalman gain $K$ using (4.63)
11: Use $Kr$ to update $\hat{x}$ using (4.64)
12: Update error-state covariance $P$ using (4.65)
13: if New keyframe is declared then
14: Save $\hat{x}$ and $P$ as edge in pose-graph back end
15: Reset state using (4.66)
16: Reset uncertainty using (4.67)

4.6 Inertial Relative Navigation (iRN)

Inertial relative navigation (iRN) estimates the vehicle’s position and attitude with respect to the current node frame. While the current node frame changes regularly, each is gravity-aligned and inertially defined. For this reason, typical UAS dynamics from the GPS/INS literature are applicable. Section 4.6.1 outlines the input, state, and dynamics for the system, including the error-state dynamics. Section 4.6.2 defines the measurement models and Section 4.6.3 outlines the keyframe reset step.

4.6.1 State Dynamics

This section derives the state propagation model used in the filter. Section 4.6.1.1 defines the state vector. Section 4.6.1.2 discusses how measurements from an inertial measurement unit (IMU) are incorporated into the propagation model, and Section 4.6.1.3 defines the state propagation model and derives the associated Jacobians.
4.6.1.1 State Vector

Vectors use a forward-right-down coordinate frame, with axes labeled $x$, $y$, and $z$. In this paper, the position vector $p^b_a$ denotes the position of frame $b$ with respect to frame $a$, expressed in frame $c$. Unless otherwise noted, position vectors are expressed in the originating frame, i.e. frames $a$ and $c$ are the same. When this is the case the prescript is usually omitted for brevity, so that $p^b_a \equiv a^p_b$. However, the prescript is occasionally included for clarity.

Inertial relative navigation estimates the position and attitude of the vehicle’s body frame $b$ with respect to and expressed in the current node frame $n$, denoted by the pose $(p^b_n, q^b_n)$. Let frame $k$ represent the vehicle’s body frame at the instant in time that the current keyframe image was taken. The estimator tracks the pose of $k$ with respect to and expressed in $n$, denoted as $(p^k_n, q^k_n)$. As illustrated in Figure 4.5a, the keyframe states $(p^k_n, q^k_n)$ include the altitude, roll, and pitch of the vehicle at the moment a keyframe is declared. The vehicle state is

\[
x \triangleq \begin{bmatrix}
p^b_n \quad q^b_n \quad v^T \quad \beta_\omega^T \quad \beta_a^T \quad p^k_n \quad q^k_n \quad \mu
\end{bmatrix}^T,
\]

and has 24 elements, where $v \in \mathbb{R}^3$ is the vehicle’s inertial velocity expressed in the body frame (i.e. $v \triangleq |v|_1$), $\beta_\omega \in \mathbb{R}^3$ and $\beta_a \in \mathbb{R}^3$ are unknown biases for the gyro and accelerometer, and $\mu \in \mathbb{R}$ is the lateral drag coefficient. The error state $\delta x \in \mathbb{R}^{22}$ is defined.
\[ \delta \mathbf{x} = \left[ \delta \mathbf{p}_n^b \delta \mathbf{v}_n^b \delta \mathbf{v}_\omega^b \delta \mathbf{v}_a^b \delta \mathbf{p}_k^b \delta \mathbf{v}_\theta^b \delta \mathbf{v}_\mu^b \right]^T. \]

### 4.6.1.2 Mechanization

Let \( \omega \triangleq b \omega_i \) and \( \mathbf{a} \triangleq b \mathbf{a}_i \) respectively define the ideal body-fixed angular rates and accelerations as measured by an IMU. For brevity we omit specifying the frames and write \( \omega \) and \( \mathbf{a} \) throughout the paper. An IMU provides measurements \( \tilde{\omega} \) and \( \tilde{\mathbf{a}} \) that are corrupted by unknown biases \( \beta_\omega \) and \( \beta_a \) and zero-mean Gaussian noise processes \( \mathbf{v}_\omega \) and \( \mathbf{v}_a \) such that

\[
\begin{align*}
\tilde{\omega} &= \omega + \beta_\omega + \mathbf{v}_\omega \quad \text{(4.68a)} \\
\tilde{\mathbf{a}} &= \mathbf{a} + \beta_a + \mathbf{v}_a \quad \text{(4.68b)}
\end{align*}
\]

The expected value of the true angular rate is

\[
\hat{\omega} \triangleq E[\omega] = \tilde{\omega} - \hat{\beta}_\omega
\]

and the expected value of the true acceleration is

\[
\hat{\mathbf{a}} \triangleq E[\mathbf{a}] = \tilde{\mathbf{a}} - \hat{\beta}_a
\]

104
Figure 4.6: Free body diagram describing the forces acting on a multirotor. The principal forces are gravity $F_G$, thrust $F_T$, and drag $F_D$. The vector $v$ represents the velocity of the vehicle, and $g$ represents the gravity vector. The gravity force acts in the direction of gravity, the thrust force acts perpendicular to the plane of the rotors, and the simplified drag force acts opposite the velocity vector.

IMU data can be treated either as a system input or as a measurement, in a trade-off based on IMU quality and model accuracy. When a reliable vehicle model is available, the controller’s output (e.g. motor commands) can be used as an input to the observer to propagate the state forward. In this case, IMU data are incorporated as measurement updates providing feedback. This approach leverages the most information but requires careful characterization of the vehicle’s dynamics. Another common approach, known as mechanization, treats the IMU measurements directly as inputs to the filter dynamics, which replaces the vehicle-specific dynamics with kinematic equations. This simplifies the propagation dynamics and eliminates sensitivity to modeling errors, but does not use any information about how the vehicle behaves.

For the filter design in this paper, the angular velocity measured by the rate gyros, $\tilde{\omega}$, is treated as an input to the propagation equations. Following [64], the $z$ component of the accelerometer measurement is also treated as an input, while the $x$ and $y$ components are used as measurement updates. The following paragraphs explain the derivation and justification for this approach.

Using Newton’s second law, the velocity dynamics can be modeled as

$$\dot{v} = [v] \omega + \frac{1}{m} \sum gF,$$  (4.71)
where \( m \) is the mass of the vehicle and \( ^B \mathbf{F} \) are the forces acting on the vehicle expressed in the body frame. As illustrated in Figure 4.6, the principal forces that act on a multirotor are gravity, \( ^B \mathbf{F}_G = m \mathbf{R}(\mathbf{q}_n^b)g \), thrust, \( ^B \mathbf{F}_T = T \mathbf{k} \), and the simplified drag force, \( ^B \mathbf{F}_D = -\mu \mathbf{v} \). Here \( \mathbf{k} \triangleq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \), \( T \) is the total rotor thrust, and \( \mathbf{g} = g \mathbf{k} \) with \( g \) being the standard acceleration due to gravity. Substituting these forces into (4.71) yields

\[
\dot{\mathbf{v}} = [\mathbf{v}] \mathbf{\omega} + \frac{1}{m} \left( ^B \mathbf{F}_G + ^B \mathbf{F}_T + ^B \mathbf{F}_D \right) \\
= [\mathbf{v}] \mathbf{\omega} + \mathbf{R}(\mathbf{q}_n^b) \mathbf{g} + \frac{T}{m} \mathbf{k} - \frac{\mu}{m} \mathbf{v} .
\] (4.72)

While technically correct, the dynamics in (4.72) are challenging to use in practice because the thrust \( T \) is difficult to model and so is generally unknown. This difficulty is addressed through mechanization by utilizing the accelerometer measurements directly in the propagation model. As explained in [64], accelerometers measure the specific force (not including gravity) expressed in the body frame, so that

\[
\mathbf{a} = \frac{1}{m} \sum ^B \mathbf{F} - ^B \mathbf{F}_G \\
= \frac{1}{m} \left( ^B \mathbf{F}_T + ^B \mathbf{F}_D \right) \\
= \frac{T}{m} \mathbf{k} - \frac{\mu}{m} \mathbf{v} .
\] (4.73)

Substituting (4.73) into (4.72) yields the common mechanization dynamics

\[
\dot{\mathbf{v}} = [\mathbf{v}] \mathbf{\omega} + \mathbf{R}(\mathbf{q}_n^b) \mathbf{g} + \mathbf{a} .
\] (4.74)

Equation (4.74) eliminates the need for modeling complicated vehicle dynamics, but ignores information about the dynamics that might improve estimator performance by building up cross-correlation terms in the covariance matrix \( \mathbf{P} \).
In practice, we have found the most success using a combination of (4.72) and (4.74). Noting that the unknown thrust term in (4.72) appears only in the $z$ component, a hybrid propagation model is obtained by substituting only the $z$ component of (4.73) into (4.72) to obtain

\[
\dot{\mathbf{v}} = [\mathbf{v}] \mathbf{\omega} + \mathbf{R}(\mathbf{q}_{n}^{b}) \mathbf{g} + a_z \mathbf{k} - \mu \mathbf{M} \mathbf{v},
\]

(4.75)

where $\mathbf{M} \triangleq \frac{1}{m}(\mathbf{I} - \mathbf{kk}^T)$ and $a_z = \mathbf{k}^T \mathbf{a}$.

As a result of this hybrid approach, we consider the gyroscope and $z$-axis accelerometer measurements as system inputs, while using the horizontal accelerometer measurements as feedback in the update step. The system input and input noise are therefore defined as

\[
\mathbf{u} = \begin{bmatrix} \mathbf{\omega} \\ \tilde{a}_z \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_\omega \\ v_{a_z} \end{bmatrix},
\]

where $\tilde{a}_z = \mathbf{k}^T \mathbf{a}$ and $v_{a_z} = \mathbf{k}^T v_a$.

### 4.6.1.3 Propagation Model

The system dynamics are modeled as

\[
\dot{\mathbf{p}}_n = \mathbf{R}(\mathbf{q}_{n}^{b}) \mathbf{v} \quad (4.76a)
\]

\[
\dot{\mathbf{q}}_n^b = \frac{1}{2} \mathbf{q}_n^{b} \otimes \begin{bmatrix} \mathbf{\omega} \\ \mathbf{0} \end{bmatrix} \quad (4.76b)
\]

\[
\dot{\mathbf{v}} = [\mathbf{v}] \mathbf{\omega} + \mathbf{R}(\mathbf{q}_{n}^{b}) \mathbf{g} + a_z \mathbf{k} - \mu \mathbf{M} \mathbf{v} + \eta_{\mathbf{v}} \quad (4.76c)
\]

\[
\dot{\mathbf{\beta}}_\omega = \eta_{\beta_\omega}
\]

\[
\dot{\mathbf{\beta}}_a = \eta_{\beta_a}
\]

\[
\dot{\mathbf{p}}_n^k = 0
\]
\[
\dot{q}_n^k = 0
\]
\[\dot{\mu} = \eta_{\mu},\]

where \(\eta_v\), \(\eta_{\beta_\omega}\), \(\eta_{\beta_a}\), and \(\eta_{\mu}\) are zero-mean Gaussian noise processes for the corresponding states. The state \(p_n^b\) is propagated according to a standard kinematic model, \(q_n^b\) is propagated according to (4.25) and the discussion in Section 4.3.4, and \(v\) is propagated according to (4.75). The dynamics for the bias states \(\beta_\omega\) and \(\beta_a\) and drag coefficient \(\mu\) are modeled as random walks, while the keyframe states \(p_n^k\) and \(q_n^k\) represent the relative pose of static coordinate frames and therefore do not change.

The state estimate is propagated using the expected value of (4.76) as

\[
\dot{p}_n^b = R^T(\hat{q}_n^b) \hat{v} \\
\dot{q}_n^b = \frac{1}{2} q_n^b \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \\
\dot{v} = [\hat{v}] \hat{\omega} + R(\hat{q}_n^b) g + \hat{a}_z k - \hat{\mu} M \hat{v} \\
\dot{\beta}_\omega = 0 \\
\dot{\beta}_a = 0 \\
\dot{p}_n^k = 0 \\
\dot{q}_n^k = 0 \\
\dot{\mu} = 0
\] (4.77a)

The error-state dynamics are found by relating (4.76) and (4.77) using the error-state definition (4.43). The first-order approximation of the error-state dynamics are

\[
\delta p_n^b \approx -R^T(\hat{q}_n^b) [\hat{v}] \delta \theta_n^b + R^T(\hat{q}_n^b) \delta v
\]
\[ \delta \dot{\theta}_n^b \approx - \left[ \omega - \hat{\beta}_\omega \right] \delta \theta_n^b - \delta \beta_\omega - \nu \omega \]
\[ \delta \dot{v} \approx \left[ R(q_n^b)g \right] \delta \theta_n^b + \left( - \left[ \omega - \hat{\beta}_\omega \right] - \mu M \right) \delta v \]
\[ = - \left[ \hat{v} \right] \delta \beta_\omega - kk^T \delta \beta_a - M \hat{v} \delta \mu \]
\[ = - \left[ \hat{v} \right] \nu - k \eta_a + \eta_v \]
\[ \delta \dot{\beta}_\omega = \eta_{\beta_\omega} \]
\[ \delta \dot{\beta}_a = \eta_{\beta_a} \]
\[ \delta \dot{\beta}_k = 0 \]
\[ \delta \dot{\beta}_k = 0 \]
\[ \delta \dot{\mu} = \eta_{\mu} \]

and are derived in Appendix 4.11. Differentiating the error state dynamics with respect to the error state and input noise results in the following propagation Jacobians:

\[ F = \begin{bmatrix}
0 & -R^T(q_n^b) \hat{v} & R^T(q_n^b) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & -[\hat{\omega} - \hat{\beta}_\omega] & 0 & -1 & 0 & 0 & 0 & 0 & 0
0 & [R(q_n^b)g] & -[\hat{\omega} - \hat{\beta}_\omega] - \hat{\mu}M & -[\hat{v}] - kk^T & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
-0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \]

and

\[ G = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
-\hat{v} & -k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}. \]

4.6.2 Measurement Models

The accelerometer, altimeter, and visual odometry algorithm provide measurements to constrain state estimates. For each sensor, the measurement model, residual model, and residual Jacobians are defined.
4.6.2.1 Accelerometer

Because the \( z \) portion of the accelerometer measurement is used as an input to the propagation, the update step uses only the \( x \) and \( y \) components of \( \tilde{a} \), such that

\[
z_{\text{acc}} = I_{2 \times 3} \tilde{a}.
\]

The accelerometer measurement model can be derived from (4.73) and (4.68b) as

\[
h_{\text{acc}}(x, u + v) = I_{2 \times 3} (a + \beta_a + \nu_a)
\]

\[
= I_{2 \times 3} \left( \frac{T}{m} k - \frac{\mu}{m} v + \beta_a + \nu_a \right)
\]

\[
= I_{2 \times 3} \left( -\mu M \hat{v} + \beta_a + \nu_a \right), \tag{4.78}
\]

where the thrust term \( \frac{T}{m} k \) drops out because \( I_{2 \times 3} \frac{T}{m} k = 0 \). We can expand this model in terms of \( \delta x \) and \( \hat{x} \) according to (4.43) and drop second-order terms\(^4\) to obtain

\[
h_{\text{acc}}(x, u + v) = I_{2 \times 3} \left( - \left( \hat{\mu} + \delta \mu \right) M \left( \hat{v} + \delta \nu \right) + \hat{\beta}_a + \delta \beta_a + \nu_a \right)
\]

\[
\approx I_{2 \times 3} \left( -\hat{\mu} M \hat{v} - \hat{\mu} M \delta \nu - \delta \mu M \hat{v}
\]

\[
+ \hat{\beta}_a + \delta \beta_a + \nu_a \right).
\]

From (4.78), the estimated measurement is

\[
h_{\text{acc}}(\hat{x}, u) = I_{2 \times 3} \left( -\hat{\mu} M \hat{v} + \hat{\beta}_a \right).
\]

\(^4\)Because the Jacobians are evaluated at \( \delta x = v = 0 \), any second-order terms in these variables will vanish. We take advantage of this fact to simplify the derivation by dropping these second-order terms earlier.
For a given acceleration measurement $z_{\text{acc}}$, the residual is

$$r_{\text{acc}} = z_{\text{acc}} - h_{\text{acc}}(\dot{x}, u),$$

and is modeled as

$$r_{\text{acc}} = h_{\text{acc}}(x, u + v) - h_{\text{acc}}(\dot{x}, u) = I_{2 \times 3} (-\hat{\mu}M\delta v - M\dot{\nu}\delta \mu + \delta \beta_a + v_a).$$

The measurement Jacobian is therefore

$$H_{\text{acc}} = I_{2 \times 3} \begin{bmatrix} 0 & 0 & -\hat{\mu}M & 0 & I & 0 & 0 & -M\dot{\nu} \end{bmatrix}.$$

The measurement noise is the $x$ and $y$ components of the accelerometer noise,

$$R_{\text{acc}} = I_{2 \times 3} E \left[ v_a v_a^T \right] I_{2 \times 3}^T.$$

### 4.6.2.2 Altimeter

The altimeter model is for an ultrasonic range finder, which reports the nearest return in its conical field of view. As a result, the sensor reports height above ground regardless of the current attitude of the vehicle, as long as roll and pitch angles are moderate. The measurement model and its estimate are

$$h_{\text{alt}}(x) = -k^T p_n + \eta_{\text{alt}}$$

$$h_{\text{alt}}(\dot{x}) = -k^T \dot{p}_n.$$
For a given altimeter measurement $z_{alt}$, the residual is

$$r_{alt} = z_{alt} - h_{alt}(\hat{x}),$$

which is modeled as

$$r_{alt} = h_{alt}(x) - h_{alt}(\hat{x})$$

$$= -k^T p_b^n + \eta_{alt} + k^T \hat{p}_n$$

$$= -k^T \delta p_b^n + \eta_{alt}$$

resulting in the measurement Jacobian

$$H_{alt} = \begin{bmatrix} -k^T & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. $$

### 4.6.2.3 Visual Odometry Translation

Incorporating measurements from visual odometry algorithms is somewhat more involved than the previous measurement models. Figure 4.7 outlines the relationship between the
visual odometry output and the state. The visual odometry output \((p_{kc}, q_{kc})\) relates the current camera frame to the keyframe camera frame. The transform \((p_b, q_b)\) express the pose of the camera frame with respect to the vehicle’s center of mass. Commonly the camera frame used in visual odometry algorithms is derived from the image plane such that the camera frame’s z-axis references forward motion (depth). In this case, \(q_b\) encodes the mapping from the body frame’s forward-right-down coordinate frame to the coordinate frame used by the camera. The transform \((p_b, q_b)\) can be found through offline calibration or online as described in [33]. In this paper we assume that the camera is rigidly mounted to the body and \((p_b, q_b)\) is static and known. In addition, since \((p_{kc}, q_{kc})\) also represents the transform between the body and camera, but at the time of the keyframe, \(p_{kc} = p_b\) and \(q_{kc} = q_b\).

Figure 4.7 can be used to informally understand how defining a relative state improves observability. The pose \((p_n, q_n)\) encodes the roll, pitch, and altitude of the vehicle when the keyframe is declared, all of which are observable using an altimeter and IMU. This fact, in connection with assuming \((p_b, q_b)\) is known, ensures that the measurements \((p_c, q_c)\) constrain the vehicle’s current pose \((p_b, q_b)\).

Using Figure 4.7, the relative translation measurement is modeled as

\[
h_{vot}(x) = p_{kc} - R(q_{kc})p_b + R(q_{kc})R(q_n)p_n + \eta_{vot}.
\]

Dropping prescripts and recalling that \(p_{kc} = p_b\) and \(q_{kc} = q_b\), this becomes

\[
h_{vot}(x) = -R(q_b)p_b + R(q_b)R(q_n)(p_n - p_b) + \eta_{vot}.
\]
We expand this model according to (4.43) and (4.30a), then use (4.32) and (4.33) to obtain

\[
\begin{align*}
\mathbf{h}_{\text{vot}}(\mathbf{x}) &= -R(\mathbf{q}_b^c)\mathbf{p}_b^c \\
&+ R(\mathbf{q}_b^c) R(\delta\mathbf{q}^k_n) R(\hat{\mathbf{q}}_n^k) (\mathbf{p}_n^b + \delta\mathbf{p}_n^b - \hat{\mathbf{p}}_n^k - \delta\mathbf{p}_n^k) \\
&+ R(\mathbf{q}_b^c) R(\delta\mathbf{q}^k_n) R(\hat{\mathbf{q}}_n^k) R^T(\mathbf{p}_n^b) R^T(\mathbf{p}_b^c) + \eta_{\text{vot}} \\
&\approx -R(\mathbf{q}_b^c)\mathbf{p}_b^c \\
&+ R(\mathbf{q}_b^c)(\mathbf{I} - [\delta\mathbf{\theta}^k_n]) R(\hat{\mathbf{q}}_n^k) (\mathbf{p}_n^b + \delta\mathbf{p}_n^b - \hat{\mathbf{p}}_n^k - \delta\mathbf{p}_n^k) \\
&+ R(\mathbf{q}_b^c)(\mathbf{I} - [\delta\mathbf{\theta}^k_n]) R(\hat{\mathbf{q}}_n^k) R^T(\hat{\mathbf{q}}_n^k) (\mathbf{I} + [\delta\mathbf{\theta}^b_n]) \mathbf{p}_b^c \\
&+ \eta_{\text{vot}}.
\end{align*}
\]

Expanding and removing second-order terms,

\[
\begin{align*}
\mathbf{h}_{\text{vot}}(\mathbf{x}) &\approx -R(\mathbf{q}_b^c)\mathbf{p}_b^c \\
&+ R(\mathbf{q}_b^c) R(\hat{\mathbf{q}}_n^k) (\hat{\mathbf{p}}_n^b + \delta\mathbf{p}_n^b - \hat{\mathbf{p}}_n^k - \delta\mathbf{p}_n^k) \\
&- R(\mathbf{q}_b^c) [\delta\mathbf{\theta}^k_n] R(\hat{\mathbf{q}}_n^k) (\mathbf{p}_n^b - \hat{\mathbf{p}}_n^k) \\
&+ R(\mathbf{q}_b^c) R(\hat{\mathbf{q}}_n^k) R^T(\mathbf{p}_b^c) \\
&- R(\mathbf{q}_b^c) [\delta\mathbf{\theta}^k_n] R(\hat{\mathbf{q}}_n^k) R^T(\mathbf{p}_b^c) \\
&+ R(\mathbf{q}_b^c) R(\hat{\mathbf{q}}_n^k) R^T(\mathbf{p}_b^c) [\delta\mathbf{\theta}^b_n] \mathbf{p}_b^c + \eta_{\text{vot}}.
\end{align*}
\]

The estimated measurement model is

\[
\begin{align*}
\mathbf{h}_{\text{vot}}(\hat{\mathbf{x}}) &= -R(\mathbf{q}_b^c)\mathbf{p}_b^c + R(\mathbf{q}_b^c) R(\hat{\mathbf{q}}_n^k) (\hat{\mathbf{p}}_n^b - \hat{\mathbf{p}}_n^k) \\
&+ R(\mathbf{q}_b^c) R(\hat{\mathbf{q}}_n^k) R^T(\hat{\mathbf{q}}_n^k) \mathbf{p}_b^c.
\end{align*}
\]

For a given relative position measurement \(\mathbf{z}_{\text{vot}}\), the residual is

\[
\mathbf{r}_{\text{vot}} = \mathbf{z}_{\text{vot}} - \mathbf{h}_{\text{vot}}(\hat{\mathbf{x}}),
\]

114
which is modeled as

\[
\begin{align*}
\mathbf{r}_{\text{vot}} &= \mathbf{h}_{\text{vot}}(\mathbf{x}) - \mathbf{h}_{\text{vot}}(\hat{\mathbf{x}}) \\
&= R(q^c_6)R(\hat{q}^k_n)(\delta \mathbf{p}^b_n - \delta \mathbf{p}^k_n) - R(q^c_6) \left[ \delta \mathbf{\theta}^k_n \right] R(\hat{\mathbf{q}}^k_n)(\hat{\mathbf{p}}^b_n - \hat{\mathbf{p}}^k_n) \\
&\quad - R(q^c_6) \left[ \delta \mathbf{\theta}^k_n \right] R(\hat{\mathbf{q}}^k_n)R^T(\hat{q}^b_n)\mathbf{p}^b_6 + R(q^c_6)R(\hat{\mathbf{q}}^k_n)R^T(\hat{q}^b_n) \left[ \delta \mathbf{\theta}^b_n \right] \mathbf{p}^c_6 + \eta_{\text{vot}} \\
&= R(q^c_6)R(\hat{\mathbf{q}}^k_n)(\delta \mathbf{p}^b_n - \delta \mathbf{p}^k_n) + R(q^c_6) \left[ \delta \mathbf{\theta}^k_n \right] R(\hat{\mathbf{q}}^k_n)(\hat{\mathbf{p}}^b_n - \hat{\mathbf{p}}^k_n) \delta \mathbf{\theta}^k_n \\
&\quad + R(q^c_6) \left[ \delta \mathbf{\theta}^k_n \right] R(\hat{\mathbf{q}}^k_n)R^T(\hat{q}^b_n)\mathbf{p}^b_6 - R(q^c_6)R(\hat{\mathbf{q}}^k_n)R^T(\hat{q}^b_n) \left[ \mathbf{\theta}^b_n \right] \delta \mathbf{\theta}^b_n + \eta_{\text{vot}}.
\end{align*}
\]

Differentiating, we obtain the residual Jacobian

\[
\mathbf{H}_{\text{vot}} = \begin{bmatrix}
H_1 & H_2 & 0 & 0 & 0 & H_3 & H_4 & 0
\end{bmatrix},
\]

where

\[
\begin{align*}
H_1 &= R(q^c_6)R(\hat{\mathbf{q}}^k_n), \\
H_2 &= -R(q^c_6)R(\hat{\mathbf{q}}^k_n)R^T(\hat{q}^b_n) \left[ \mathbf{\theta}^b_n \right], \\
H_3 &= -R(q^c_6)R(\hat{\mathbf{q}}^k_n), \\
H_4 &= R(q^c_6) \left[ \mathbf{\theta}^k_n \right] R(\hat{\mathbf{q}}^k_n) \left( \hat{\mathbf{p}}^b_n - \hat{\mathbf{p}}^k_n + R^T(\hat{q}^b_n)\mathbf{p}^b_6 \right) \left[ \mathbf{\theta}^b_n \right].
\end{align*}
\]

### 4.6.2.4 Visual Odometry Rotation

The relative rotation measurement model also follows from Figure 4.7. Specifically,

\[
\begin{align*}
\mathbf{h}_{\text{vor}}(\mathbf{x}) &= q^c_{k_c} \\
&= (q^c_6)^{-1} \otimes (q^k_n)^{-1} \otimes q^b_n \otimes q^c_6 \otimes (\eta_{\text{vot}})^{\wedge}.
\end{align*}
\]
We expand this according to (4.29a) as

\[ h_{\text{vor}}(x) \]

\[ = (q_b^{e})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes \delta q_n^{k} \otimes q_b^{e} \otimes (\eta_\text{vor})^\wedge \]

\[ = (q_b^{e})^{-1} \otimes (\delta q_n^{k})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes \delta q_n^{k} \otimes q_b^{e} \otimes (\eta_\text{vor})^\wedge. \]

The estimated measurement model is

\[ h_{\text{vor}}(\hat{x}) = (q_b^{e})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes q_b^{e}. \]

For a given relative attitude measurement \( z_{\text{vor}} \), from (4.61) the residual is

\[ r_{\text{vor}} = \left( h_{\text{vor}}(\hat{x})^{-1} \otimes z_{\text{vor}} \right)^\vee, \]

which is modeled and then simplified using (4.40) and (4.41) as

\[ r_{\text{vor}} = \left( h_{\text{vor}}(\hat{x})^{-1} \otimes h_{\text{vor}}(x) \right)^\vee \]

\[ = \left( (q_b^{e})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes q_b^{e} \otimes (q_b^{e})^{-1} \otimes (\delta q_n^{k})^{-1} \otimes \right. \]

\[ \left. \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes \delta q_n^{k} \otimes q_b^{e} \otimes (\eta_\text{vor})^\wedge \right)^\vee \]

\[ = \left( (q_b^{e})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \dot{q}_n^{k} \otimes (\delta q_n^{k})^{-1} \otimes (\dot{q}_n^{k})^{-1} \otimes \right. \]

\[ \left. \otimes \dot{q}_n^{k} \otimes \delta q_n^{k} \otimes q_b^{e} \right)^\vee + \eta_\text{vor} \]

\[ = R(q_b^{e}) \left( -R(q_b^{e}) R^T(q_b^{e}) \delta \theta_n^{k} \right) + \Delta \theta_n^{k} + \eta_\text{vor}. \]

The measurement Jacobian is

\[ H_{\text{vor}} = \begin{bmatrix} 0, R(q_b^{e}), 0, 0, 0, -R(q_b^{e}) R^T(q_b^{e}) R^T(q_b^{e}), 0 \end{bmatrix}. \]
4.6.3 Keyframe Reset

When a new keyframe is established, the relative part of the state is reset, as described in Section 4.5. As shown in Figure 4.5b, the new node frame is positioned at zero altitude directly below the vehicle’s current true position such that the position portions of the state are reset as

\[ \mathbf{p}_n^+ = \mathbf{p}_n^k = \begin{bmatrix} 0 \\ 0 \\ p_{nz}^b \end{bmatrix} \]

and estimated as

\[ \hat{\mathbf{p}}_n^+ = \hat{\mathbf{p}}_n^k = \begin{bmatrix} 0 \\ 0 \\ \hat{p}_{nz}^b \end{bmatrix}. \] (4.80)

As such, the error state reset is

\[ \delta \mathbf{p}_n^+ = \delta \mathbf{p}_n^k = \begin{bmatrix} 0 \\ 0 \\ \delta p_{nz}^b \end{bmatrix} \]

and

\[ \frac{\partial \delta \mathbf{p}_n^+}{\partial \delta \mathbf{p}_n^b} = \frac{\partial \delta \mathbf{p}_n^k}{\partial \delta \mathbf{p}_n^b} = \mathbf{k} \mathbf{k}^\top. \]

The other vector portions of the state, including \( \mathbf{v}, \mathbf{\beta}_\omega, \mathbf{\beta}_a, \) and \( \mu \), do not change.

Resetting the yaw portion of the attitude states is slightly more complicated. Each new node frame is established such that the vehicle’s yaw is identically zero. Setting \( \psi = 0 \) in (4.37), we see that

\[ \mathbf{q}_n^+ = \mathbf{q}_n^k = \mathbf{q}_\delta \otimes \mathbf{q}_\phi \] (4.81)
where \( \hat{\phi} \) and \( \hat{\theta} \) are computed from \( \mathbf{q}_{bn}^k \) using (4.35). The covariance update for the attitude error states is governed by

\[
\mathbf{N}_\theta = \begin{bmatrix}
\sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\
0 & \cos \hat{\phi}^2 & -\cos \hat{\phi} \sin \hat{\phi} \\
0 & -\cos \hat{\phi} \sin \hat{\phi} & \sin \hat{\phi}^2
\end{bmatrix},
\]

which is derived in Appendix 4.13. When roll and pitch are approximately zero, we note that

\[
\mathbf{N}_\theta \approx \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}.
\]

In summary, the keyframe reset requires updating the state estimate according to (4.80) and (4.82) and updating the covariance according to (4.67) where

\[
\mathbf{N} = \begin{bmatrix}
\mathbf{k} \mathbf{k}^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{N}_\theta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 \\
\mathbf{k} \mathbf{k}^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{N}_\theta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.
\]
4.7 Body-fixed Relative Navigation (bRN)

While defining the vehicle’s pose with respect to an inertially defined origin uses more conventional dynamic equations and results in more intuitive state estimates, an increasing number of estimators use body-fixed or robocentric state definitions. In the body-fixed case, the position and attitude of an inertially-fixed origin is estimated with respect to the current vehicle’s pose. Body-fixed state definitions have been shown to improve filter consistency [14] and facilitate local guidance and control algorithms [65, 66]. For example, the obstacle avoidance and visual-servoing problems requires a vehicle to make navigation decisions after estimating the pose of objects with respect to its current pose.

The relative estimator in the relative navigation architecture can be formulated using a body-fixed state definition. In this case, body-fixed relative navigation (bRN) estimates the state of the node frame with respect to the current body. As before, when a new keyframe is declared, the horizontal position and heading states are reset. The principal difference is that body-fixed dynamics are used and attitude error is defined differently.

Unless explicitly specified, the equations and definitions in Section 4.6 are also assumed for bRN. This section follows the same outline as Section 4.6, first describing the input, state, and dynamics for the system in Section 4.7.1, and then defining the measurement models and keyframe reset in Sections 4.7.2 and 4.7.3 respectively.

4.7.1 State Dynamics

Body-fixed relative navigation estimates the pose of the node frame $n$ with respect to and expressed in the current body frame $b$, denoted as $(b_p^n, q^n_b)$. The states $b_p^n$ and $n_p^b$ represent the same displacement, but are pointed in opposite directions and are expressed
in different frames. Specifically,

\[ b_P^n = -R(q^n_b) n_b. \]  

(4.83)

The states \( q^n_b \) and \( q^n_k \) are inverses of each other:

\[ q^n_b = (q^n_k)^{-1}. \]  

(4.84)

The estimator also tracks the pose of frame \( n \) with respect to and expressed in frame \( k \), denoted as \((k_p^n, q^n_k)\) and shown in Figure 4.8. For bRN, the vehicle’s state is

\[ x \triangleq \begin{bmatrix} p^n_b & q^n_b & v^T & \beta^T & \beta_a^T & p^n_k & q^n_k & \mu \end{bmatrix}^T, \]

while the input remains unchanged from Section 4.6.1. The state dynamics are

\[ \dot{p}^n_b = -[\omega] p^n_b - v \]
\[ \dot{q}^n_b = -\frac{1}{2} \begin{bmatrix} \omega \\ 0 \end{bmatrix} \otimes q^n_b \]  

(4.85a)

\[ \dot{v} = [v] \omega + R^T(q^n_b) g + a_z k - \mu M v + \eta_v \]  

(4.85b)

\[ \dot{\beta}_w = \eta_{\beta_w} \]
\[ \dot{\beta}_a = \eta_{\beta_a} \]
\[ \dot{p}^n_k = 0 \]
\[ \dot{q}^n_k = 0 \]

\[ \dot{\mu} = \eta_{\mu}, \]

where derivations for the position and attitude dynamics are found in Appendix 4.12. Note that (4.85b) only differs from (4.76c) by a single transpose.
For bRN, the error state $\delta x \in \mathbb{R}^{22}$ is defined as

$$\delta x = \left[ \delta p_n^T \delta \theta_{\omega}^T \delta \nu^T \delta \beta_{\omega}^T \delta \beta_{a}^T \delta p_k^T \delta \theta_{\omega}^n T \delta \mu \right]^T.$$ 

The first-order approximation of the error-state dynamics, also derived in Appendix 4.12, is

$$\begin{align*}
\delta \dot{p}_n^n &\approx -\left[ \tilde{\omega} - \hat{\beta}_{\omega} \right] \delta p_n^n - \delta \nu - [\hat{\nu}^n] \delta \beta_{\omega} - [\hat{\nu}^n] \nu_{\omega} \\
\delta \dot{\theta}_b^n &\approx -\left[ \tilde{\omega} - \hat{\beta}_{\omega} \right] \delta \theta_b^n + \delta \beta_{\omega} + \nu_{\omega} \\
\delta \dot{\nu} &\approx -\left[ R^T(\hat{q}_b^n)g \right] \delta \theta_b^n + \left( -\left[ \tilde{\omega} - \hat{\beta}_{\omega} \right] - \hat{\mu}_M \right) \delta \nu \\
&\quad - [\hat{\nu}] \delta \beta_{\omega} - kk^T \delta \beta_{a} - M \dot{\nu} \delta \mu - [\hat{\nu}] \nu_{\omega} \\
&\quad - k \eta_{a_z} + \eta_{\nu} \\
\delta \dot{\beta}_{\omega} &= \eta_{\beta_{\omega}} \\
\delta \dot{\beta}_{a} &= \eta_{\beta_{a}} \\
\delta \dot{p}_k^n &= 0 \\
\delta \dot{\theta}_k^n &= 0 \\
\delta \dot{\mu} &= \eta_{\mu}.
\end{align*}$$

Differentiating the error-state dynamics with respect to the error state and input noise results in the following propagation Jacobians:

$$F = \begin{bmatrix}
-|\tilde{\omega}| & 0 & -I & -[\hat{\nu}^n] & 0 & 0 & 0 \\
0 & -|\tilde{\omega}| & 0 & I & 0 & 0 & 0 \\
0 & -[R^T(\hat{q}_b^n)g] & -|\tilde{\omega}| - \hat{\mu}_M & -[\hat{\nu}] & -kk^T & 0 & 0 & 0 & -M \hat{\nu} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

and

$$G = \begin{bmatrix}
-\hat{\nu}^n \\
1 \\
-k \hat{\nu} \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix},$$

where $\tilde{\omega}$ is defined in (4.69).
4.7.2 Measurement Models

The measurement models differ when using a body-fixed parameterization as outlined below.

4.7.2.1 Accelerometer

Because the accelerometer model is independent of attitude and position, the model remains unchanged for bRN. See Section 4.6.2.1.

4.7.2.2 Altimeter

As described in Section 4.6.2.2, the sonar altimeter measures height-above-ground. By relating (4.83) and (4.79), the vehicle’s altitude is

\[ h_{\text{alt}}(x) = k^T R(q_b^n)p_b^n + \eta_{\text{alt}}, \]

where the rotation expresses the height in an inertial frame. The estimated measurement model is

\[ h_{\text{alt}}(\hat{x}) = k^T R(\hat{q}_b^n)\hat{p}_b^n. \]

For a given altimeter measurement \( z_{\text{alt}} \), the residual is

\[ r_{\text{alt}} = z_{\text{alt}} - h_{\text{alt}}(\hat{x}) \]

which is approximated using (4.30b), (4.43a), (4.32), and (4.4) as

\[ r_{\text{alt}} = h_{\text{alt}}(x) - h_{\text{alt}}(\hat{x}) \]

\[ = k^T R(q_b^n)p_b^n + \eta_{\text{alt}} - k^T R(\hat{q}_b^n)\hat{p}_b^n \]

\[ = k^T R(\hat{q}_b^n)R(\delta q_b^n)(\hat{p}_b^n + \delta p_b^n) + \eta_{\text{alt}} - k^T R(\hat{q}_b^n)\hat{p}_b^n \]

\[ = k^T R(\hat{q}_b^n)\delta q_b^n(\hat{p}_b^n + \delta p_b^n) + \eta_{\text{alt}}. \]
\[ \approx k^T R(\hat{q}_b^n) (I - [\delta \theta_b^n]) (\hat{p}_b^n + \delta p_b^n) + \eta_{alt} - k^T R(\hat{q}_b^n) \hat{p}_b^n \]
\[ \approx k^T R(\hat{q}_b^n) \delta p_b^n - k^T R(\hat{q}_b^n) [\delta \theta_b^n] \hat{p}_b^n + \eta_{alt} \]
\[ = k^T R(\hat{q}_b^n) \delta p_b^n + k^T R(\hat{q}_b^n) [\hat{p}_b^n] \delta \theta_b^n + \eta_{alt} , \]
resulting in the measurement Jacobian

\[ H_{alt} = \begin{bmatrix} k^T R(\hat{q}_b^n), k^T R(\hat{q}_b^n) \hat{p}_b^n \end{bmatrix}, 0, 0, 0, 0, 0 . \]

### 4.7.2.3 Visual Odometry Position

Figure 4.8 presents the relative position and attitude measurement model for bRN. Note that \((p_b^n, q_b^n), (p_k^n, q_k^n),\) and \((p_c^{kc}, q_c^{kc})\) are reversed when compared to Figure 4.7. A relative measurement can generally be measured either direction. By having both the state and measurement pointing back to the node frame, the measurement model has fewer terms, reducing any unnecessary coupling.

From Figure 4.8, the relative position update is

\[ h_{vot}(x) = c_p^{kc} + \eta_{vot} \]
\[ = -c_p^c + c_p^n + c_p^n + c_p^{kc} + \eta_{vot} \]
\begin{align*}
= & -R(q_b^c)b p_b^c + R(q_b^c)b p_b^n \\
& - R(q_b^c)R^T(q_b^n)R(q_k^n) k p_k^n \\
& + R(q_b^c)R^T(q_b^n)R(q_k^n) k p_k^{kc} + \eta_{vot} .
\end{align*}

Dropping prescripts and recalling that \( p_k^{kc} = p_b^c \) and \( q_k^{kc} = q_b^c \), this becomes

\begin{align*}
h_{vot}(x) & = R(q_b^c)(p_b^n - p_b^c) \\
& + R(q_b^c)R^T(q_b^n)R(q_k^n)(p_b^c - p_k^n) + \eta_{vot} .
\end{align*}

Expanding according to (4.43), (4.30b), (4.32), and (4.33) gives

\begin{align*}
h_{vot}(x) = & R(q_b^c)(\hat{p}_b^n + \delta p_b^n - p_b^c) \\
& + R(q_b^c)R^T(\delta q_b^n)R(q_b^n)R(\hat{q}_k^n)R(\delta q_k^n)(p_b^c - \hat{p}_k^n - \delta p_k^n) + \eta_{vot} \\
= & R(q_b^c)(\hat{p}_b^n + \delta p_b^n - p_b^c) \\
& + R(q_b^c)(I + [\delta \theta_b^n])R^T(\hat{q}_b^n)R(\hat{q}_k^n)(I - [\delta \theta_k^n])(p_b^c - \hat{p}_k^n - \delta p_k^n) + \eta_{vot} .
\end{align*}

Expanding and removing higher-order terms,

\begin{align*}
h_{vot}(x) \approx & R(q_b^c)(\hat{p}_b^n + \delta p_b^n - p_b^c) \\
& + R(q_b^c)R^T(\hat{q}_b^n)R(\hat{q}_k^n)(p_b^c - \hat{p}_k^n - \delta p_k^n) \\
& + R(q_b^c)[\delta \theta_b^n]R^T(\hat{q}_b^n)R(\hat{q}_k^n)(p_b^c - \hat{p}_k^n) \\
& - R(q_b^c)R^T(\hat{q}_b^n)R(\hat{q}_k^n)[\delta \theta_k^n](p_b^c - \hat{p}_k^n) + \eta_{vot} .
\end{align*}

The estimated measurement model is

\begin{align*}
h_{vot}(\hat{x}) = & R(q_b^c)(\hat{p}_b^n - p_b^c) \\
& + R(q_b^c)R^T(\hat{q}_b^n)R(\hat{q}_k^n)(p_b^c - \hat{p}_k^n) .
\end{align*}
We then compute the residual according to (4.60) as

\[ r_{\text{vot}} = h_{\text{vot}}(x) - h_{\text{vot}}(\hat{x}) \]

\[ = R(q^c_b)\delta p^n_b - R(q^c_b)R^T(\hat{q}^n_b)R(\hat{q}^n_k)\delta p^n_k \]

\[ + R(q^c_b)\left[ \delta \theta^n_b \right] R^T(\hat{q}^n_b)R(\hat{q}^n_k) (p^c_b - \hat{p}^n_b) \]

\[ - R(q^c_b)R^T(\hat{q}^n_b)R(\hat{q}^n_k) \left[ \delta \theta^n_k \right] (p^c_b - \hat{p}^n_b) + \eta_{\text{vot}} \]

\[ = R(q^c_b)\delta p^n_b - R(q^c_b) \left[ R^T(q^c_b)R(q^c_k) (p^c_b - \hat{p}^n_k) \right] \delta \theta^n_b \]

\[ - R(q^c_b)R^T(q^c_b)R(q^c_k) \delta p^n_k \]

\[ + R(q^c_b)R^T(q^c_b)R(q^c_k) \left[ p^c_b - \hat{p}^n_k \right] \delta \theta^n_k + \eta_{\text{vot}}, \]

resulting in the residual Jacobian

\[ H_{\text{vot}} = \begin{bmatrix} H_1 & H_2 & 0 & 0 & 0 & H_3 & H_4 & 0 \end{bmatrix} \]

where

\[ H_1 = R(q^c_b), \]

\[ H_2 = -R(q^c_b) \left[ R^T(q^c_b)R(q^c_k) (p^c_b - \hat{p}^n_k) \right], \]

\[ H_3 = -R(q^c_b)R^T(q^c_b)R(q^c_k), \]

\[ H_4 = R(q^c_b)R^T(q^c_b)R(q^c_k) \left[ p^c_b - \hat{p}^n_k \right]. \]

### 4.7.2.4 Visual Odometry Rotation

The relative rotation measurement model also follows from Figure 4.8. Specifically,

\[ h_{\text{vor}}(x) = q^{kc} \]
\[
= (\eta_{vor})^\vee \otimes (\hat{q}_b)^{-1} \otimes q^n_b \otimes (\hat{q}^n_k)^{-1} \otimes q^c_b
\]
\[
= (\eta_{vor})^\vee \otimes (\hat{q}_b)^{-1} \otimes \delta q^n_b \otimes \hat{q}^n_b \otimes (\hat{q}^n_k)^{-1} \otimes q^c_b
\]

The estimated measurement model is
\[
h_{vor}(\hat{x}) = (q^n_b)^{-1} \otimes \hat{q}^n_b \otimes (\hat{q}^n_k)^{-1} \otimes q^c_b.
\]

For a given relative attitude measurement \( z_{vor} \), the residual is
\[
r_{vor} = \left( z_{vor} \otimes h_{vor}(\hat{x})^{-1} \right)^\vee,
\]

which is modeled and then simplified using (4.40) and (4.41) as
\[
r_{vor} = \left( h_{vor}(x) \otimes h_{vor}(\hat{x})^{-1} \right)^\vee
\]
\[
= \left( (\eta_{vor})^\vee \otimes (q^n_b)^{-1} \otimes \delta q^n_b \otimes \hat{q}^n_b \otimes (\hat{q}^n_k)^{-1} \otimes q^c_b \right)^\vee
\]
\[
= \left( (\eta_{vor})^\vee \otimes (q^n_b)^{-1} \otimes \delta q^n_b \otimes \hat{q}^n_b \otimes (\hat{q}^n_k)^{-1} \otimes q^c_b \right)^\vee
\]
\[
= R(q^n_b) \left( \delta \theta^n_b - R^T(q^n_b) R(q^c_b) \delta \theta^n_k \right) + \eta_{vor}
\]
\[
= R(q^n_b) \delta \theta^n_b - R(q^n_b) R^T(q^n_b) R(q^c_b) \delta \theta^n_k + \eta_{vor},
\]

resulting in the measurement Jacobian
\[
H_{vor} = \left[ 0, R(q^n_b), 0, 0, 0, 0, -R(q^n_b) R^T(q^n_b) R(q^c_b), 0 \right].
\]
4.7.3 Keyframe Reset

The keyframe reset step for bRN is somewhat less intuitive due to the body-centric representation of position and orientation states, but can be derived by following the iRN reset step. Following (4.84) and (4.81),

\[
\hat{q}_b^{n+} = (\hat{q}_b^{n})^{-1} = (\hat{q}_\theta \otimes \hat{q}_\phi)^{-1} = \\
\begin{bmatrix}
- \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\
- \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\
\sin \frac{\theta}{2} \sin \frac{\phi}{2} \\
\cos \frac{\theta}{2} \cos \frac{\phi}{2}
\end{bmatrix},
\]

where \(\phi\) and \(\theta\) are computed from \(\hat{q}_b^{n} = (\hat{q}_b^{n})^{-1}\) using (4.35). Note that even though bRN expresses attitude in the body frame, the angles \(\phi\), \(\theta\), and \(\psi\) continue to represent conventional roll, pitch, and yaw Euler angles which express the orientation of a body with respect to an inertial frame. The Jacobian for the attitude reset is derived in Appendix 4.13, and happens to be identical to that for the iRN attitude reset:

\[
\frac{\partial \delta \theta_b^{n+}}{\partial \delta \theta_b^{n}} = \frac{\partial \delta \theta_b^{n+}}{\partial \delta \theta_b^{n}} = N_\theta.
\]

The derivation for the position reset is more involved, and is given in Appendix 4.13. The resulting reset operation is

\[
\hat{p}_b^{n+} = N_p \hat{p}_b^{n},
\]

where

\[
N_p = R^T(\hat{q}_b^{n+}) k k^T R(\hat{q}_b^{n+}).
\]
Because frames $b$ and $k$ are at the same location when the reset occurs, we also have

$$\hat{p}_{kn}^{n+} = \hat{p}_{bn}^{n+},$$

$$\hat{q}_{kn}^{n+} = \hat{q}_{bn}^{n+}.$$

The total keyframe reset Jacobian, also derived in Appendix 4.13, is given by

$$N = \begin{bmatrix}
N_p & ( - [N_p p_{bn}^n] + N_p [p_{bn}^n] ) N_\theta & 0 & 0 & 0 & 0 & 0 \\
0 & N_\theta & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & I \\
\end{bmatrix}.
$$

4.8 Results

The RMEKF was implemented in C++ and validated for a multirotor vehicle in simulation using the Gazebo/ROS environment. This paper presents a brief discussion of simulation results to illustrate typical RMEKF performance. While [8] thoroughly compares relative navigation to other estimation frameworks, the performance comparison of the full-state RMEKF to other estimators is left as future work.

The simulation provided accelerometer, gyro, altimeter, and visual odometry measurements corrupted by normally-distributed noise. Slowly-drifting biases were also added to the simulated IMU data. Sensor noise ($\nu_\omega$, $\nu_a$, $\eta_{vot}$, $\eta_{vor}$, and $\eta_{alt}$) was sampled from normal distributions with the following standard deviations that are typical of low-cost hardware.
sensors: $\sigma_\omega = 0.13\text{rad/s}$, $\sigma_a = 1.15\text{m/s}^2$, $\sigma_{\text{vol}} = 0.02\text{m}$, $\sigma_{\text{vor}} = 0.01\text{rad}$, and $\sigma_{\text{alt}} = 0.01\text{m}$.

New keyframes were established when the vehicle moved more than 0.2 m or yawed more than 20 degrees. The estimator was evaluated during various maneuvers ranging from reserved to aggressive, where during the aggressive maneuvers the vehicle’s speed exceeded 25 m/s and the bank angle exceeded 45 degrees.

The RMEKF successfully tracked truth throughout the simulated trajectories. Figures 4.9 and 4.10 show a three-second snapshot of the performance of the iRN and bRN estimator at tracking the vehicle’s pose, where the small time window was selected to make the relative state reset visible. During these three seconds, the vehicle was moving forward at nearly-constant velocity while maintaining a nominal height above ground of 1.25 m, and while gradually slowing its clockwise yaw motion. The vertical gray lines indicate the time when a new keyframe is declared. While the state is defined using quaternions, Figure 4.9b uses (4.35) to plot roll, pitch, and yaw angles.

Figure 4.9 highlights several interesting practicalities of relative navigation. As discussed in Section 4.5, Figure 4.9 illustrates how the forward, right, and yaw states are reset to zero with each newly declared keyframe, while the altitude, roll, and pitch states remain continuous. It should be noted that while roll and pitch are continuous, discontinuities appear in each of the four quaternion states. Because the discontinuities in the relative states occur at known times, they are easily accounted for and so in practice do not cause problems with control stability. Figure 4.9 also illustrates that keyframes do not reset at fixed intervals, but rather reset based on how far the vehicle has traveled since the previous keyframe. Certain sensors, such as a laser scanner with a long range and wide field of view, facilitate longer distances between keyframes.

The RMEKF performed very similarly when using the body-fixed dynamics presented in Section 4.7. Figure 4.10 shows the performance of the bRN estimator for position states
Figure 4.9: RMEKF estimation performance for inertial relative navigation (iRN). The vertical gray lines indicate when a new keyframe is declared.
over the same window of time as Figure 4.9. The attitude performance of bRN (not shown) is nearly identical to the attitude performance of iRN with the exception of a change in sign for the angles. Note in Figure 4.10 that each of the position states experience slight discontinuities at the keyframe reset and that the position estimates do not reset to zero. This is because the keyframe reset step removes all horizontal translation from the state but continues to track the vehicle’s height above ground. Since the bRN position state is expressed with respect to the rolled and pitched body frame, some of the height above ground is mapped into the forward and right components. When the bRN state estimates are expressed with respect to the current node frame using (4.83) and (4.84), they are nearly identical to the estimates shown in Figure 4.9.

The RMEKF has also undergone significant validation in hardware. A detailed discussion of the hardware implementation and results are presented in [32]. To summarize those results, as demonstrated in a number of prolonged GPS-denied flights, a variety of vision sensors were successfully incorporated into the RMEKF to produce smooth, accurate, and consistent relative state estimates that enabled robust UAS navigation.
4.9 Conclusions

While the global state of a UAS is not observable when navigating through GPS-denied environments, it is possible to maintain observability by subtly restructuring the problem using the relative navigation framework. Previous simulation studies show strong theoretical motivations for the relative navigation framework, including improved consistency, bounded covariance, and improved accuracy [8]. In addition, significant hardware results have demonstrated the efficacy and practicality of relative navigation for small UAS when using a relative MEKF for state estimation [32]. This paper rigorously derives the RMEKF used in [32], presenting the mathematics necessary to apply relative navigation to the UAS state estimation problem. Its tutorial nature and step-by-step derivations make this paper a self-contained resource for extending the approach to other applications.

By defining the state with respect to a local coordinate frame, this paper demonstrates how to leverage relative measurements from a visual odometry algorithm to ensure an observable state. New visual odometry measurement models are proposed and a unique keyframe reset step is presented to ensure filter states are fully observable even when global information is not available. The RMEKF is demonstrated in simulation to work effectively for both inertially-defined and body-fixed vehicle dynamics to produce accurate state estimates with bounded uncertainty. While the inertial and body-fixed definitions yield similar results, this paper explicitly outlines the differences that arise in the state estimator, including differences in the error state definition, measurement models, quaternion integration procedure, and keyframe reset step. Additional contributions of this paper include a tutorial introduction to indirect multiplicative extended Kalman filtering, an exposition of Hamilton quaternions, and the derivation of several novel properties of error quaternions necessary for partially updating a quaternion and its covariance.
Funding

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547 along with significant contributions from C-UAS industry members. This work was also supported in part by Air Force Research Laboratory Science and Technology (AFRL S&T) sponsorship. Daniel Koch was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

4.10 Zero-Mean Error State

In this appendix we show that the expected error state remains zero mean. Specifically, we show that during the propagation step the error-state dynamics are trivial, as indicated in (4.44), and we show that when the state estimate $\hat{x}$ is updated according to (4.64) during the update step, the error state remains zero mean.

Properties (4.44) stems from the linearity of the expectation and quaternion multiplication operations. When $\hat{x} = E[x]$ and given (4.43a),

\[
E[\delta x_v] = E[x_v - \hat{x}_v]
\]
\[
= E[x_v] - \hat{x}_v
\]
\[
= 0
\]

and

\[
E[\dot{x}_v] = E[\dot{x}_v - \hat{x}_v]
\]
\[
= E[\dot{x}_v] - \hat{x}_v
\]
\[
= 0.
\]
Using (4.29a) as the attitude error definition for (4.43b),

\[
E[\delta x_\theta] = E \left[ (x_q \otimes \hat{x}_q^{-1})^\vee \right] \\
= (E [x_q] \otimes \hat{x}_q^{-1})^\vee \\
= (\hat{x}_q \otimes \hat{x}_q^{-1})^\vee \\
= \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\vee \\
= 0.
\]

For attitude error dynamics we incorporate (4.25) to see that

\[
E[\delta \dot{x}_\theta] = E \left[ \frac{d}{dt} \left( x_q \otimes \hat{x}_q^{-1} \right)^\vee \right] \\
= E \left[ \dot{x}_q \otimes \hat{x}_q^{-1} + x_q \otimes \hat{x}_q^{-1} \right]^\vee \\
= (E [\dot{x}_q] \otimes \hat{x}_q^{-1} + E [x_q] \otimes \hat{x}_q^{-1})^\vee \\
= \left( \frac{1}{2} \hat{x}_q \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \otimes \hat{x}_q^{-1} - \frac{1}{2} \hat{x}_q \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \otimes \hat{x}_q^{-1} \right)^\vee \\
= 0.
\]

A similar derivation is also possible for the attitude dynamics definition in (4.26).

We similarly establish (4.64) by exploring vector and quaternion states in turn. For
vector states we see the expected value of the error state after applying (4.64a) is

\[
E \left[ x_v - x_{\hat{v}} \right] = E \left[ x_v - (\hat{x}_v + \Delta v) \right] \\
= E \left[ (x_v - \hat{x}_v) - \Delta v \right] \\
= E \left[ \delta x_{\hat{v}} \right] - \Delta v
\]
\[ \Delta v \times \Delta v = 0. \]

When quaternion error is defined according to (4.29a) as for iRN, the expected value of the error state after applying (4.64b) is

\[
E \left[ (\hat{x}_q^+)^{-1} \otimes x_q \right]^\vee = E \left[ (\hat{x}_q \otimes \Delta \theta^\wedge)^{-1} \otimes x_q \right]^\vee
= E \left[ (\Delta \theta^\wedge)^{-1} \otimes \hat{x}_q^{-1} \otimes x_q \right]^\vee
= ( (\Delta \theta^\wedge)^{-1} \otimes E \left[ \delta x_q^+ \right] )^\vee
= ( (\Delta \theta^\wedge)^{-1} \otimes \Delta \theta^\wedge )^\vee
= 0.
\]

When quaternion error is defined according to (4.29b) as for bRN, the expected value of the error state after applying (4.64c) is

\[
E \left[ x_q \otimes (\hat{x}_q^+)^{-1} \right]^\vee
= E \left[ x_q \otimes (\Delta \theta^\wedge \otimes \hat{x}_q)^{-1} \right]^\vee
= ( E \left[ x_q \otimes \hat{x}_q^{-1} \right] \otimes (\Delta \theta^\wedge)^{-1} )^\vee
= ( E \left[ \delta x_q^+ \right] \otimes (\Delta \theta^\wedge)^{-1} )^\vee
= (\Delta \theta^\wedge \otimes (\Delta \theta^\wedge)^{-1} )^\vee
= 0.
\]

4.11 Inertial Dynamics

This appendix derives the error-state dynamics for an inertially-defined state.
4.11.1 Position

We begin from (4.43a):

\[ \delta \dot{p}_b = \dot{p}_n - \dot{p}_b \]

\[ = R^T(q_n) v - R^T(q_b) \dot{v} . \]

We then use (4.30a), (4.33), (4.43a), and (4.4) and drop second-order terms to obtain

\[ \delta \dot{p}_b = R^T(\dot{q}_n) R^T(\delta q_n) (\dot{v} + \delta \dot{v}) - R^T(\dot{q}_b) \dot{v} \]

\[ \approx R^T(\dot{q}_n) \left( I + \left[ \delta \theta_n \right] \right) (\dot{v} + \delta \dot{v}) - R^T(\dot{q}_b) \dot{v} \]

\[ = R^T(\dot{q}_n) \dot{v} + R^T(\dot{q}_n) \left[ \delta \theta_n \right] \dot{v} + R^T(\dot{q}_b) \delta v \]

\[ + R^T(\dot{q}_b) \left[ \delta \theta_n \right] \delta v - R^T(\dot{q}_n) \dot{v} \]

\[ = R^T(\dot{q}_n) \left[ \delta \theta_n \right] \dot{v} + R^T(\dot{q}_n) \delta v + R^T(\dot{q}_b) \left[ \delta \theta_n \right] \delta v \]

\[ \approx - R^T(\dot{q}_n) \left[ \delta \theta_n \right] \dot{v} + R^T(\dot{q}_b) \delta v. \]

4.11.2 Attitude

We begin with (4.28a) and differentiate with respect to time:

\[ q_n = q_n \otimes \delta q_n \]

\[ \implies q_n = \dot{q}_n \otimes \delta q_n + \ddot{q}_n \otimes \delta q_n. \]

Multiplying on the left by \((\dot{q}_n)^{-1}\) gives

\[ (\dot{q}_n)^{-1} \otimes q_n = (\dot{q}_n)^{-1} \otimes \dot{q}_n \otimes \delta q_n + \delta q_n \]

\[ \implies \delta q_n = (\dot{q}_n)^{-1} \otimes \dot{q}_n - (\dot{q}_n)^{-1} \otimes \dot{q}_n \otimes \delta q_n. \]
Using (4.76b), (4.77a), (4.29a) and simplifying yields

\[
\delta \dot{q}^b_n = \frac{1}{2}(\dot{q}^b_n)^{-1} \otimes q^b_n \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} - \frac{1}{2}(\dot{q}^b_n)^{-1} \otimes \dot{q}^b_n \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \otimes \delta q^b_n
\]

\[
= \frac{1}{2} \delta q^b_n \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \otimes \delta q^b_n.
\]

(4.86)

Using (4.68a) and (4.69), let

\[
\delta \omega \triangleq \omega - \hat{\omega}
\]

(4.87a)

\[
= (\dot{\omega} - \beta_\omega - \nu_\omega) - (\dot{\hat{\omega}} - \hat{\beta}_\omega)
\]

\[
= - (\hat{\beta}_\omega + \delta \beta_\omega) - \nu_\omega + \hat{\beta}_\omega
\]

\[
= - \delta \beta_\omega - \nu_\omega.
\]

(4.87b)

Applying (4.3) and (4.87a) to (4.86) yields

\[
\delta \dot{q}^b_n = \frac{1}{2} \begin{bmatrix} - |\omega| & \omega \\ -\omega^T & 0 \end{bmatrix} \delta q^b_n - \frac{1}{2} \begin{bmatrix} |\hat{\omega}| & \hat{\omega} \\ -\hat{\omega}^T & 0 \end{bmatrix} \delta q^b_n
\]

\[
= \frac{1}{2} \begin{bmatrix} -2\dot{\omega} + \delta \omega & \delta \omega \\ -\delta \omega^T & 0 \end{bmatrix} \delta q^b_n,
\]

which implies that

\[
\begin{bmatrix} \frac{1}{2} \delta \dot{\theta}^b_n \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -2\dot{\omega} + \delta \omega & \delta \omega \\ -\delta \omega^T & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \delta \theta^b_n \\ 1 \end{bmatrix}.
\]

Dropping the scalar equation and second-order terms yields

\[
\delta \dot{\theta}^b_n = \frac{1}{2} \begin{bmatrix} -2\dot{\omega} + \delta \omega \end{bmatrix} \delta \theta^b_n + \delta \omega
\]

\[
\approx - |\dot{\omega}| \delta \theta^b_n + \delta \omega
\]

\[
\approx - (\dot{\omega} - \hat{\beta}_\omega) \delta \theta^b_n - \delta \beta_\omega - \nu_\omega.
\]
4.11.3 Velocity

We begin by applying (4.43a) to the dynamics:

\[
\delta \dot{v} = [v] \omega + R(q^n_b)g + a_z k - \mu M v + \eta_v
\]

\[
- \left( [\dot{v}] \hat{\omega} + R(\hat{q}_n^b)g + \hat{a}_z k - \hat{\mu} M \hat{v} \right)
\]

\[
= [v] \omega - [\dot{v}] \hat{\omega} \quad (4.88a)
\]

\[
+ R(q^n_b)g - R(\hat{q}_n^b)g \quad (4.88b)
\]

\[
- \mu M v + \hat{\mu} M \hat{v} \quad (4.88c)
\]

\[
+ a_z k - \hat{a}_z k \quad (4.88d)
\]

\[
+ \eta_v .
\]

We will simplify (4.88) one term at a time. We expand term (4.88a) using (4.43a) and (4.87a), drop second-order terms, then use (4.4), (4.69), and (4.87b) to obtain

\[
[v] \omega - [\dot{v}] \hat{\omega} \approx [\dot{v}] + \delta v (\hat{\omega} + \delta \omega) - [\dot{v}] (\hat{\omega})
\]

\[
= [\delta v] \hat{\omega} + [\dot{v}] \delta \omega
\]

\[
= -[\hat{\omega}] \delta v + [\dot{v}] \delta \omega
\]

\[
= -[\hat{\omega} - \hat{\beta}_\omega] \delta v + [\dot{v}] (-\delta \beta_\omega - v_\omega)
\]

\[
= -[\hat{\omega} - \hat{\beta}_\omega] \delta v - [\dot{v}] \delta \beta_\omega - [\dot{v}] v_\omega .
\]

For term (4.88b) we use (4.30a) and (4.32) to obtain

\[
R(q^n_b)g - R(\hat{q}_n^b)g = R(\delta q^n_b)R(\hat{q}_n^b)g - R(\hat{q}_n^b)g
\]

\[
\approx \left( I - [\delta \theta^n] \right) R(\hat{q}_n^b)g - R(\hat{q}_n^b)g
\]
\[ = - \left[ \delta \theta^b_n \right] \mathbf{R}(\hat{q}_n^b) \mathbf{g} \]
\[ = \left[ \mathbf{R}(\hat{q}_n^b) \mathbf{g} \right] \delta \theta^b_n . \]

We use (4.43a) and drop second-order terms to simplify term (4.88c) as

\[-\mu \mathbf{M} \mathbf{v} + \hat{\mu} \mathbf{M} \hat{\mathbf{v}} = - (\hat{\mu} + \delta \mu) \mathbf{M} (\hat{\mathbf{v}} + \delta \mathbf{v}) + \hat{\mu} \mathbf{M} \hat{\mathbf{v}} \]
\[ = -\hat{\mu} \mathbf{M} \delta \mathbf{v} - \delta \mu \mathbf{M} \hat{\mathbf{v}} - \delta \mu \mathbf{M} \delta \mathbf{v} \]
\[ = -\hat{\mu} \mathbf{M} \delta \mathbf{v} - \delta \mu \mathbf{M} \hat{\mathbf{v}} . \]

Finally term (4.88d) is simplified using (4.68b), (4.70), and (4.43a) as

\[ a_z \mathbf{k} - \hat{a}_z \mathbf{k} = (\hat{a}_z - \beta a_z - \eta a_z) \mathbf{k} - (\hat{a}_z - \hat{\beta} a_z) \mathbf{k} \]
\[ = (-\delta \beta a_z - \eta a_z) \mathbf{k} \]
\[ = -\mathbf{k} \mathbf{k}^T \delta \beta a - \mathbf{k} \eta a_z . \]

Substituting all of these results into (4.88) and gathering terms gives

\[ \delta \mathbf{v} = \left[ \mathbf{R}(\hat{q}_n^b) \mathbf{g} \right] \delta \theta^b_n + \left( - [\mathbf{\hat{\omega}} - \hat{\beta} \omega] - \hat{\mu} \mathbf{M} \right) \delta \mathbf{v} - \mathbf{k} \mathbf{k}^T \delta \beta a \\
- [\hat{\mathbf{v}}] \delta \beta \omega - \mathbf{M} \hat{\mathbf{v}} \delta \mu - [\hat{\mathbf{v}}] \mathbf{v} - \mathbf{M} \hat{\mathbf{v}} \mathbf{v} - \mathbf{k} \eta a_z + \eta v . \]

4.12 Body-Fixed Dynamics

This appendix derives the body-fixed state dynamics from their inertial counterparts, and then derives the body-fixed error-state dynamics.
4.12.1 Position

We represent (4.83) as

\[
\begin{bmatrix}
\mathbf{p}_b^n \\
0
\end{bmatrix} = -(\mathbf{q}_n^b)^{-1} \otimes \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b.
\]

Left-multiplying by \( \mathbf{q}_n^b \), taking the time derivative, and left-multiplying by \((\mathbf{q}_n^b)^{-1}\) results in the following sequence of equations:

\[
\begin{align*}
\mathbf{q}_n^b \otimes \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} &= - \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b \\
\mathbf{q}_n^b \otimes \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} + \mathbf{q}_n^b \otimes \begin{bmatrix}
\dot{\mathbf{p}}_b^n \\
0
\end{bmatrix} &= - \begin{bmatrix}
\dot{\mathbf{p}}_b^n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b - \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} \otimes \dot{\mathbf{q}}_n^b \\
\frac{1}{2} \mathbf{q}_n^b \otimes \begin{bmatrix}
\omega \\
0
\end{bmatrix} \otimes \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} + \frac{1}{2} \mathbf{q}_n^b \otimes \begin{bmatrix}
\dot{p}_b^n \\
0
\end{bmatrix} &= -(\mathbf{q}_n^b)^{-1} \otimes \begin{bmatrix}
\dot{\mathbf{p}}_b^n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b - \frac{1}{2} (\mathbf{q}_n^b)^{-1} \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b \otimes \begin{bmatrix}
\omega \\
0
\end{bmatrix}.
\end{align*}
\]

Rearranging and using (4.83), (4.7), (4.76a), (4.3), and (4.4) gives

\[
\begin{align*}
\begin{bmatrix}
\dot{\mathbf{p}}_b^n \\
0
\end{bmatrix} &= -(\mathbf{q}_n^b)^{-1} \otimes \begin{bmatrix}
\dot{\mathbf{p}}_b^n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b - \frac{1}{2} (\mathbf{q}_n^b)^{-1} \begin{bmatrix}
\mathbf{p}_n \\
0
\end{bmatrix} \otimes \mathbf{q}_n^b \otimes \begin{bmatrix}
\omega \\
0
\end{bmatrix} - \frac{1}{2} \begin{bmatrix}
\omega \\
0
\end{bmatrix} \otimes \begin{bmatrix}
\mathbf{p}_b^n \\
0
\end{bmatrix} \\
&= -(\mathbf{R}(\mathbf{q}_n^b)\mathbf{p}_b^n)^\top + \frac{1}{2} \begin{bmatrix}
\mathbf{b}^\top \\
0
\end{bmatrix} \otimes \begin{bmatrix}
\omega \\
0
\end{bmatrix} - \frac{1}{2} \begin{bmatrix}
\mathbf{b}^\top \\
0
\end{bmatrix} \otimes \begin{bmatrix}
\mathbf{b}^\top \\
0
\end{bmatrix} \\
&= -\begin{bmatrix}
\mathbf{v} \\
0
\end{bmatrix} + \frac{1}{2} \begin{bmatrix}
\mathbf{b}^\top \omega \\
-\mathbf{b}^\top \omega
\end{bmatrix} - \frac{1}{2} \begin{bmatrix}
\omega \mathbf{b}^\top \\
-\omega \mathbf{b}^\top
\end{bmatrix} \\
&= -\begin{bmatrix}
\mathbf{v} \\
0
\end{bmatrix} - \frac{1}{2} \begin{bmatrix}
\mathbf{b}^\top \omega \\
\mathbf{b}^\top \omega
\end{bmatrix} - \frac{1}{2} \begin{bmatrix}
\omega \mathbf{b}^\top \\
-\omega \mathbf{b}^\top
\end{bmatrix}.
\end{align*}
\]
Dropping the scalar equation we see that

\[ \dot{p}_b^n = -v - [\omega] p^n_b. \]

Letting \( \delta \omega = \omega - \hat{\omega} \), the error-state position dynamics when using the body frame are

\[ \delta \dot{p}_b^n = \dot{p}_b^n - \dot{\hat{p}}_b^n \]

\[ = (-v - [\omega] p^n_b) - (-\dot{\hat{v}} - [\hat{\omega}] \dot{\hat{p}}_b^n) \]

\[ = -\dot{\hat{v}} - \delta v - [\hat{\omega} + \delta \omega] \dot{\hat{p}}_b^n - [\hat{\omega} + \delta \omega] \delta p_b^n - \dot{\hat{v}} + [\hat{\omega}] \dot{\hat{p}}_b^n \]

\[ = -\delta v - [\delta \omega] \delta p_b^n - [\hat{\omega}] \delta p_b^n \]

\[ = -\delta v - [\hat{p}_b^n] \delta \omega - [\hat{\omega}] \delta p_b^n \]

\[ = -\delta v - [\hat{p}_b^n] (-\delta \beta_{\omega} - \nu_{\omega}) - [\hat{\omega} - \hat{\beta}_{\omega}] \delta p_b^n \]

\[ = -[\hat{\omega} - \hat{\beta}_{\omega}] \delta p_b^n - \delta v - [\hat{p}_b^n] \delta \beta_{\omega} - [\hat{p}_b^n] \nu_{\omega}. \]

### 4.12.2 Attitude

The body-fixed attitude dynamics are defined by (4.26). They can alternately be derived from (4.84) and (4.25) as

\[ \dot{q}_b^n = (\dot{q}_n^b)^{-1} \]

\[ = \frac{1}{2} \left( q_n^b \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} \right)^{-1} \]

\[ = \frac{1}{2} \begin{bmatrix} -\omega \\ 0 \end{bmatrix} \otimes (q_n^b)^{-1} \]

\[ = -\frac{1}{2} \begin{bmatrix} \omega \\ 0 \end{bmatrix} \otimes q_b^n. \]
Using (4.28b), the body-fixed attitude is decomposed into an estimate and error state as

\[ q^n_b = \delta q^n_b \otimes \dot{q}^n_b. \]

Differentiating with respect to time gives

\[ \dot{q}^n_b = \delta \dot{q}^n_b \otimes \dot{q}^n_b + \delta q^n_b \otimes \dot{\hat{q}}^n_b. \]

Solving for the attitude error dynamics and using (4.85a) (4.28b), and (4.3) results in

\[
\delta \dot{q}^n_b = \left( \dot{q}^n_b - \delta q^n_b \otimes \dot{\hat{q}}^n_b \right) \otimes (\hat{q}^n_b)^{-1} \\
= -\frac{1}{2} \begin{bmatrix} \omega \\ 0 \end{bmatrix} \otimes q^n_b - \delta q^n_b \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} \\
= -\frac{1}{2} \begin{bmatrix} \omega \\ 0 \end{bmatrix} \otimes \delta q^n_b - \delta \omega \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \\
= -\frac{1}{2} \begin{bmatrix} \omega \\ -\omega^T \end{bmatrix} \delta q^n_b - \begin{bmatrix} -\hat{\omega} \\ -\hat{\omega}^T \end{bmatrix} \delta \omega \\
= -\frac{1}{2} \begin{bmatrix} 2\hat{\omega} + \delta \omega \\ -\delta \omega^T \end{bmatrix} \delta q^n_b ,
\]

which given (4.31) implies that

\[
\begin{bmatrix} \frac{1}{2} \delta \theta^n_b \\ 1 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 2\hat{\omega} + \delta \omega \\ -\delta \omega^T \end{bmatrix} \delta \omega \begin{bmatrix} \frac{1}{2} \delta \theta^n_b \\ 1 \end{bmatrix}.
\]
Multiplying both sides by two, dropping the scalar term, and ignoring higher order terms yields

\[ \delta \dot{\theta}_b^n = - \left[ 2 \dot{\omega} + \delta \omega \right] \delta \omega \left[ \frac{1}{2} \delta \theta_b^n \right] \]

\[ = - \frac{1}{2} \left[ 2 \dot{\omega} + \delta \omega \right] \delta \theta_b^n - \delta \omega \]

\[ \approx - \left[ \dot{\omega} \right] \delta \theta_b^n - \delta \omega \]

\[ = - \left[ \ddot{\omega} - \dot{\beta}_\omega \right] \delta \theta_b^n + \delta \beta_\omega + \nu_\omega . \]

### 4.12.3 Velocity

The only change in the velocity error dynamics is the gravity term, so that

\[ \delta \dot{v} = R^T(q_b^n)g - R^T(q_b^n)g \]

\[ = R^T(\delta q_b^n)R^T(q_b^n)g - R^T(q_b^n)g \]

\[ \approx (I + [\delta \theta_b^n]) R^T(q_b^n)g - R^T(q_b^n)g \]

\[ = [\delta \theta_b^n] R^T(q_b^n)g \]

\[ = - \left[ R^T(q_b^n)g \right] \delta \theta_b^n . \]

### 4.13 Keyframe-Reset Derivation

During the keyframe reset step, introduced in Section 4.5 and detailed in Sections 4.6.3 and 4.7.3, the relative states and their associated covariance are reset to zero. For the position states only the altitude of the vehicle is kept, while for the attitude states the uncertainty associated with yaw is removed from the filter while the uncertainty associated with roll
and pitch is maintained. Section 4.13.1 establishes some preliminary results that are used in Sections 4.13.2 and 4.13.3 to show that

$$N_\theta \triangleq \frac{\partial \delta \theta^+}{\partial \delta \theta} = \begin{bmatrix} 1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\ 0 & \cos \hat{\phi}^2 & -\cos \hat{\phi} \sin \hat{\phi} \\ 0 & -\cos \hat{\phi} \sin \hat{\phi} & \sin \hat{\phi}^2 \end{bmatrix}$$ (4.89)

for both iRN and bRN attitude definitions. Section 4.13.4 derives the position reset and its Jacobian for bRN.

### 4.13.1 Attitude Reset Preliminaries

We start by deriving roll error $\delta q_\phi$, pitch error $\delta q_\theta$, and yaw error $\delta q_\psi$, as well as a minimal representation for each, namely $\delta \theta_\phi$, $\delta \theta_\theta$, and $\delta \theta_\psi$. The error quaternions for the Euler attitude decomposition can be computed from (4.37) and (4.29). For example, we can compute $\delta q_\psi$ for iRN using (4.29a) as

$$\delta q_\psi = \hat{q}_\psi^{-1} \otimes q_\psi = \begin{bmatrix} 0 \\ 0 \\ -\sin(\frac{1}{2} \hat{\psi}) \\ \cos(\frac{1}{2} \hat{\psi}) \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ \sin(\frac{1}{2} \hat{\psi}) \\ \cos(\frac{1}{2} \hat{\psi}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sin(\frac{1}{2} \hat{\psi}) \cos(\frac{1}{2} \hat{\psi}) - \cos(\frac{1}{2} \hat{\psi}) \sin(\frac{1}{2} \hat{\psi}) \\ \cos(\frac{1}{2} \hat{\psi}) \cos(\frac{1}{2} \hat{\psi}) + \sin(\frac{1}{2} \hat{\psi}) \sin(\frac{1}{2} \hat{\psi}) \end{bmatrix}$$
\[
\begin{bmatrix}
0 \\
0 \\
\sin\left(\frac{1}{2}(\psi - \hat{\psi})\right) \\
\cos\left(\frac{1}{2}(\psi - \hat{\psi})\right)
\end{bmatrix}.
\] (4.90)

Following a similar analysis, we see (4.90) holds for bRN using (4.29b). We likewise see that
\[
\delta q_\phi = \begin{bmatrix}
\sin\left(\frac{1}{2}(\phi - \hat{\phi})\right) \\
0 \\
0 \\
\cos\left(\frac{1}{2}(\phi - \hat{\phi})\right)
\end{bmatrix}, \quad \delta q_\theta = \begin{bmatrix}
0 \\
\sin\left(\frac{1}{2}(\theta - \hat{\theta})\right) \\
0 \\
\cos\left(\frac{1}{2}(\theta - \hat{\theta})\right)
\end{bmatrix}
\] (4.91)

for both attitude error definitions. Because these errors are all expressed in different frames, \(\delta q \neq \delta q_\psi \otimes \delta q_\theta \otimes \delta q_\phi\) in general.

Assuming small attitude errors, we approximate (4.90) and (4.91) with the first-order Taylor series as
\[
\delta q_\phi \approx \begin{bmatrix}
\frac{\phi - \hat{\phi}}{2} \\
0 \\
0 \\
1
\end{bmatrix}, \quad \delta q_\theta \approx \begin{bmatrix}
0 \\
\frac{\theta - \hat{\theta}}{2} \\
0 \\
1
\end{bmatrix}, \quad \delta q_\psi \approx \begin{bmatrix}
0 \\
0 \\
\frac{\psi - \hat{\psi}}{2}
\end{bmatrix}.
\]

These error states are represented minimally according to (4.18) as
\[
\delta \theta_\phi = \begin{bmatrix}
\phi - \hat{\phi} \\
0 \\
0
\end{bmatrix}, \quad \delta \theta_\theta = \begin{bmatrix}
0 \\
\theta - \hat{\theta} \\
0
\end{bmatrix}, \quad \delta \theta_\psi = \begin{bmatrix}
0 \\
0 \\
\psi - \hat{\psi}
\end{bmatrix}.
\]
We then let
\[
\Delta \triangleq \delta \theta_\phi + \delta \theta_\theta + \delta \theta_\psi
\]
\[
= \begin{bmatrix}
\phi - \hat{\phi} \\
\theta - \hat{\theta} \\
\psi - \hat{\psi}
\end{bmatrix}.
\]

Again, it should be noted that \( \delta \theta \neq \Delta \) in general, but rather the attitude error \( \delta \theta \) is a particular linear combination of the Euler attitude errors. To update the attitude uncertainty, we solve for \( \delta \theta \) and \( \delta \theta^+ \) in terms of \( \delta \theta_\phi \), \( \delta \theta_\theta \), and \( \delta \theta_\psi \). We solve for iRN and bRN in turn.

### 4.13.2 iRN Attitude Reset

Continuing the notation presented in Section 4.6, we describe iRN attitude error with \( \delta q^b \) and \( \delta \theta^b \). Starting with (4.29a) and using (4.37) and (4.41), we see that
\[
\delta q^b = \hat{q}^{-1} \otimes q
\]
\[
= \hat{q}_\phi^{-1} \otimes \hat{q}_\theta^{-1} \otimes \hat{q}_\psi^{-1} \otimes
\]
\[
\otimes \hat{q}_\psi \otimes \delta q^b \otimes \hat{q}_\theta \otimes \delta q^b \otimes \hat{q}_\phi \otimes \delta q^b
\]
\[
= (\hat{q}_\phi^{-1} \otimes (\hat{q}_\theta^{-1} \otimes \delta q^b \otimes \hat{q}_\theta) \otimes \delta q^b \otimes \hat{q}_\phi) \otimes \delta q^b
\]
which implies that
\[
\delta \theta^b_n = R_\phi R_\theta \delta \theta_\psi + R_\phi \delta \theta_\theta + \delta \theta_\phi. \tag{4.92}
\]
We express \( \delta \theta^b_n \) in terms of \( \Delta \) as
\[
\delta \theta^b_n = \begin{bmatrix}
(\phi - \hat{\phi}) - (\psi - \hat{\psi}) \sin \hat{\theta} \\
(\theta - \hat{\theta}) \cos \hat{\phi} + (\psi - \hat{\psi}) \sin \hat{\phi} \cos \hat{\theta} \\
-(\theta - \hat{\theta}) \sin \hat{\phi} + (\psi - \hat{\psi}) \cos \hat{\phi} \cos \hat{\theta}
\end{bmatrix}
\]
\[
\begin{bmatrix}
1 & 0 & -\sin \hat{\theta} \\
0 & \cos \hat{\phi} & \sin \hat{\phi} \cos \hat{\theta} \\
0 & -\sin \hat{\phi} & \cos \hat{\phi} \cos \hat{\theta}
\end{bmatrix} \Delta . \tag{4.93}
\]

The determinant of the matrix in (4.93) equals \(\cos \hat{\theta}\). Therefore, assuming that \(\hat{\theta} \neq \pm \frac{\pi}{2}\), we invert (4.93) to obtain

\[
\Delta = \begin{bmatrix}
1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\
0 & \cos \hat{\phi} & -\sin \hat{\phi} \\
0 & \sin \hat{\phi} \sec \hat{\theta} & \cos \hat{\phi} \sec \hat{\theta}
\end{bmatrix} \delta \theta_n^h . \tag{4.94}
\]

Following a similar process for \(\delta q_n^h \) with \(q_n^h = q_\theta \otimes q_\phi \) and \(\hat{q}_n^h = \hat{q}_\theta \otimes \hat{q}_\phi \), we obtain

\[
\delta \theta_n^h = R_\phi \delta \theta_\theta + \delta \theta_\phi \\
= \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \hat{\theta} & 0 \\
0 & -\sin \hat{\phi} & 0
\end{bmatrix} \Delta , \tag{4.95}
\]

which differs from (4.93) by removing the yaw and its uncertainty. Substituting (4.94) into (4.95) we see that

\[
\delta \theta_n^h = \begin{bmatrix}
1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\
0 & \cos \hat{\phi}^2 & -\cos \hat{\phi} \sin \hat{\phi} \\
0 & -\cos \hat{\phi} \sin \hat{\phi} & \sin \hat{\phi}^2
\end{bmatrix} \delta \theta_n^h . \tag{4.96}
\]

verifying (4.89) for iRN.
Continuing the notation presented in Section 4.7, we describe bRN attitude error with \( \delta q^b_n \) and \( \delta \theta^b_n \). The typical 3-2-1 Euler angles assume an inertial attitude. To express a body-fixed attitude while still maintaining the intuitive roll, pitch, and yaw rotations, the order and sign of the rotation sequence must be flipped as

\[
q^n_b = (q^n_n)^{-1} = (q^b_n \otimes q^b_\theta \otimes q^b_\phi)^{-1} = q^{-1}_\phi \otimes q^{-1}_\theta \otimes q^{-1}_\psi .
\]

Expanding using the error state definition (4.29b),

\[
q^n_b = \hat{q}^{-1}_\phi \otimes \delta q^{-1}_\phi \otimes \hat{q}^{-1}_\theta \otimes \delta q^{-1}_\theta \otimes \hat{q}^{-1}_\psi \otimes \delta q^{-1}_\psi
\]

which is approximated as

\[
q^n_b = \hat{q}^{-1}_\phi \otimes \hat{q}^{-1}_\theta \otimes \hat{q}^{-1}_\psi .
\]

Combining (4.97) and (4.98) using (4.29b) gives

\[
\delta q^n_b = q^n_b \otimes (\hat{q}^n_b)^{-1} = \hat{q}^{-1}_\phi \otimes \delta q^{-1}_\phi \otimes \hat{q}^{-1}_\theta \otimes \delta q^{-1}_\theta \otimes \hat{q}^{-1}_\psi \otimes \delta q^{-1}_\psi
\]

which implies that

\[
\delta \theta^b_n = -R^b_\phi \delta \theta_\phi - R^b_\phi R^b_\theta \delta \theta_\theta - R^b_\phi R^b_\theta R^b_\psi \delta \theta_\psi
\]

148
Note that

$$R_\phi \delta \theta_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \phi - \hat{\phi} \end{bmatrix} = \delta \theta_\phi.$$ 

Similarly, $R_\theta \delta \theta_\theta = \delta \theta_\theta$, and $R_\psi \delta \theta_\psi = \delta \theta_\psi$. Using these results to simplify (4.99) and comparing the resulting equation to (4.92), we obtain

$$\delta \theta_\theta^n = -\delta \theta_\phi - R_\phi \delta \theta_\theta - R_\phi R_\theta \delta \theta_\psi = -\delta \theta_\theta^b. \quad (4.100)$$

Again, the reset step removes yaw, giving

$$\delta \theta_\theta^{n+} = -\delta \theta_\phi - R_\phi \delta \theta_\theta = -\delta \theta_\theta^{b+}. \quad (4.101)$$

Substituting (4.101) and (4.100) into (4.96), we see that

$$(-\delta \theta_\theta^{n+}) = \begin{bmatrix} 1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\ 0 & \cos \hat{\phi}^2 & -\cos \hat{\phi} \sin \hat{\phi} \end{bmatrix} \begin{bmatrix} -\delta \theta_\theta^n \end{bmatrix}$$

$$\Rightarrow \delta \theta_\theta^{n+} = N_\theta \delta \theta_\theta^n \quad (4.102)$$

which confirms (4.89) for bRN.
4.13.4 bRN Position Reset

Following (4.83) we can write

\[ p_{b}^{n+} = -R(q_{b}^{n+})p_{b}^{n+} \]
\[ = -R^T(q_{b}^{n+})p_{b}^{n+}, \]

where

\[ p_{b}^{n+} = kk^T p_{b}^{n} \]
\[ = -kk^T R^T(q_{b}^{n})p_{b}^{n} \]
\[ = -kk^T R(q_{b}^{n})p_{b}^{n} \] (4.103)

by rearranging (4.83). We note from (4.84) and (4.36) that

\[ q_{b}^{n} = q_{\phi}^{-1} \otimes q_{\theta}^{-1} \otimes q_{\psi}^{-1} \]
\[ \implies R(q_{b}^{n}) = R^T(q_{\psi})R^T(q_{\theta})R^T(q_{\phi}), \]

so (4.103) becomes

\[ p_{b}^{n+} = -kk^T R^T(q_{\psi})R^T(q_{\theta})R^T(q_{\phi})p_{b}^{n}. \]

We next observe that

\[ kk^T R^T(q_{\psi}) = \begin{bmatrix} 0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\
\sin \psi & \cos \psi & 0 \\
0 & 0 & 1 \end{bmatrix} = kk^T , \]
so that 
\[ p_n^{b+} = -kk^T R^T(q_\theta) R^T(q_\theta^b) p_n^b. \]

Noting that
\[ q_b^{n+} = q_\theta^{-1} \otimes q_\theta^{-1} \]
\[ \implies R(q_b^{n+}) = R^T(q_\theta) R^T(q_\theta), \]
we have
\[ p_n^{b+} = -kk^T R(q_b^{n+}) p_n^b \]
and so
\[ p_b^{n+} = R^T(q_b^{n+}) kk^T R(q_b^{n+}) p_b^n. \]

We then expand and simplify according to (4.30b), (4.32), and (4.33) while dropping second-order terms as
\[ p_b^{n+} = R^T(q_b^{n+}) kk^T R(q_b^{n+}) p_b^n \approx \]
\[ = R^T(q_b^{n+}) kk^T R(q_b^{n+}) p_b^n + R^T(q_b^{n+}) kk^T R(q_b^{n+}) \delta p_b^n \]
\[ + R^T(q_b^{n+}) kk^T R(q_b^{n+}) \delta p_b^n \]
\[ + R^T(q_b^{n+}) kk^T R(q_b^{n+}) \delta p_b^n \]
\[ - R^T(q_b^{n+}) kk^T R(q_b^{n+}) \delta p_b^n. \]

From (4.104) the estimated reset is
\[ \hat{p}_b^{n+} = R^T(q_b^{n+}) kk^T R(q_b^{n+}) \hat{p}_b^n. \]
Using (4.4) and (4.102), the error state reset is then

\[
\delta p_b^{n+} = p_b^{n+} - \hat{p}_b^{n+}
\]

\[
= R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \delta p_b^n
\]

\[
+ \begin{bmatrix} \delta \theta_b^n \end{bmatrix} R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \hat{p}_b^n
\]

\[
- R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \begin{bmatrix} \delta \theta_b^n \end{bmatrix} \hat{p}_b^n
\]

\[
= R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \delta p_b^n
\]

\[- \begin{bmatrix} R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \hat{p}_b^n \end{bmatrix} \delta \theta_b^n
\]

\[+ R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \begin{bmatrix} \hat{p}_b^n \end{bmatrix} \delta \theta_b^n
\]

\[
= R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \delta p_b^n
\]

\[- \begin{bmatrix} R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \hat{p}_b^n \end{bmatrix} N_\theta \delta \theta_b^n
\]

\[+ R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \begin{bmatrix} \hat{p}_b^n \end{bmatrix} N_\theta \delta \theta_b^n ,
\]

and the non-zero Jacobian terms are

\[
\frac{\partial \delta p_b^{n+}}{\partial \delta p_b^n} = R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+})
\]

\[
\frac{\partial \delta p_b^{n+}}{\partial \delta \theta_b^n} = \left(- \begin{bmatrix} R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \hat{p}_b^n \end{bmatrix}
\]

\[+ R^T(\hat{q}_b^{n+})kk^T R(\hat{q}_b^{n+}) \begin{bmatrix} \hat{p}_b^n \end{bmatrix} \right) N_\theta .
\]
Chapter 5

Relative Navigation of Autonomous GPS-Degraded Micro Air Vehicles

5.1 Introduction

Economists anticipate that autonomous micro air vehicles (MAVs) will give rise to a handful of billion-dollar markets, including infrastructure inspection, security, precision agriculture, transportation, and delivery [3]. Using MAVs to inspect bridges, dams, chemical plants, and refineries is particularly motivating as it would take the place of dangerous, time consuming, and expensive human inspections; however, these markets are still largely speculative because autonomous MAV navigation is an active research problem, especially in confined, unknown environments where global positioning system (GPS) measurements are unavailable or degraded.

---

1This paper was written by David O. Wheeler, Daniel P. Koch, James S. Jackson, Gary J. Ellingson, Paul W. Nyholm, Timothy W. McLain, and Randal W. Beard, and is available at [32]

Figure 5.1: MAV smoothly navigating through a GPS-degraded environment.
Current MAV navigation approaches rely heavily on GPS for estimation, guidance, and control; however, GPS signals can be spoofed, jammed, or blocked by structures and foliage. GPS measurements can be further degraded by multipath, atmospheric delays, or poor positioning of visible satellites. When GPS is unavailable, the MAV’s global position and heading is not observable [5, 30, 31]. As a result, the state estimates eventually drift, leading to filter inconsistency and non-optimal sensor fusion [2, 7]. Significant reliability issues arise when working with respect to a globally-referenced state during prolonged GPS dropout and heading uncertainty [8].

Despite these issues, many current GPS-denied MAV navigation approaches continue to estimate and control with respect to a single, inertial reference frame: either the GPS origin or the MAV’s initial pose. This formulation is convenient; however, there are several underlying issues that commonly arise in GPS-degraded environments when estimation and control are carried out with respect to a global reference frame:

- Controlling with respect to the unobservable global state precludes any guarantee on the stability of the system.
- In the absence of global measurements, estimates of the unobservable global state drift over time and the uncertainty grows without bound. If GPS is reacquired after a prolonged period of dropout and used as an update in the filter, the global state may jump considerably. This jump, if not accounted for, may in turn produce extreme control inputs [23, 30, 45, 46, 67]. A large global uncertainty also reduces the filter’s ability to properly reject degraded GPS measurements, causing the state estimate to degrade.
- During prolonged GPS dropout, the unobservable global states become inconsistent [7, 8], resulting in a poor understanding of the uncertainty of the vehicle’s global pose. Inconsistency reduces estimator optimality [2], can cause the estimator to gate valid GPS
measurements if GPS is eventually reacquired, and can negatively impact applications such as geofencing that require a good understanding of the global uncertainty.

While various methods have been introduced in the literature to help mitigate or work around these issues, ultimately the root cause is unobservability.

This paper uses the recently proposed relative navigation (RN) framework [28] as an alternative, observable approach for GPS-denied MAV navigation. By using a view matcher, such as camera-based visual odometry [19,20] or laser-based scan matching [21,22], relative navigation estimates the MAV’s state with respect to its local environment. The relative state estimator ensures that the state is observable and the uncertainty remains bounded, consistent, and normally-distributed [8]. By removing the global-state estimation from the front end, RN also ensures that large or delayed global-state updates, which come from incorporating loop-closure constraints or eventual global measurements, do not impact the flight-critical control and estimation feedback. Rather, the global state is estimated independently using a pose-graph back end where the non-Gaussian uncertainties can be better represented and robust optimization methods can identify and reject gross GPS outliers and false-positive loop closures.

The contributions of this paper are twofold. First, the details necessary to implement the complete RN framework are presented. Specifically, we describe the relative estimator reset operation necessary to maintain observability, and present the relative guidance and control strategy necessary to ensure smooth, stable flight. We discuss how to reconstruct the global state with consistent banana-shaped uncertainty distributions, and describe how to incorporate GPS and loop-closure information to improve the global state estimate. We explain how the high-level path planner facilitates autonomous missions and show how to leverage off-the-shelf algorithms for visual odometry, place recognition, and robust pose-graph optimization.
The second contribution consists of several prolonged hardware flight tests demonstrating the effectiveness of RN for autonomous GPS-degraded MAV navigation in varied, unknown environments, such as that shown in Figure 5.1. We demonstrate that the relative front end successfully fuses multiple vision sensors, works indoors and outdoors, and results in low drift with no state jumps. We further demonstrate the onboard generation of a globally-consistent, metric, and localized map by identifying and incorporating loop-closure constraints and intermittent GPS measurements. Using this map, we demonstrate the fully-autonomous completion of mission objectives, including performing a position-hold about a global position waypoint while in a GPS-denied environment.

Section 5.2 reviews current state-of-the-art methods for GPS-degraded MAV navigation and Section 5.3 overviews the relative navigation framework. Sections 5.4 and 5.5 describe the components of the relative front end and global back end of the RN architecture respectively. In addition to outlining each component’s role, the specific algorithms used for the hardware implementation are also presented. Section 5.6 describes the experimental flight tests, including the hardware and test procedures, while Section 5.7 describes the flight test results. Finally, Section 5.8 summarizes the contributions of the paper.

5.2 Related Work

Because of the many applications of MAVs in GPS-denied and GPS-degraded environments, significant research has been performed in improving the capability and robustness of state estimation in these situations. Much of this work builds upon the simultaneous localization and mapping (SLAM) literature, but is adapted for MAVs. The full SLAM problem involves concurrently estimating the position of surrounding landmarks while reconstructing the vehicle’s complete trajectory; however, due to the strict size, weight, power, and timing requirements associated with autonomous MAV operation, the SLAM problem
is often simplified when applied to MAVs, only solving for the current pose of the vehicle and surrounding landmarks.

Early work [21, 41, 68–70] demonstrated indoor MAV flight and provides approaches for many MAV navigation problems such as mapping, path planning, and control of GPS-denied multirotor platforms for short indoor trajectories. Refs. 41 and 68 present a graph-based SLAM approach to leverage laser scan-matching constraints, while Refs. 21 and 69 fuse scan matching data with inertial measurements in an extended Kalman filter (EKF), demonstrating a vision-aided navigation solution. Refs. 70 uses an EKF to track the global pose of individual landmarks, demonstrating a successful EKF-SLAM approach.

Some more recent work in this area [23, 25, 35, 71], has focused on improving the consistency of pose estimation without global measurements, extending the length of autonomous trajectories, and diversifying the environments in which MAVs can operate. Chowdhary et al. demonstrated a successful GPS-denied monocular vision-aided inertial navigation system (INS) including autonomous landing and takeoff [25]. Scaramuzza et al. were the first to demonstrate prolonged (350 m) autonomous MAV flight in a GPS-denied environment [35]. Their work used a single monocular camera for onboard stabilization and control. Shen et al. introduced a method for simultaneously fusing multiple relative view-matchers to increase robustness in difficult environments and demonstrated autonomous flight on a prolonged (440 m) indoor-outdoor flight [23]. They used a stochastic cloning filtering approach [29], which is designed to better propagate uncertainty but allows the global state covariance to grow unbounded in the absence of global measurement updates. Scherer et al. presented a graph-based state estimation system that fuses visual odometry, inertial measurements, and intermittent GPS information [71]. The relative navigation approach shares many ideas with this approach, but removes the pose-graph optimization from the flight-critical path by additionally incorporating a front-end estimator.
Each of these previously mentioned methods ultimately track the unobserved global state. As shown in Ref. 8, methods that directly estimate the global state are susceptible to inconsistency and state jumps during prolonged GPS dropout. The value of a relative parameterization is well-documented in the full-SLAM literature [7, 15, 16, 72], but has not been fully applied to MAV navigation. Moore et al. noted the limitations of using either a body-fixed or a globally-fixed reference frame for ground vehicles, and proposed using a local frame in which the vehicle moves smoothly [72]. Bailey et al. showed that estimating the vehicle and landmark location with respect to a global coordinate frame results in inconsistency as heading uncertainty increases, and asserted that submapping was the only method at the time of publication for implementing consistent large-scale EKF-SLAM [7]. Relative submapping methods [15, 16] estimate the state of the vehicle and landmarks with respect to a local coordinate frame. These submaps are subsequently fused and form a more consistent global estimate. In essence, relative navigation demonstrates how to apply these relative submapping ideas discussed in the full-SLAM literature to computationally constrained MAV platforms using an EKF to ensure smooth flight in GPS-degraded environments.

5.3 Relative Navigation Overview

The intuition behind relative navigation is straightforward. An alert driver can safely navigate indefinitely, even if completely lost or disoriented. This is because humans instinctively perceive the world and make decisions with respect to the current local environment, as opposed to working with respect to an arbitrary global reference point. When a driver is lost, ideally an accompanying passenger looks for landmarks, references a map or GPS unit, plans the optimal global route, and then provides low-frequency, high-level instructions to the driver in the local frame—for example, “turn around when possible” or “make the next right turn.” In this way, time- and safety-critical estimation and control decisions are decoupled from potentially erroneous global information.
Figure 5.2: Relative navigation architecture. Using relative motion measurements, such as from visual odometry or scan matching algorithms, the vehicle estimates its local state. These estimates are used for flight-critical path planning and control. As a separate process, the global back end incorporates any available global information. Its only influence on the front end is through locally-defined guidance objectives.

Figure 5.2 presents the relative navigation architecture introduced in Ref. 28, where the decoupled responsibilities of the relative front end and global back end are analogous to a driver and passenger. Using relative motion measurements, available from a view-based odometry source such as visual odometry or scan matching, the vehicle estimates its pose with respect to its local environment. This observable, relative state is used for flight-critical path planning and control. As a separate process, the global back end uses a pose-graph map to combine these relative states into a global map, and to incorporate any available global information such as place recognition constraints or GPS measurements. The only way the global back end influences the front end is through locally-defined guidance objectives.

The relative navigation architecture is readily applied to existing systems, as it does not make any assumptions about the vehicle platform or sensor suite. A wide variety of algorithms can be used to implement each component, and due to the modular nature of RN, it is straightforward to interchange the algorithms as needed. The RN framework also allows multiple view-matchers to be used simultaneously for increased robustness in difficult environments. In the next two sections we describe the details of the relative front end and the global back end.
5.4 Relative Front End

By working with respect to the local environment, the relative front end ensures that the flight-critical estimation, guidance, and control always operates with respect to an observable state, allowing smooth, stable flight even when global information is degraded or undergoing large corrections. When relative navigation was first presented in Ref. 28, the discussion emphasized a particular choice for a visual odometry algorithm, estimator, path planner, and controller. This section generalizes that discussion by outlining the fundamental nature of each front-end component, highlighting how existing algorithms would need to be adapted to fit into the relative navigation architecture. Specifically, we describe how to incorporate current state-of-the-art view-based odometry algorithms, describe the relative estimator reset operation necessary to maintain observability, and present the relative guidance and control strategy necessary to ensure smooth, stable flight.

5.4.1 View-Based Odometry

When GPS is not available, MAVs commonly use odometry measurements computed from exteroceptive sensors such as laser scanners or cameras. A variety of odometry algorithms exist including laser scan matching [21, 22] and visual odometry [19, 20]. While some odometry methods compare consecutive frames (scans or images), others compare the current frame to a recent keyframe. When a keyframe is used, a series of odometry measurements are computed with respect to this keyframe. Generally the keyframe is updated only when there is insufficient overlap to provide a reliable odometry measurement. As a result, keyframe-based odometry reduces temporal drift in the computed odometry as compared to frame-to-frame matching [23, 24]. Many view-based odometry algorithms use bundle adjustment to further improve accuracy [19].

While view-matching algorithms only measure relative motion, implementations of these algorithms commonly concatenate the measurements to output a global odometry estimate.
Rather than treating the odometry as a black-box pseudo-global measurement like in Refs. 35 and 36, the view-matcher used in the proposed RN architecture directly outputs relative measurements. View-matching algorithms that output pseudo-global measurements can be readily adapted to supply relative updates.

As shown in Figure 5.2, the view-matcher is only loosely coupled to the estimator. As such, it is straightforward to accept relative measurements from any source or sensor, such as monocular, stereo, and RGB-D visual odometry or a laser scan matcher. The framework even handles multiple relative sensors, which can increase the robustness of the system in difficult environmental conditions. For example, Ref. 73 demonstrates using RN to simultaneously incorporate relative measurements from a laser scanner and RGB-D camera. While the scan matcher breaks down in long hallways and the visual odometry breaks down in a dark room, the redundant sensing allowed the vehicle to successfully navigate. Many tightly-coupled vision-aided INS estimators, such as Refs. 25, 74, and 26, could be adapted and treated as a view-based odometry source for a relative estimator. For the results in Section 5.7 we used DEMO [48] for visual odometry and CSM [75] for scan matching.

5.4.2 Relative State Estimation

Most MAV navigation approaches continue to estimate the global state, even when GPS-dropout makes the global state unobservable. Given an inertial measurement unit, altimeter, and even visual odometry measurements, the global position and heading of a MAV in the horizontal plane cannot be observed [5, 30]. With time, the associated state estimates drift and become inconsistent.

One fundamental advantage of RN is that the front-end state always remains observable. RN maintains observability by defining the state with respect to a local node frame. The node frame is defined as the gravity-aligned coordinate frame that is positioned on the ground exactly under the MAV when the current keyframe was taken. Because each node frame is
Figure 5.3: 2D illustration of node frame reset operation. (a) The relative estimator tracks the MAV’s position and heading \((x, y, \psi)\) with respect to the current node frame \(n_k\). The estimated state (blue) will not perfectly align with the true MAV state (green), but the estimator’s covariance (blue oval) should correctly represent the underlying uncertainty. (b) When a new keyframe is declared, the new node frame \(n_{k+1}\) is defined at the true, yet globally unknown, MAV pose. The estimated pose (gray) and covariance (gray oval) are saved as an edge constraint in the back-end pose graph and the MAV’s \((x, y, \psi)\) states and their corresponding covariance terms are reset to zero.

Gravity-aligned and positioned on the ground, the MAV’s altitude, roll, and pitch \((z, \phi, \theta)\) with respect to the node frame are estimated no differently than if defined with respect to a global origin. By referencing the current node frame, however, the horizontal position and yaw states \((x, y, \psi)\) now correspond to the relative position and heading of the MAV with respect to the most recent odometry keyframe. In this way, relative measurements provided by a view-matcher directly measure the MAV’s relative position and heading, causing the state to be observable by construction. With regular, direct updates, the uncertainty of the vehicle’s relative state remains consistent, bounded, and approximately Gaussian [8].

A variety of estimation techniques are used for MAV navigation and could be adapted to become a relative estimator. The fundamental concept is that the estimator’s state and covariance should be reset whenever a new keyframe is declared. Figure 5.3 illustrates the process of transitioning from one keyframe to the next. The relative estimator tracks the MAV’s position and heading \((x, y, \psi)\) relative to the current node frame \(n_k\). Naturally the estimated state will not perfectly align with the true MAV state, but the estimator’s covariance should correctly represent the underlying uncertainty. When a new keyframe
Figure 5.4: Typical mid-flight state estimates. The vertical gray lines indicate a new node frame, and the labels indicate the associated node identifier. With each new node frame, the forward, right, and yaw states are reset to zero, while the down, roll, and pitch states are unaffected. The vehicle was yawing from 142 s to 146 s and moving forward at a constant velocity from 150 s to 152 s. While the state estimates are discontinuous, the relative navigation approach facilitates smooth, stable navigation in GPS-degraded environments.

is declared, the new node frame $n_{k+1}$ is defined at the true, yet globally unknown, MAV position. The current pose and covariance are saved as an edge constraint in the back-end pose graph and then the MAV’s position and heading states and their corresponding covariance terms are reset to zero. In this way, the global uncertainty is removed from the front-end filter and delegated to the global back end.

Figure 5.4 shows example state estimates, where the horizontal position and heading states are reset at each new node frame. While the discontinuities in the state estimates may appear concerning from a control perspective, they occur at known times and thus are reliably handled by the relative path planner and controller to produce smooth, stable control. It is important to note that while the front-end filter tracks the full six degrees-of-freedom pose, it is sufficient to only optimize the relative states $(x, y, \psi)$ in the back end.
For the flight results described in Section 5.7, we used an indirect formulation of the multiplicative extended Kalman filter as presented in Ref. 39. A unit quaternion is used to represent the MAV’s attitude while attitude error is propagated using a minimal three-state representation. When a new keyframe is declared, care is taken to only reset the unobserved horizontal position and heading, leaving roll, pitch, altitude and their associated uncertainties unchanged. Refer to Ref. 39 for additional estimator details including the state, dynamics, sensor models, and specific details about the reset step.

Some similarities and differences exist between the RN relative estimator and the popular robocentric estimation approach. As described in Ref. 39, a relative estimator can be defined in either an inertial or robocentric representation. A robocentric approach tracks the position of landmarks with respect to the current body’s pose. While robocentric dynamics are less intuitive, this formulation ensures improved observability and consistency properties for the landmark states, similar to RN. Most robocentric approaches, however, continue to track the global state of the MAV with respect to the current body. After prolonged flight without global information, the global uncertainty is not well represented by a Gaussian distribution in typical Cartesian coordinates, leading to estimator inconsistency [8]. The relative navigation framework provides a method to use either inertial or body-fixed dynamics, produces smooth, observable state estimates for control, and represents the global state consistently.

5.4.3 Relative Path Planning and Control

Within the relative navigation framework, all front-end guidance and control is computed with respect to the current node frame. Many current MAV controllers drive the estimated global state to a desired global state. These same controllers can be directly applied to drive the estimated relative state to a desired relative state. Any control approach can be used as long as care is taken to ensure that the estimator and controller are working with respect to the same reference frame.
Figure 5.5: Updating a relative goal when a new node frame is declared. The goal with respect to the previous keyframe, $x_{g/k-1}^{k-1}$, is expressed with respect to the new keyframe, $x_{g/k}^k$, using the edge constraint $x_{k/k-1}^k$ provided by the relative estimator.

Each time the relative estimator resets to a new node frame, the path planner and controller must also update to ensure that they are operating with respect to this new frame. Depending on the chosen control strategy, this update operation may range from updating an entire potential field to requiring no action as in the case of a body-fixed velocity controller. Let $x_{a/b}^c$ represent the state $a$ with respect to frame $b$, expressed in frame $c$. Using this notation, Figure 5.5 illustrates the process of updating a position goal that is expressed with respect to the previous node frame $n_{k-1}$ to the current node frame $n_k$. In short, the relative path planner uses the estimated edge constraint between subsequent node frames provided by the relative estimator, $x_{k/k-1}^{k-1}$, to express the previous goal $x_{g/k-1}^{k-1}$ in the new node frame, $x_{g/k}^k$. Because each node frame is gravity-aligned and positioned on the ground, any roll, pitch, altitude, or body-fixed velocity components of the goal remain unchanged.

As a practical note, we recommend that the relative controller incorporate logic to monitor if the relative estimator’s node frame identifier matches the node frame identifier of the current goal. If the node frame identifiers are not in sync or no goal is supplied by the path planner, the MAV is directed to hover in place. While this step is an important safety precaution, the controller did not enter this state during our flight testing. With a careful implementation, the control performance does not degrade due to the relative state reset.
Figure 5.6 presents the control architecture used to avoid collisions and produce the smooth, flight-critical control needed to safely operate the MAV in unknown, dynamic environments with unpredictable external disturbances. The onboard computer uses its current relative state estimate and a path planning algorithm to calculate a trajectory to the current relative goal. We use the reactive obstacle avoidance plugin framework [76] to use the latest sensor information to modify the current trajectory when needed to avoid a pending collision. Control loops are then closed around this modified trajectory to produce desired accelerations. At this point, the non-linear model of the MAV dynamics is inverted [77, 78], providing a desired roll, pitch, yaw rate, and thrust command. These attitude setpoints are passed to the autopilot where high-rate attitude feedback control is performed.

For the results in Section 5.7, the path planner uses position feedback to supply high-rate velocity goals. These velocity goals are then modified using the cushioned extended-periphery obstacle avoidance algorithm [76]. An LQR feedback controller is closed around the modified velocity setpoints to produce desired accelerations, which are then passed through the model inversion to produce the roll, pitch, yaw rate, and thrust command that is sent to the autopilot.

5.5 Global Back End

While the relative front end ensures flight-worthiness, if a MAV is tasked with performing a global mission then a global state estimate is required. This section describes how the global state and its uncertainty are reconstructed. While the overall concept of the RN back end was presented previously in Ref. 28, the implementation details presented in this section are unique contributions of this paper.
Figure 5.6: Control architecture. The autopilot performs high-rate feedback control about roll, pitch, yaw rate, and thrust commands provided from the onboard computer. Diagram adapted from Ref. 1.
5.5.1 Pose-Graph Map

Before resetting the state and establishing a new node frame, the front end saves the estimated relative pose and associated uncertainty. Because each node frame is defined to be located at the true (yet globally unknown) position of the MAV, the uncertainty is reset with each node frame. This ensures that the saved pose estimates from one node frame to the next are mutually independent. This facilitates structuring the back end as a pose graph.

A pose graph is a conventional graph where each vertex or node corresponds to the global pose of a vehicle at a certain instant in time, and graph edges represent the relative change in position and attitude from one node to another. Odometry measurements, such as the relative pose estimates from the relative estimator, provide edge constraints between sequential nodes. If a place recognition algorithm detects that the vehicle has returned to a previous pose, an edge constraint between non-consecutive poses, known as a loop closure, is introduced in the graph. The vehicle’s global pose can be reconstructed by first traversing the graph from the origin to the current node, compounding each estimated edge in the path, and then incorporating the relative state. When loop closures are added, the graph is over-constrained and multiple paths, and therefore multiple pose estimates, are possible. This is illustrated in Figure 5.7. A weighted-least-squares optimization can be performed to reconcile these discrepancies, removing accumulated drift. Other, more involved frameworks
leverage the factor graph data structure which uses Bayesian methods to infer the pose of the MAV over time by representing edge constraints as factors. Factor graph methods have the added benefit of being able to solve for the global uncertainty of each pose and can incorporate other measurements such as range-only or IMU preintegration factors [27, 79]. Both factor-graph and pose-graph formulations are able to solve for the optimal set of poses given odometry and loop-closure edge constraints with associated uncertainties.

Formulating the back-end optimization problem as a pose graph results in the following beneficial properties:

- A variety of well-developed pose-graph optimization frameworks exist to find a consistent global representation of the trajectory after accounting for all constraints [42, 43, 80, 81].
- Robust pose-graph optimization techniques can identify and remove the effect of erroneous constraints such as false-positive loop closures or degraded GPS [82–84].
- A pose-graph representation provides a straightforward method to consistently represent a MAV’s global state uncertainty. When global measurements are unavailable, representing error using the vector space formed by the Lie algebra se(3) produces banana-shaped, Gaussian uncertainty distributions that better parameterize the underlying distribution [8, 40, 50].
- A pose graph provides a lightweight representation of a trajectory, ensuring scalability and practicality on resource-constrained platforms or networks. Long trajectories with a large number of loop closures can benefit from node removal techniques which further reduce the complexity of the optimization problem [85].

Pose graphs are commonly used for MAV back ends; however, many approaches that track the global state in the front end do not provide a clear method to construct independent edge constraints and covariances, an issue addressed explicitly by relative navigation.
5.5.2 Place Recognition

An important aspect of pose-graph back ends is the ability to remove accumulated drift if the MAV detects that it has returned to a previously visited location. Place recognition algorithms efficiently compare the current keyframe image or scan to each previous keyframe image or scan. When a strong correspondence is detected, the relative transformation is computed and an edge constraint between non-consecutive nodes, known as a loop closure, is included in the pose graph.

Place recognition is a challenging problem, but a variety of approaches have been successfully demonstrated [86]. To scale well, the method must be fast and efficient. Additionally, the algorithm should correctly detect loop closures when there are partial occlusions, varied viewpoints or lighting conditions, or minor scene changes. It should also correctly avoid perceptual aliasing, which is falsely correlating nearly identical, yet non-unique, scenes such as two similar brick walls.

To ensure scalability, many approaches use a bag-of-words approach [87, 88]. Salient features are identified in a representative set of images and are clustered to form a set of common, yet visually distinct, image features. This precomputed set of features, known as the vocabulary, is then used to describe each vehicle image. Using a common vocabulary allows for a sparse representation and facilitates rapid comparison. Commonly, hierarchical trees are also used for quicker comparisons. Some methods use the estimated global uncertainty to limit the set of past images that are compared.

While any place recognition algorithm could be used, we use fast, appearance-based mapping (FAB-MAP), a linear-complexity algorithm that uses Bayesian probabilities to infer the likelihood of a match while explicitly rejecting perceptual aliasing in the environment [89, 90]. This appearance-based matching technique provides only an image pair, so the RGB-D visual odometry algorithm described in Ref. 28 is used to calculate the full six degrees-
of-freedom transform between the two images. This algorithm uses RANSAC [91] to find the transform between the RGB-D image pair, and the number of outliers in the RANSAC model can be used to filter false loop closures. With this method, no false loop closures have been detected in the entirety of our flight-testing experience, and it has been shown to be computationally tractable on a MAV. An example loop closure is shown in Figure 5.8.

5.5.3 GPS Integration

While loop closures and odometry can be used in a pose graph formulation to produce a metric map of previous states, globally-referenced measurements, such as GPS, can be used to localize the map in the global frame and further improve global-state estimation. Measurements to landmarks with known global positions can also be used to localize the map globally. For example, while the results presented in Section 5.7 do not use any \textit{a priori} information, it is trivial to seed the place recognition algorithm with a set of geo-located images.

Many MAV navigation methods estimate the global state in the front end and can directly fuse global measurements. This works well when global information is regularly available and accurate, but is shown to lead to inconsistency when the estimates drift during prolonged GPS dropout [7, 8]. Furthermore, directly applying a global measurement to remove drift induces a large state update, often causing the control effort to jump which can destabilize the system [23, 30, 45, 46, 67]. Several methods have been proposed to address this, such as
Figure 5.9: Back-end GPS integration method. For each GPS measurement, one node and two edge constraints are added. The new node (green circle) is related to the virtual-zero node using the measurement and uncertainty reported by the GPS receiver (dashed green line), and is related to the current node frame using the current relative state estimate (solid green line). A virtual constraint with maximum uncertainty is added between the first node and the virtual zero node to ensure connectedness (black line).

Simultaneously tracking a GPS-corrected and odometry-only global trajectory [23] or using a series of measurement gates [46].

Alternatively, global measurements can be handled exclusively in the back-end pose graph using a virtual-zero node. Described in Ref. 71,92, the virtual-zero node represents the GPS origin. To ensure the pose graph is fully connected, an arbitrary edge constraint with infinite uncertainty, known as the virtual constraint, is applied between the virtual-zero node and the node representing the MAV’s origin. For each GPS measurement received, one node and two edge constraints are added to the pose graph, as shown in Figure 5.9. A node is added to represent the current vehicle pose. This node is related to the virtual-zero node using the measurement and uncertainty reported from the GPS, and is related to the current node frame using the current relative state estimate. Upon optimizing the pose graph after the first GPS measurement, the virtual constraint will correctly estimate the global position of the MAV’s starting point. Incorporating subsequent GPS measurements will refine this position estimate and provide a heading estimate for the MAV’s starting point, causing the entire pose graph map to be globally localized. Similar concepts have been used to incorporate multiple agents with unknown initial starting points [16].
In practice, pose graph optimizers are less likely to diverge when all constraints are of a similar order of magnitude. GPS constraints are challenging because the GPS origin is generally far away. To address this issue, we save the initial GPS measurement and subtract it from each GPS measurement before adding the edge constraint. As a result, the virtual zero constraint represents the position of the first node with respect to the first GPS measurement, as opposed to representing the position of the first node with respect to the GPS origin. If it is necessary to express the pose graph in a global coordinate frame, such as for visualizing the graph on an ortho-rectified image, the initial GPS measurement is simply added to each pose.

There are several significant advantages of using pose graphs for incorporating GPS measurements. First, due to the decoupled nature of the relative navigation framework, global state jumps cannot degrade flight-critical control. This also means that processing or networking delays can be tolerated. Second, robust optimization techniques can be used to detect erroneous GPS measurements. Once detected, any negative effect is completely removed from the system. Such a claim is not possible using conventional, front-end filtering methods. Finally, as few as two global measurements can be leveraged to localize the pose graph map, a research problem originally motivated in Ref. 92.

5.5.4 Optimization

Pose graph optimization is formulated as a weighted least-squares problem. The objective of the optimization is to find the set of global poses $\mathbf{x}$ for each node such that the set of relative edge constraints $\xi$ are best satisfied collectively. Edge constraints are partitioned into three sets: odometry constraints $\mathcal{O}$, loop-closure constraints $\mathcal{L}$, and GPS constraints $\mathcal{G}$. Each edge constraint $\xi_{ij}$ has an associated information matrix $\Omega_{ij}$ to represent the confidence of the constraint connecting nodes $i$ and $j$. A particular estimate of global node poses $\hat{\mathbf{x}}$ can
be used to determine the currently estimated relative relationship between nodes:

\[ \hat{\xi}_{ij} = h_{ij}(\hat{x}) . \]

Using this notation, the optimization is formulated as

\[ \hat{x}^* = \arg\min_{\hat{x}} \sum_{\xi_{ij} \in \{O, L, G\}} (h_{ij}(\hat{x}) - \xi_{ij})^T \Omega_{ij}(h_{ij}(\hat{x}) - \xi_{ij}) . \]

Before loop-closure and GPS constraints are introduced into the system, the optimization problem is not overconstrained and a zero-cost, odometry-only trajectory is available. When additional constraints are added, the optimization works to modify the trajectory, particularly adjusting the portions of the trajectory with the greatest uncertainty.

Pose-graph optimization is a well-researched problem. The optimization is commonly solved using iterative Gauss-Newton techniques. First, the global position of each node is estimated, often using the odometry-only trajectory. Then, for each iteration, the cost function is linearized about the current state estimate and the optimal state update for the linearized system is computed and applied. There are several known issues with this method that are addressed in the literature:

- A naive implementation requires large matrix inversions and therefore does not scale well. However, several popular pose-graph optimization frameworks have been presented that leverage sparse matrix properties and show improved scalability [42, 43].
- Gauss-Newton approaches can converge to a local minimum or even diverge, particularly when the initial state estimate is poor, which is common for drifting MAVs in GPS-denied environments. Several approaches have been presented to address initialization issues, including Ref. 93.
• Least-squares optimization is highly sensitive to outliers. While outliers are unlikely for odometry constraints, false-positive loop-closure constraints or degraded GPS measurements can significantly impact the optimization. Switching constraints [82], dynamic covariance scaling (DCS) [83], max-mixture models [94] and the RRR algorithm [84] are all proven methods for detecting outliers and mitigating their effect on the optimization.

While these and similar methods help prevent the back end from diverging, they do not guarantee convergence, nor do they necessarily provide smooth or timely global-state estimates. This further highlights the importance of decoupling flight-critical processes from global information. For the flight-test results in Section 5.7 we used g2o [42] with dynamic covariance scaling [83].

5.5.5 Global Path Planning

The role of the global path planner is first to determine the optimal MAV trajectory by assessing relevant global information, and second to transform the plan to be with respect to the current node frame for use in the relative front end. A variety of path planning algorithms could be used depending on the mission objective, including autonomous exploration, mapping, target tracking, waypoint following, cooperative control, or landing. After a plan is determined, the global path planner passes relative goals to the relative path planner. When a new keyframe is declared, these goals are updated to be expressed with respect to the latest node frame. These relative goals are the only way the global back end influences the MAV, which helps isolate the front end from destabilizing or erroneous global information. This idea is illustrated in Figure 5.10.

A simple global path planner was implemented for the flight test results in Section 5.7. Since the MAV begins without any global information, a user initially takes the place of
the global path planner by supplying a series of position or velocity setpoints. After the MAV travels for some distance and creates a global map, the user specifies a desired global waypoint on the map. At this point, Dijkstra’s algorithm is used to search through the back-end pose graph to find the shortest known path to the desired waypoint. The global path planner then supplies velocity setpoints to the relative front end to direct the MAV along the path to the global waypoint. This method is sufficient for autonomous MAV navigation in unknown environments and demonstrates the role of the global path planner, but more sophisticated planners could be implemented for other mission scenarios.

5.6 Experimental Setup

The experimental platform, shown in Figure 5.11, is a hexacopter with a diameter of 0.69 m through the prop centers and a mass of 4.8 kg. The vehicle carries a 3DR Pixhawk autopilot, onboard computer, IMU, RGB-D camera, planar laser scanner, GPS receiver, and ultrasonic altimeter. The details of the hardware configuration are summarized in Table 5.1. It is important to note that the purpose of this research is to demonstrate a successful framework for GPS-degraded MAV navigation and not to meet a particular specification or optimally address a specific application. We selected common sensors, processors, and algorithms without much consideration for optimizing the MAV’s size, weight, speed, or endurance.

The data flow and networking between the various system components are illustrated in Figure 5.12. The relative navigation framework was implemented entirely on the onboard computer in C++ using the Robot Operating System (ROS) [95] middleware. Attitude control was performed by a 3DR Pixhawk autopilot running a customized version of the PX4 firmware\(^\text{2}\). During fully autonomous sections of flight, a ground station laptop was used

\(^2\)The PX4 firmware is customized to accept inputs from the onboard computer while also allowing an RC safety pilot to override these commands if necessary. We have subsequently transitioned to using the ROSflight autopilot [96]; see http://rosflight.org.
Figure 5.10: Illustrations of how incorporating global information influences vehicle control. The columns respectively represent estimation and planning, and the dashed arrows indicate optional relationships. (a) Introducing global information into a conventional approach causes a global state jump which directly influences control [23, 30, 45, 46, 67]. (b) With the relative navigation approach, a global state jump never affects the relative state estimate. Vehicle control is only influenced as the global path planner provides an updated relative goal to the relative path planner.

Figure 5.11: The vehicle used for the flight tests
Table 5.1: Hardware details

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Hexacopter, 4.8 kg, 0.69 m diameter</td>
</tr>
<tr>
<td>Autopilot</td>
<td>3DR Pixhawk</td>
</tr>
<tr>
<td>RGB-D Camera</td>
<td>ASUS Xtion Pro Live</td>
</tr>
<tr>
<td>Laser Scanner</td>
<td>Hokuyo UTM-30LX</td>
</tr>
<tr>
<td>IMU</td>
<td>MicroStrain 3DM-GX3-15</td>
</tr>
<tr>
<td>Altimeter</td>
<td>I2CXL-MaxSonar-EZ MB1242</td>
</tr>
<tr>
<td>GPS</td>
<td>U-blox LEA-6T</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Core i7-2710QE (2.1 GHz × 4)</td>
</tr>
<tr>
<td>Memory</td>
<td>8GB DDR3</td>
</tr>
</tbody>
</table>

Figure 5.12: The data flow and networking between the various system components
Table 5.2: Summary of flight tests

<table>
<thead>
<tr>
<th>Flight Test</th>
<th>Environment</th>
<th>Distance</th>
<th>Duration</th>
<th>Sensor</th>
<th>GPS</th>
<th>Loop Closures</th>
<th>Nodes</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (§5.6.1)</td>
<td>outdoor (dusk)</td>
<td>320 m</td>
<td>9 min</td>
<td>RGB-D</td>
<td>denied</td>
<td>0</td>
<td>491</td>
<td>5.17</td>
</tr>
<tr>
<td>2 (§5.6.2)</td>
<td>indoor</td>
<td>390 m</td>
<td>12 min</td>
<td>RGB-D</td>
<td>denied</td>
<td>139</td>
<td>659</td>
<td>5.13,5.14,5.16,5.18</td>
</tr>
<tr>
<td>3 (§5.6.3)</td>
<td>indoor/outdoor (night)</td>
<td>240 m</td>
<td>9 min</td>
<td>laser</td>
<td>intermittent</td>
<td>30</td>
<td>891</td>
<td>5.1,5.15,5.19</td>
</tr>
</tbody>
</table>

to send waypoint commands to the onboard computer over Wi-Fi via the ROS messaging system. During semi-autonomous sections of flight, velocity commands were sent to the onboard computer by a human operator using a wireless Microsoft Xbox controller. At all times, a human safety pilot had a direct RC link to the Pixhawk autopilot to override attitude commands from the onboard computer if necessary. Safety pilot intervention was not required during the flight tests described in this paper.

The following three flight tests demonstrate autonomous MAV navigation in a variety of challenging unknown environments using the relative navigation framework. All perception, estimation, control, and mapping was performed onboard the vehicle and in real time. Estimation and control were performed at the rate of the IMU measurements, which was 100 Hz. Visual odometry was performed at 15 Hz using the RGB-D camera, and laser-scan matching was performed at 40 Hz. No adjustments or tuning were required to prepare the vehicle for the different scenarios other than choosing between the RGB-D camera and laser scanner, illustrating that the framework does not make environment-specific assumptions. The flight tests are described in the following sections, and are summarized in Table 5.2. A discussion of the results demonstrated by these flight tests is given in Section 5.7.

5.6.1 Flight test 1: Outdoor GPS-denied

In the first flight test, the vehicle flew a trajectory around the perimeter of a large building, marked in black in Figure 5.17. The flight lasted 9 min, and the total distance traveled was 320 m. For this flight test the system obtained visual odometry from the RGB-D camera. A human operator provided velocity setpoints to the vehicle through the
Figure 5.13: Flight test 2. The vehicle started at the blue circle moving clockwise, following the blue path, red path, yellow path, and then purple path. The vehicle flew in the middle of the hallway and was facing the direction of motion except for the path indicated by purple dots when the vehicle traveled backwards.

Xbox controller. Because the MAV flew within a few meters of the building throughout the flight, reliable GPS measurements were not available. Because the vehicle did not revisit any portion of the flight path, loop-closure constraints were also unavailable.

5.6.2 Flight test 2: Indoor GPS-denied

This flight test was conducted indoors through a series of hallways. The flight path of the vehicle is overlaid on the floor plan of the building in Figure 5.13. The flight lasted 12 min, and the total distance traveled was 390 m. Visual odometry was obtained using the RGB-D camera. The odometry was of high quality throughout most of the flight, but its accuracy degraded in the southeast corner when the camera was pointed at a blank wall. A human operator provided velocity setpoints to the vehicle using the Xbox controller to guide the vehicle through the hallways. A total of 139 loop closures were detected using the RGB-D
camera. This flight test was originally attempted in Ref. [28]; however, the trajectory flown was significantly shorter, no loops were closed, and the back-end place recognition, map optimization, and global path planner had not yet been implemented.

Figure 5.14 shows the vehicle flying down one of the hallways. The hallways were relatively nondescript, with few visually interesting features. Despite this, the odometry and place recognition algorithms performed well. Another challenge presented by the hallways was their narrow width; the hallways ranged between only 1.8 m and 2.5 m wide, as compared to the 1.1 m total diameter of the vehicle. The narrow confines produced significant aerodynamic ground and wall effect. To highlight the significance of this effect, a highly experienced safety pilot attempted to fly the trajectory in attitude stabilized mode via RC control, and struggled to maintain stability in the wider hallways to the extent that flying in the narrower hallways was unfeasible.

5.6.3 Flight test 3: Indoor/outdoor intermittent GPS

The third flight test consisted of two loops through both indoor and outdoor environments through and near a building. This flight test incorporated loop closures, intermittent degraded GPS, and autonomous path planning and flight into a single experiment. The
Figure 5.15: Flight test 3. The vehicle started at the blue circle, moving clockwise, following the blue path, red path, and then yellow path. The vehicle was facing the direction of motion. Purple indicates regions of autonomous waypoint following and black indicates the doorways.

flight lasted 9 min and traveled a distance of 240 m. The path that the vehicle flew is overlaid on a satellite image of the building in Figure 5.15. The vehicle started inside the southeast wing of the building, flew through the courtyard into the northeast wing of the building, down the alleyway to the east of the building and back into the southeast wing, then repeated the same path. In all, there were four transitions from indoor to outdoor, and four transitions from outdoor to indoor. These transitions are commonly troublesome for GPS-degraded navigation approaches because odometry algorithms can sometimes degrade and GPS accuracy can vary significantly through the transition.

Odometry was obtained from the laser scanner, while loop closures were obtained using the RGB-D camera. The flight test was conducted at night, so loop closures were obtained only in the well-lit indoor portions. In all, 30 loop closures were detected. Due to the close proximity to the buildings, GPS updates were very limited. GPS measurements were
gated until the GPS receiver’s self-reported accuracy estimate dropped below a reasonable threshold. As a result, all GPS measurements were gated until the second time the vehicle flew down the alley between buildings. Even then, only ten GPS updates were received, and these updates were biased to the north by about two meters.

During the first loop, the vehicle was guided by velocity setpoints provided by a human operator using the Xbox controller. After the first loop-closure constraints were detected and the map was optimized to remove drift, fully autonomous waypoint following was demonstrated. A human operator clicked on a previously visited point on the map, and the vehicle retraced its previous path to arrive at the desired waypoint. Three of these fully autonomous segments were carried out, marked in purple in Figure 5.15, including one during an outdoor to indoor transition.

In addition to the results presented in this paper, this same indoor/outdoor flight was also performed a second time during the day using the RGB-D camera instead of the laser scanner. The alternate odometry source produced comparable front-end estimation and control, introduced 45 loop-closure constraints, successfully incorporated 36 GPS measurements, and performed four autonomous waypoint missions. This helps to highlight the modularity and extensibility of the relative navigation framework. We chose to present the laser scanner results because they demonstrate the use of a different odometry source than that used in the other two flight tests.

5.7 Results

This section discusses the results from the flight tests described in Section 5.6 as they relate to various aspects of the relative navigation architecture. In general, these results demonstrate that the proposed architecture runs onboard the vehicle in real time, and that it enables missions involving real vehicles in realistic environments. The results show that the system is able to operate in both indoor and outdoor environments, and handle transitions
Figure 5.16: Pose-graph map for the first 130 m of flight test 2. At this point the first loop closure (red) was detected and used to improve the global map without affecting local stability. Before optimization, the global pose estimate created by compounding relative edges had accumulated 1.8 m of drift. Notably, no tweaking or tuning of the system was required between the flight tests other than choosing which sensor (the RGB-D camera or laser scanner) would be used for odometry. This demonstrates that the architecture does not make environment-specific assumptions, and that it is not tied to one particular sensor.

Section 5.7.1 discusses the estimation accuracy and consistency from using the relative navigation approach. Section 5.7.2 discusses the performance of the pose-graph optimization, and Section 5.7.3 discusses the capabilities for autonomous flight demonstrated by the tests.

5.7.1 Estimation Accuracy and Consistency

Figure 5.16 shows the pose-graph map for the first 130 m of flight test 2. Up to this point no loop closures had been detected, meaning that the pose graph simply compounds the relative edges produced by the front-end estimator to reconstruct the global pose without...
Figure 5.17: Pose-graph map for flight test 1. Heading errors cause the position uncertainty to grow. The global back end compounds the small, Gaussian edge covariances to form banana-shaped uncertainty estimates that correctly represent the underlying uncertainty. The 90 percent confidence regions are shown for several instances throughout the trajectory.

any additional optimization. The accuracy of this global pose therefore directly reflects the accuracy of the front-end estimator. Figure 5.16 shows that only 1.8 m of drift were accumulated in the first 130 m of flight, yielding a drift rate of 1.4 percent per distance traveled. For the 139 loop closures in flight test 2, the maximum drift rate was 1.5 percent with an average drift rate of 0.85 percent. For the 30 loop closures in flight test 3, the maximum drift rate was 2.8 percent with an average of 1.8 percent. The overall accuracy of an approach depends on the environment, quality of sensors and calibration, and sophistication of odometry algorithms. These flight tests highlight that RN facilitates good performance with off-the-shelf algorithms and sensors.

Another advantage of the pose-graph representation is that it accurately captures the uncertainty in the global pose of the vehicle. Approaches that estimate the global pose directly in the filter represent the uncertainty as a Gaussian normal distribution characterized by its covariance matrix, which produces an ellipsoidal confidence region. Yet, it has been shown that the true uncertainty distribution produced as a vehicle moves through the environment with uncertainty in its heading is a banana-shaped distribution [18], which is a Gaussian distribution expressed in exponential coordinates [50]. A pose graph represents the global
pose as a sequence of short transforms, each with an associated ellipsoidal uncertainty. It was shown in Ref. 40 that this series of uncertainties can be combined to produce a total uncertainty estimate that is an excellent approximation to the true banana-shaped distribution. Therefore, the pose-graph representation contains all of the information that is necessary to produce an accurate estimate of the global pose uncertainty. Figure 5.17 shows the 90 percent confidence regions created from the pose graph at several points using the method in Ref. 8. This method samples from the individual edge covariances in a Monte-Carlo fashion, then fits a Gaussian distribution in exponential coordinates to the resulting distribution of final pose estimates.\(^3\) As can be seen, the resulting distributions correctly capture the banana shape of the true uncertainty distribution. In addition, at every point along the trajectory, the 90 percent confidence region captures the true location of the vehicle. This demonstrates that the uncertainty estimate in the global pose reconstructed using the pose graph is consistent. More details on the consistency of the relative navigation approach, and how it compares with other state-of-the-art methods, are given in Ref. 8.

### 5.7.2 Map Optimization

Figure 5.18 shows the pose-graph optimization results for flight test 2. Figure 5.18a shows the unoptimized map produced by compounding the relative front-end pose estimates. These odometry edges are represented by the blue lines with keyframes marked as dots, and loop closures detected between keyframes are represented by red lines. Over the course of the 390 m flight, several meters of drift accumulated so that the resulting map lies outside of confines of the hallway where the vehicle actually flew. Figure 5.18c shows that after the map has been optimized, this drift has been removed and the estimates of the vehicle’s global trajectory lie within the hallways. The complete optimization took seven iterations to converge and took less than 8 ms running onboard the vehicle during flight.

---

\(^3\)Individual edge covariances can also be combined using the fourth-order analytical approximation presented in Ref. 40.
During flight test 2, the place recognition algorithm did not produce any false-positive loop-closure detections. This is particularly impressive given the fairly uniform appearance of the hallways that the vehicle flew through (see Figures 5.8 and 5.14). To demonstrate the impact that false-positive loop-closure constraints can have, and to demonstrate the ability of the robust optimization algorithm to detect and reject these spurious constraints, three false-positive loop-closure constraints were artificially introduced to the pose graph. These are shown in yellow in Figure 5.18a. Figure 5.18b shows the optimized pose graph obtained by a non-robust optimization algorithm that naively incorporates the false-positive constraints. The three false constraints have a drastic impact on the accuracy of the optimized map, even though there are 139 valid loop closures constraining the map. Figure 5.18c, on the other hand, demonstrates the effectiveness of dynamic covariance scaling in correctly detecting and rejecting the false-positive loop closures to produce a highly accurate optimized map.

The unoptimized pose-graph map for flight test 3 is shown in Figure 5.19a. As with flight test 2, the relative edges from the front-end estimator are shown in blue, and the loop-closure constraints are shown in red. Again, no false-positive loop closures were detected during this
Figure 5.19: Pose-graph map for flight test 3. (a) Trajectory (blue) before incorporating loop closures (red) and GPS measurements (green). For plotting purposes, the GPS is plotted relative to the first received GPS measurement. (b) After incorporating the ten available GPS measurements (green), the trajectory is globally localized. Black indicates the doorways. Note that because all of the available GPS measurements were slightly biased to the north, the final map is also biased.
flight test. In addition to loop-closure constraints, flight test 3 introduces intermittent GPS measurements. The ten valid GPS measurements are plotted as green points in Figure 5.19a, and the corresponding edges in the graph are represented by green lines. As described in Section 5.5.3, the GPS constraints were defined with respect to the first GPS measurement, which is plotted at the origin. The final optimized map incorporating both loop closures and GPS measurements is shown in Figure 5.19b. While truth is not available, the accuracy of the final map can be evaluated by comparing it to the satellite image of the building. The doors of the building that the vehicle flew through are marked as black lines in Figure 5.19b. Due to the challenging urban canyon environment, all of the GPS measurement were biased to the north by a few meters, and so the resulting map is also biased to the north. Correcting for this bias, however, it can be seen that the optimized trajectory passes through each of the doors and matches the path that the vehicle actually flew.

One important result that this flight test demonstrates is the ability of the relative navigation architecture to perform delayed localization using few GPS points. Before the first GPS measurement is received, the map is metrically consistent with respect to the starting location of the vehicle, but is not localized globally. In other words, the vehicle knows where it is relative to its starting point, but has no knowledge of where it is in the world. This unlocalized map, however, is still sufficient for navigation purposes, and the vehicle was able to fly autonomous waypoints before it received GPS measurements. When the vehicle received the first GPS measurement, however, it was able to pin the map to a location in the world. Subsequent measurements allowed it to orient the map and refine its position estimate. For flight test 3, this localization did not occur until several minutes into the flight. In addition, the localization is accomplished using few (only ten) GPS measurements. This is significant in the context of other GPS-degraded approaches that require GPS for a prolonged (the first 80 seconds of flight) initial alignment phase [35] or have GPS for a majority of their flight [23].
5.7.3 Autonomous Flight

A basic requirement for autonomous flight is robust and stable control of the vehicle. While difficult to quantify, the robustness of the relative navigation architecture is demonstrated by the scope of flight tests presented in this paper. For example, flight test 2 demonstrates smooth, stable flight down narrow hallways that produce significant aerodynamic ground and wall effect. The high-rate feedback control and accurate relative state estimates facilitated missions that would be unfeasible for experienced human pilots. In flight test 3, the vehicle smoothly transitions through eight doorways. Between the three flight tests presented, the platform was flown for almost a kilometer through congested environments without incident. Throughout the flight tests, the control performance did not suffer from the resetting of the relative states.

A unique advantage of relative navigation that is demonstrated by the flight-test results is the architecture’s innate ability to handle jumps in the global-state estimate. For example, the pose-graph optimization at the first loop closure in flight test 2 resulted in a global state jump of 1.8 m, and the optimization at the first loop closure in flight test 3 resulted in a jump of 2.3 m. In addition, the first GPS measurements received in flight test 3 caused a large state jump as the map was rotated counterclockwise by 90 deg and translated approximately 28 m when it was first localized globally. Despite these large state jumps, the control of the vehicle did not suffer at all because, as described in Figure 5.10, control is carried out in the relative frame and insulated from global state jumps by the path planner. Conceptually, this allows the MAV’s perception of the local environment to remained fixed while the global map shifts beneath it.

In addition to smooth local control, flight test 3 also demonstrated autonomous global navigation. After the first loop closures were received and the drift in the map was removed, a waypoint was provided by an operator clicking on a previously visited location on the gen-
erated pose graph. The vehicle then autonomously followed the map back to this location. Autonomous waypoint following was demonstrated three times, traveling 35 m through congested environments including during an outdoor to indoor transition. The regions where this took place are highlighted in purple on Figure 5.15. The final waypoint was selected after GPS measurements were incorporated into the pose-graph map. The user, by selecting a pixel on an ortho-rectified image, was effectively establishing a desired GPS waypoint for the vehicle. Of note, this global waypoint was located indoors. The vehicle then autonomously navigated to that global waypoint and stabilized its position. This result is particularly compelling because the vehicle correctly stabilized itself about a global waypoint despite being in a GPS-denied environment.

5.8 Conclusion

Developing dependable, autonomous MAV solutions that are robust to GPS degradation is a challenging but highly relevant field of research. This paper demonstrates that the relative navigation framework offers a compelling alternative paradigm for approaching the problem. By decoupling flight-critical estimation, guidance, and control algorithms from unobservable global states that are prone to inconsistency and state jumps, relative navigation avoids many issues that plague other state-of-the-art approaches.

This paper presents the details necessary to implement the complete relative navigation framework, including resetting the relative estimator to ensure observability and adapting existing view-matching, path planning, and control algorithms for reliable, smooth flight. We describe how to leverage pose graphs to opportunistically incorporate loop-closure and GPS constraints, and outline how the high-level path planner facilitates autonomous missions while insulating the vehicle from the negative effects of global state jumps.

Through a series of prolonged flight tests, we demonstrated the effectiveness of the relative navigation approach for autonomous GPS-degraded MAV navigation in varied, unknown en-
environments. We showed that the system can utilize a variety of vision sensors, works indoors and outdoors, runs in real-time with onboard processing, and does not require special tuning for particular sensors or environments. We demonstrated stable front-end performance with low drift while leveraging off-the-shelf sensors and algorithms. We further demonstrated the onboard generation of a globally-consistent, metric, and localized map by identifying and incorporating loop-closure constraints and/or intermittent GPS measurements. With this map, we demonstrated the fully autonomous completion of mission objectives, including performing a position-hold about a GPS waypoint while in a GPS-denied environment.

One of the most important aspects of the relative navigation architecture is that it does not make any assumptions about a particular platform, sensor suite, environment, or use case. Many existing systems could be readily modified to fit within the relative navigation framework, and thereby benefit from its theoretical and practical advantages.

Acknowledgments

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547, along with significant contributions from C-UAS industry members. This work was also supported in part by Air Force Research Laboratory Science and Technology (AFRL S&T) sponsorship. Daniel Koch was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. The authors would like to thank Kevin Brink of the Air Force Research Laboratory Munitions Directorate (AFRL/RW) for his support of this project and for his valuable insights.
Chapter 6

Cushioned Extended-Periphery Avoidance: a Reactive Obstacle Avoidance Plugin\(^1\)

6.1 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 6.1, 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 6.1).

\(^1\)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 [76]
Figure 6.1: 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.

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 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 [97].

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 [98]. 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 [99], 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 [100], 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 [101] and obstacle avoidance [102]. 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 [103]. 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 6.2 and propose a new reactive algorithm, Cushioned Extended-Periphery Avoidance (CEPA) in Section 6.3 as a specific implementation of this framework. We present simulation and hardware results of CEPA and the ROAP architecture in Section 6.4 and conclude in Section 6.5.

6.2 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 6.2. 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
Figure 6.2: 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.

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 [104–107].

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 6.1, where in general, avoiding collisions and maintaining stable flight is of paramount importance. This is particularly relevant 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) [104] 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 [105], stressing the importance of low computation requirements. Schopferer et al. has presented a novel decoupled iterative planning method [108] 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 [107] 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.
6.3 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 [109, 110]. 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 [107], 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 6.2. 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.

### 6.3.1 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 \( \mathbf{\hat{v}} \), expressed by

\[
k_1 \left( \mathbf{v}^\top \mathbf{\hat{v}} \right) + k_2 \frac{\| \mathbf{\hat{v}} \|}{\| \mathbf{v} \|}.
\]  \hspace{1cm} (6.1)

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 6.3, 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
Figure 6.3: An example steering configuration. \( \mathbf{v} \) is the obstacle-laden goal vector supplied by the path planner. CEPA identifies \( \hat{\mathbf{v}} \) 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.

\( r \) at specified bearing angle \( \phi \) is defined analytically as

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

where \( d = \|\hat{\mathbf{v}}\| T \) is the look-ahead distance and \( \gamma = \text{atan2}(d, r) \). Note that Equation 6.2 assumes \( \hat{\mathbf{v}} \) 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

$$\Omega (\tilde{v} | LS) = \sum_{i}^{N} \kappa (\phi_i | \tilde{v}, LS),$$

(6.3)

where

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

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

A weighted sum of Equations 6.1 and 6.3 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:

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

6.3.2 Map Memory

Applying a command in a direction that is not currently observed is inherently presumptuous. Previous ROAP algorithms [104–106] 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
Figure 6.4: 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 $180^\circ$ laser scan (blue-solid), appending previous laser scans gives the MAV a limited $360^\circ$ understanding about the entire shaded area and allows the MAV to safely move backwards.

$360^\circ$ understanding of the environment, as illustrated in Figure 6.4. 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.

6.3.3 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 6.1, the command provided by the map-based path planner is temporarily ignored as emergency action is taken.
Figure 6.5: 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 \( \vec{v} \).

As illustrated in Figure 6.5, the periphery-enhanced 360-degree obstacle map is filtered such that

\[
\frac{d\rho}{d\phi} \leq K,
\]

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 \( \vec{v} \). 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.

6.4 Experimentation and Results

CEPA was implemented in ROS [95] and tested in a Gazebo simulator, adapted from [111], 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 [112] was used to control the system. Yaw was controlled with an under-damped proportional controller, 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, $K = 0.01$, $r_{LB} = 0.55$ m, $r_{UB} = 1.0$ 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 [104] 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.

6.4.1 Simulation Results

Two tightly-constrained environments were used to validate the algorithm. The first environment, shown in Figure 6.6 consists of a dense grid of cylinders requiring tight maneuvering. While the high-level path planner commands the MAV directly towards the
Table 6.1: Simulation results for scenario 1

<table>
<thead>
<tr>
<th></th>
<th>FFLAO</th>
<th>CEPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion Rate</td>
<td>0.2188</td>
<td>0.9863</td>
</tr>
<tr>
<td>Average Duration (s)</td>
<td>71.61</td>
<td>63.54</td>
</tr>
</tbody>
</table>

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 6.1.

As can be seen from Table 6.1, 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 recently closed door, as shown in Figure 6.7. 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
Figure 6.6: 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.

In 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.

6.4.2 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 less than 1/16 of the available processing time. State estimation was performed using the Relative Multiplicative Extended Kalman Filter described in [33] provided...
Figure 6.7: 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.

with position measurements from an RGB-D visual odometry algorithm described in [48]. 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 it 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 6.8. The high-level path planner continuously provided commands at 0.8 m/s directly towards 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, and could no longer directly see the first obstacle, it responded correctly by
Figure 6.8: Hardware validation of CEPA in a GPS-denied environment using strictly onboard computation and sensing.

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/35Og9PYwX0I.
6.5 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.

Acknowledgements

This research was supported by the NSF Center for Unmanned Aircraft Systems (C-UAS), and Brigham Young University.
Chapter 7

Relative Navigation in GPS-Degraded Environments

7.1 Introduction

As processing, sensing, and battery technologies continue to develop there are increased opportunities for unmanned air vehicles (UAVs) to contribute to society. Emerging applications include fire surveillance, search and rescue, infrastructure and agriculture monitoring, and the delivery of medical supplies to remote locations [113–116]. However, the majority of these applications require additional technology development and will likely be restricted to less populated environments. The integration of autonomous aircraft into mainstream life will depend in large part on the ability to safely and effectively operate in varied environments and with varied tasking. For example, current autonomous systems typically require external sensing or computation, such as a motion capture system, Global Positioning System (GPS) localization, a priori maps of the environment, or at least offboard sensor fusion and decision making. Other systems are accompanied with strong, limiting assumptions, such as a highly structured environment (vertical walls, flat floors, stationary scenes, etc). Small UAVs are also limited by size, weight, and power (SWaP) constraints. While autonomous flight is currently possible in specialized circumstances, the development of robust, real-time, onboard methods for autonomous control in cluttered, non-structured environments without external or a priori information remains an open field of research.

1 This paper was written by David O. Wheeler, Paul W. Nyholm, Daniel P. Koch, Gary J. Ellingson, Timothy W. McLain, and Randal W. Beard, and published in the Encyclopedia of Aerospace Engineering [1].
Table 7.1: Sources of GPS uncertainty

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-path</td>
<td>Signal bounces before reaching receiver (false pseudo-range).</td>
</tr>
<tr>
<td>Number of satellites</td>
<td>Few visible satellites increase sensitivity to timing errors.</td>
</tr>
<tr>
<td>Dilution of precision</td>
<td>Visible satellites are poorly spaced.</td>
</tr>
<tr>
<td>Spoofing</td>
<td>Signal is locally recreated with false information.</td>
</tr>
<tr>
<td>Atmospheric delays</td>
<td>Signal is delayed due to ionosphere and troposphere influences.</td>
</tr>
</tbody>
</table>

A noted obstacle in reaching the navigation robustness necessary for the integration of UAVs in the national airspace is the heavy reliance on GPS. 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” [4]. GPS not only provides global position estimates to constrain the drift introduced by noisy rate sensors like MEMS-based inertial measurement units (IMUs) but also provides a way to estimate ground speed and orientation, allowing the vehicle to estimate wind effects. A robust navigation solution cannot depend on accurate GPS measurements due to the varied sources of uncertainty presented in Table 7.1. Further, GPS measurements may be unavailable in the presence of GPS jammers, when shadowed by buildings or foliage, and simply cannot be applied indoors. Various sources report that GPS loss, even for a brief period, often results in catastrophic failure. As a result, GPS-denied navigation has become a strong emphasis of research over the last decade.

One important approach to GPS-denied navigation is known as simultaneous localization and mapping (SLAM). This approach involves estimating the vehicle’s state relative to a local frame by creating or adding to a map and localizing the vehicle within that map. This method allows for the direct use of relative measurements, such as visual odometry (VO), to estimate the vehicle’s state. This article will serve as a tutorial outlining the basic components of a SLAM-based relative navigation solution. Although the framework is applicable to other airframes and implementations, several of the relative navigation framework modules have
Figure 7.1: The multirotor vehicle used for the example implementation in this paper. Details on the specific hardware are given in Table 7.2

<table>
<thead>
<tr>
<th>Table 7.2: Hardware details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
</tr>
<tr>
<td>Vehicle</td>
</tr>
<tr>
<td>Autopilot</td>
</tr>
<tr>
<td>RGB-D Camera</td>
</tr>
<tr>
<td>IMU</td>
</tr>
<tr>
<td>Ultrasonic Altimeter</td>
</tr>
<tr>
<td>Processor</td>
</tr>
</tbody>
</table>

been implemented on a multirotor aircraft and are presented as examples. Figure 7.1 and Table 7.2 describe the multirotor platform used in these implementations. Section 7.2 is an overview of the framework while Sections 7.3 and 7.4 explain in detail the components of the relative front end and global back end. Conclusions are presented in Section 7.5.

7.2 Relative Navigation Framework

Just as the driver of an automobile is the safest as he or she focuses on the road ahead of them rather than on their map, GPS, or communication devices, the relative navigation approach (see Figure 7.2) uniquely decouples the relative, in-flight control from less critical global updates. Sections 7.2.1 and 7.2.2 outline the relative navigation framework, while Section 7.2.3 presents several scenarios that prove challenging for non-relative frameworks.
7.2.1 Relative Front End Overview

The core of the front end is a state estimation scheme (Section 7.3.2) that fuses sensor data, generally from a high rate IMU and infrequent exteroceptive sensors, such as an altimeter or a camera for visual odometry (Section 7.3.1). The state is estimated relative to a local, gravity-aligned coordinate frame (within about one meter for a multirotor vehicle) known as a node frame. As the vehicle moves, new node frames are established and the transformation between node frames is stored as an edge (see Figure 7.3). A local path planner (Section 7.3.3) uses sensor data for obstacle avoidance and route planning and provides inputs to the aircraft’s control (Section 7.3.4).

7.2.2 Global Back End Overview

Decoupled from the flight-critical front end is the need for a globally accurate map. The back end, represented as a pose graph, is seeded with the node frames and edges created by the front end (Section 7.4.1). To eliminate accumulated drift error, place recognition algo-
Figure 7.3: 2D illustration of estimated and true states relative to the current node frame. Roll and pitch (not shown) are inertially defined because the node frame z-axis is aligned with gravity. Position ($p_x$, $p_y$, $p_z$) and yaw ($\psi$) are defined relative to the current node frame, $k$. The state and covariance are estimated as usual with respect to this node frame. When a new node frame is declared, the estimated $p_x$, $p_y$, $p_z$, and $\psi$ along with the associated covariance are saved as an edge in the global back end pose graph. Node frame $k+1$ is then defined as the current true state (with the roll and pitch removed), and the filter zeros $p_x$, $p_y$, $p_z$, and $\psi$ and their corresponding covariance values. The result is that the state error and uncertainty are removed from the front end and delegated to the global back end.

Algorithms efficiently compare the current image with all previous images (Section 7.4.2). When the vehicle returns to a previously visited location, known as a loop closure, an additional constraint is introduced to the set of edges, thereby over-defining the map. Optionally, if GPS measurements are available, they can be added in the back end as constraints (Section 7.4.3). As desired, a robust optimization step minimizes constraint errors introduced by loop closures, GPS information, and odometry estimates (Section 7.4.4). With these refinements, the global map is then used by the high-level planner to fulfill global missions through a series of relative goals (Section 7.4.5). Subsequent loop closures are used to refine and improve the global map, making it sufficient for persistent, repeatable navigation. This global framework only interacts with the aircraft by influencing what low-level, relative goals are introduced, providing safer, more robust control in GPS-degraded environments.

7.2.3 Motivating Scenarios

While working with respect to a single, inertial reference frame makes intuitive sense, the following scenarios highlight advantages to be gained from a relative framework. In each example the UAV is assumed to be controlled with respect to its globally estimated position:
• A UAV loses GPS signal and receives IMU measurements only for several minutes. Upon reacquiring GPS, the state estimate jumps drastically and the plane is unable to recover.

• A multirotor vehicle moves from indoors to outdoors. Upon acquiring GPS signal for the first time its global state estimates, with respect to an arbitrary origin inside the building, will jump drastically.

• After a loop closure, a UAV’s current estimated global state may jump significantly resulting in sudden, unintentional, and unpredictable vehicle motion.

• After flying for some time the size of the optimization problem delays any updates to the UAV’s estimated global state. The vehicle’s control suffers as a result.

• A vehicle receives an erroneous loop closure or GPS measurement. The estimated global state degrades without a method to later remove the effects of the outlying measurement.

7.3 Relative Front End

The principal components of the relative front end, as introduced in Section 7.2.1, are explained in greater detail below:

7.3.1 Visual Odometry

Visual odometry (VO) is the process of computing the motion of a camera by comparing the captured images [117]. VO algorithms fall into two general categories: appearance-based and feature-based. Appearance-based methods use information from all pixels in the images to compute motion, while feature-based methods use visually distinct features in the environment, such as corners, that are tracked from one image to the next. VO can be implemented using either monocular or stereo cameras, and more recently has been
implemented using RGB-D cameras (e.g. [118]). In addition to providing a standard color image, RGB-D cameras, like the Microsoft Kinect sensor, provide a depth image that encodes how far the object imaged at each pixel is from the camera. With monocular VO the change in orientation of the camera can be computed, but the translation can be computed only up to an unknown scale factor. The depth information provided by stereo and RGB-D cameras enables the calculation of this scale factor.

Some VO algorithms compute the camera’s motion between consecutive images while others find the transformation to a chosen reference image, referred to as a key frame. While most VO algorithms could be used in the relative navigation framework, a key-frame approach fits best with the pose graph SLAM paradigm. A new key frame is chosen once the vehicle has moved far enough that there is insufficient overlap between features in the current and key frame images. The key frame approach results in low drift in the VO estimates, especially for a hovering multirotor vehicle.

Good general tutorials on visual odometry are found in [19, 20]. Additional approaches for VO using RGB-D cameras include those presented in [48, 118], and a comparison between various methods can be found in [119]. Approaches utilizing laser scanners have also been widely explored in the literature (e.g. [21, 120]), and techniques using point clouds from LiDAR or other sensors have been explored as well (e.g. [121]). Increased UAV robustness is possible when relative measurements are incorporated from multiple sensors [23, 122].

7.3.2 Estimation

To maintain stable flight, a filter must fuse available sensor data to provide robust estimates for attitude and velocity. Furthermore, many high-level goals require position to be estimated reliably. Several probabilistic methods have been developed to fuse a motion model with intermittent measurements for vehicle state estimation, including the complimentary filter, the particle filter, and the Kalman filter, where the latter is the most prevalent for
UAV platforms. Due to the nonlinearity of UAV dynamics, the state uncertainty is often propagated by either using a second-order linearization by applying the analytic Jacobian of the dynamics (Extended Kalman Filter), or by applying the nonlinear update to sigma points to reconstruct the state uncertainty (Unscented Kalman Filter [123]). While the following description outlines a Kalman filter approach, any estimation scheme could be used.

What most distinguishes the relative navigation approach from its conventional global counterparts is the decoupling of local state estimation from global states. Position and yaw states are defined relative to a local, gravity-aligned node frame. As illustrated in Figure 7.3, state estimates will drift from truth with the uncertainty estimated by the Kalman filter’s covariance matrix. After moving a small distance, the current true state (with pitch and roll removed) is declared to be the origin of a new relative coordinate frame. The estimated position \((p_x, p_y, p_z)\) and yaw \((\psi)\), together with their associated covariance, are saved as an edge in the global back-end pose graph representing the estimated transformation between node frames with some uncertainty. These states are then replaced with states relative to the new node frame. Since the position and yaw states with respect to the new coordinate frame are now exactly zero, the associated covariance values are set to zero. This process can be thought of as augmenting and subsequently marginalizing the filter’s states. The accumulated error and its accompanied uncertainty is effectively removed from the front-end filter and delegated to the back-end pose graph.

A multiplicative extended Kalman filter (MEKF) implementation of this relative estimation framework is presented in [28] and [33]. The filter estimates relative position, attitude (with relative yaw), body-fixed velocity, and gyroscope and accelerometer biases. The attitude of the vehicle is represented by a quaternion. While quaternions require four scalars to define a three degree of freedom rotation, they are computationally more efficient than Euler angles and avoid the singularity known as gimbal lock. When using quaternions however, the estimated attitude error, found in the typical update step of an extended Kalman filter
Figure 7.4: Performance plots for a MEKF implementation using the relative navigation framework. The estimated forward and yaw states (blue) are compared to truth (green). The discontinuities indicate a change in node frame.

(EKF), cannot simply be added to the current state estimate. A common approach is to multiply the attitude quaternion by the estimated attitude error, maintaining the quaternion norm and earning the term multiplicative EKF (MEKF) [53]. The underlying dynamics of the filter make use of the typical kinematic relationships [124] coupled with an enhanced rotorcraft drag model presented in [64]. The velocity estimates are constrained by including, in conjunction with accelerometer measurements, drag terms that are proportional to the body-fixed forward and right velocities. In this implementation, gyroscope measurements are considered as inputs to the system (mechanization), while accelerometer measurements are used as updates. A feature-based VO for a forward-facing RGB-D camera, using FAST features [125] with BRIEF descriptors [126] was used (see [28] for details). Flight tests were performed using the hardware outlined in Table 7.2 and the MEKF described above. Figure 7.4 compares the MEKF’s estimated forward and yaw states with truth as measured by a motion capture system. The discontinuities occur as the vehicle transitions from one node frame to the next.
7.3.3 Low-Level Path Generation and Following

The autonomous vehicle must be able to maneuver relative to the local environment without reliance on global state information. This is accomplished as local obstacles are determined from sensor information and goals with respect to the local node frame are received from the high-level path planner (Section 7.4.5). Low-level path planning typically takes the form of limited obstacle avoidance, while the global path is calculated by the high-level path planner.

For example, one approach when using a LiDAR or RGB-D sensor is to simplify the path generation problem onto the 2D horizontal plane at the elevation of the vehicle. Obstacles are identified from the most recent depth information and saved with respect to the current node frame. A 2D path is defined relative to the current node frame that progresses towards the goal while avoiding obstacles. Obstacles, goal locations, and the current path are transformed into each new node frame. A path following approach, such as the one described in [124], can be used to provide a desired state to the controller.

Other approaches may be more appropriate given different vehicles and sensor information and may be extended beyond 2D. For example, if a fixed-wing aircraft is flying with a LiDAR then paths may be generated that account for the vehicle dynamics to avoid obstacles (e.g. [127]). This includes Dubins paths [124, 128] and vector fields for path following. A 3D multirotor approach could include biologically inspired steering algorithms, where obstacles are identified in the local frame and trajectories are chosen based on distance and bearing to obstacles and goals [104]. Other maneuver-based planning techniques generate a series of poses to smoothly navigate through the observed local environment while avoiding obstacles [129].
Figure 7.5: Control system architecture for the example implementation. The position estimation and control run on the onboard computer, while attitude estimation and control run at a higher rate on the autopilot. In this figure $x^c$ is the commanded position, $u$ is the motor commands, and $(\phi^c, \theta^c, \psi^c, T^c)^T$ are the roll, pitch, yaw rate, and thrust commands sent from the onboard computer to the autopilot.

7.3.4 Control

Control approaches for both fixed-wing and multirotor vehicles have been widely explored in the literature and will not be reviewed in detail here. One of the key differences between a controller implementation for a relative framework to its global counterparts is that the desired states of the vehicle are expressed in the local node frame rather than in the global frame. However, because the actual state of the vehicle is also expressed in the local frame, the error between the actual and desired states will be the same as if both were expressed in the global frame. Therefore many controllers designed to work in a global framework will also work well within the relative framework.

The position controller used in the example implementation for this paper is adapted from the controller detailed in [112], while attitude stabilization and control is achieved using the standard PID loops implemented on the autopilot. The position estimation and control run at approximately 100Hz on the onboard computer, while the attitude estimation and control run at a higher rate on the autopilot. The outputs of the position controller are...
roll, pitch, yaw rate, and thrust set points that become the inputs to the attitude controller. This architecture is illustrated in Figure 7.5.

7.4 Global Back End

The principal components of the global back end, as introduced in Section 7.2.2, are explained in greater detail below.

7.4.1 Pose Graph

To maintain a spatially, or globally consistent map, the back end stores information about the vehicle’s trajectory as a pose graph. A pose graph is a graphical representation of a vehicle’s path that encodes global vehicle pose estimates as graph vertices and the relative transformation between two poses as graph edges. A pose graph representation is effective because efficient graph optimization algorithms have been thoroughly developed and can be applied to refine pose estimates; additionally, the graph conveniently serves as an abstract map that can be used for 3D and human-readable map construction as well as high-level path planning.

Figure 7.6 depicts how a pose graph is constructed from vehicle odometry. In a relative framework, a pose graph vertex represents the global estimate of a node frame’s translation and orientation. The graph edge connecting two consecutive poses encodes the relative transformation between two node frames and its respective covariance. The front-end estimator provides relative transformation edges to the back end. The transformations are compounded with all previous transformations to estimate the global position of the new node frame and a corresponding vertex is added to the graph.

A pose graph is also capable of encoding measurements other than odometry. Loop closure measurements introduce edges between two existing, non-consecutive nodes (Section
Similarly GPS measurements can be added as edges between the vertex where the measurement was received and the origin of the global coordinate system (Section 7.4.3). Since Lu and Milios’s seminal work in [130], pose-graph SLAM has become the predominate method of SLAM in the literature. An excellent tutorial on pose-graph SLAM is found in [41].

### 7.4.2 Place Recognition

A vehicle’s global position estimate will drift significantly over time in the absence of global updates. A common solution in the SLAM literature makes use of place recognition (PR). In place recognition the current image is compared with previous images to determine if the vehicle has returned to the same location. While a naive solution would require significant computation and memory, not generally available with the size, weight, and power (SWaP) constraints of a UAV, the computer vision community has developed efficient vocabulary-based PR algorithms. As with VO algorithms, distinct image features are represented by mathematical descriptors. Using a large, representative training dataset of images, the most prominent, distinct feature descriptors are saved offline and referred to as words in a vocabulary. During flight, any image can be succinctly represented by the set of nearest vocabulary words found in the image. Images can be quickly compared using word occurrences, similar to many search engine algorithms [131]. Further work has improved place recognition performance in the presence of aliasing, where high correlation is found on non-correlated images, like pictures of brick walls [132].

In a relative navigation approach, each key frame image (Section 7.3.1) is passed through the place recognition software and archived. After a match is found, the images are recalled and visual odometry methods provide the estimated transformation from one node frame to another. The transform is communicated to the pose graph as an edge between non-consecutive nodes, allowing for optimization and reduction of accumulated drift.
Figure 7.6: A loop closure edge (solid purple) over-constrains the back-end pose graph. After optimization, the estimated node frame states (orange) will more closely resemble truth (green). The tear drops represent graph vertices (node frames) while lines represent graph edges. While not shown here for clarity, each estimated and loop-closure edge also stores a covariance matrix representing the uncertainty in that transform. Over-constraining the graph in this manner enables the optimization routine to adjust the graph edges to minimize the total uncertainty in the pose graph (Section 7.4.4).
7.4.3 Intermittent GPS Integration

With optimization, odometry and loop closures can be integrated to maintain a globally-consistent, local map; the first node frame is considered to be the origin of a global coordinate system and the subsequent node frames are defined with respect to it. This type of map is useful for navigation but does not allow for high level path planning techniques such as navigating to desired GPS coordinates. To position the globally-consistent, local map on the earth’s coordinate frame, GPS measurements are incorporated into the back end.

A virtual zero is added to the graph to enable translation and rotation of the local map on the global plane [92]. The virtual zero can be thought of as the origin of the Earth’s global coordinate frame. A virtual constraint is added as an edge between the virtual zero vertex and any other vertex on the graph. The virtual constraint has infinite covariance, meaning that there is no certainty about the relative transformation encoded by the edge. Adding a virtual zero and virtual constraint to the graph allows the map to translate freely within the global coordinate system.

Figure 7.7 depicts the process of adding a GPS measurement to the pose graph. First, when a measurement is received, an odometry edge and vertex is added to the graph representing the location where the measurement was received. This edge comes from the front end’s current estimated state relative to the latest node frame. When the odometry has been established, an additional edge is created linking the virtual zero vertex and the newly created vertex. As with odometry and loop closures edges, the GPS edge encodes the relative transformation between the two vertices with its associated covariance. With GPS edges in place, the graph can be optimized to yield a globally consistent map that aligns with a known global coordinate system.
Figure 7.7: A virtual zero vertex is added to the pose graph and represents the origin of a global coordinate system. Edges between the virtual zero vertex and node frame vertices allow the map to be translated and oriented to its proper global location.

7.4.4 Map Optimization

Because of erroneous odometry measurements, graph optimization is critical to maintaining a globally consistent map. A key advantage of using a pose graph to represent a map is that it can easily be formulated as a least-squares optimization problem where poses and edges become free variables and constraints respectively. Least-squares optimization attempts to find the arrangement of poses that most likely results from a given set of odometry, loop closures, and GPS measurements. Many efficient graph optimization algorithms exist, some of which are popular open-source projects [133].

Figure 7.8 shows a pose-graph map before and after optimization. The data was gathered using the relative front end implementation described in Section 7.3.2 and the map was optimized with the popular g2o library [42]. A multirotor aircraft was flown approximately 125 meters through a series of hallways, forming a loop. About 25 meters of overlap exist in the path. It is clear that prior to optimization, errors accumulated and the global pose
estimate drifted from truth. Optimization mitigates these errors and returns a globally consistent map that matches the flight path.

Intuitively, pose graph optimization can be thought of a mass spring system settling in a state of least energy. In this analogy, graph vertices are masses and edges are springs. A spring's stiffness corresponds to its edge's covariance, becoming increasingly stiff as the covariance approaches zero. In a thought experiment, one can visualize that when a graph constructed of only odometry edges is optimized, the output is simply the original graph because there are no additional loop closure or GPS springs to deform the system. When loop closures are added to the graph, additional stiff springs are added between non-consecutive masses and the system settles into its new lowest energy state, resulting in a graph that more closely represents the true vehicle trajectory. Finally, in this example the virtual zero and virtual constraint can be thought of as an infinitely long string tying the mass-spring system to the origin of the global coordinate system, as shown in Figure 7.9. This establishes a reference to the global coordinate system and allows the mass system to translate about the origin. GPS edges are then added as springs between the virtual zero and corresponding masses, pushing the map into its appropriate absolute position.

One drawback of least-squares optimization is that it is inherently sensitive to outliers. This means that false positive loop closures and erroneous GPS measurements have catas-
Figure 7.9: Illustration of mass spring optimization example. Springs represent edges in the pose graph, with the stiffness of the spring being analogous to the uncertainty associated with the edge (a stiffer spring corresponds to lower uncertainty). A virtual zero is established as the origin of the global coordinate frame and is tied to the mass spring system. GPS edges are added as springs between the virtual zero and the poses where a measurement is received. Map optimization is analogous to releasing the over-constrained mass spring system and allowing it to settle into its lowest energy state.

trophic effects on graph optimization. To compensate for these effects, dynamic covariance scaling (DCS) [83], a variation of least-squares that is robust to outliers, can be used. DCS can be thought of as giving no certainty to edges that are deemed as outliers. By using DCS one is able to produce globally consistent maps even in the presence of erroneous loop closures and GPS measurements. Other methods exist for making graph optimization robust to outliers [84, 94, 134, 135]. A comparison of several of these methods is given in [82].

7.4.5 High-Level Path Planning

The purpose of the high-level path planner is to transform global information, such as waypoints, into the current node frame. As the relative estimator transitions from one node frame to the next, the high-level path planner passes up-to-date relative goals to the low-level path planner. These relative goals are the only way that the global back end influences the front-end control, effectively isolating the UAV from the effects of jumps in the global state estimate due to optimization or global measurements. The high-level path planner can
be used for autonomous waypoint following, exploration and mapping, target tracking, or landing.

7.5 Conclusion

The integration and widespread use of unmanned air systems for many applications depends heavily on the ability of these aircraft to operate in a safe and reliable manner, often in the presence of degraded GPS signals. A relative SLAM-based approach has been presented as a viable solution for robust navigation in GPS-degraded environments. By decoupling the real-time local estimation and control from the global position estimation, the vehicle is able perform essential tasks such as stabilization and obstacle avoidance without being dependent on consistent and accurate global estimates. A pose-graph based global back end allows techniques such as place recognition, loop closures, GPS integration, and map optimization to opportunistically improve global estimates. The separation between the relative front end and global back end also eliminates control issues that can arise when global estimates change as new information is incorporated. An example implementation has been demonstrated for a multirotor aircraft; however, the presented relative navigation framework is not implementation specific and could be adapted to other vehicles, sensors, and mission profiles.
Chapter 8

Conclusions and Future Work

Global drift is inevitable when MAVs fly in GPS-denied environments because the global position and heading states are unobservable when only relative measurements are available. While most navigation frameworks estimate the global state directly despite this unobservability, relative navigation maintains local observability by estimating the vehicle states with respect to a local frame. Estimating and controlling with respect to a local frame produces more consistent global position estimates and avoids many of the consistency and stability issues common to existing global approaches. Relative navigation also provides a framework for robustly incorporating intermittent global information. Moving from a global to a relative estimation approach is a fairly small adjustment conceptually and in terms of implementation, but yields significant advantages that can benefit systems that currently use a global estimation approach.

Given the many, well-established MAV navigation systems currently being developed, there is considerable momentum to continue with global estimation techniques. Many groups have developed successful methods to work around the various issues caused by the loss of observability, and have demonstrated impressive results. That said, the relative navigation framework offers a compelling alternative paradigm for autonomous MAV navigation in GPS-degraded environments. The remainder of this section presents limitations of relative navigation and describes promising avenues of future work that may further motivate the general adoption of the relative navigation paradigm:
Limitations of relative navigation.

1. Requires some level of a back end to reconstruct the global state. This limitation is not particularly onerous as many existing approaches already do something similar.

2. Requires additional care to ensure the estimator, path planner, and controller are all working with respect to the same frame.

3. Incorporates GPS indirectly, which may not be necessary or ideal in many scenarios when GPS is available and reliable.

4. The practical advantages of relative navigation become less apparent in low drift situations such as during short flights, in known or small environments, or when high quality sensors are available.

Tightly-coupled fixed-wing. One of the most important aspects of the relative navigation architecture is that it does not make any assumptions about a particular platform, sensor suite, environment, or use case. Many existing systems could be modified to fit within the relative navigation framework, and thereby benefit from its theoretical and practical advantages. As the relative navigation framework is used on more platforms, with other estimation, path planning, and control algorithms, and for more applications, the merits of the approach may become more pronounced. Future work includes demonstrating relative navigation on fixed-wing vehicles using a monocular camera by leveraging a tightly-coupled relative front end. Individual features would be tracked with respect to body, while the pose of the vehicle is tracked with respect to the node frame.

Multi-agent. So far the relative navigation framework has been demonstrated with a single agent creating a single map. However, because the flight-critical processes are safely handled by the relative front end, the relative navigation framework facilitates a cooperative
global back end. Many agents can contribute information to a common global back end, which in turn can direct the efforts of the swarm. With relative navigation, if an agent’s connection to this common back end is lost, flight stability is not compromised. The relative navigation architecture appears to be a scalable method for distributed control.

**Autonomous missions.** Future work also includes evaluating relative navigation with more sophisticated high-level planning, including autonomous exploration, inspection, and cooperative missions. For infrastructure inspection, the high-level path planner can use a low-resolution map provided by the back end to dynamically plan a flight path that ensures thorough coverage of the scene. Another important research area is ensuring stability when transitioning between controllers using global states and relative states.

**GPS-integration** GPS-INS navigation is actually a subset of relative navigation, where the GPS origin defines a node frame that does not require resetting. As such, future work includes leveraging GPS within the front end to allow the MAV to transition between controlling with respect to relative or global states depending on the quality of measurements.

**Edge optimization** Pose graph optimization is a well developed field of research that is leveraged in the relative navigation back end. Conventionally the non-linear problem is solved using gradient descent techniques, where the system is recursively linearized about the current best estimate. Current approaches parameterize the optimization about the current global pose estimates. In GPS-denied environments, these estimates can be arbitrarily far from truth due to drift. In these cases, the poor linearization can cause current approaches to settle to a local minimum. Future work includes developing a method for parameterizing the optimization about the current edge estimates rather than the current (potentially drifted)
pose estimates. This should provide for a better linearization and reduce the likelihood that the optimization diverges.

**Robocentric consistency**  Chapter 3 raises some interesting questions about the estimation performance of various state parameterizations. The keyframe-robocentric approach discussed in the paper was derived from the original robocentric paper [14] and showed reasonably good estimation performance. Many current robocentric approaches such as [26] do not use the additional delta-state described in [14]. However, this alternate formulation results in poor consistency performance. Future work includes exploring why these approaches break down. It would also be interesting to extend the simulation presented in Chapter 3 to understand how a landmark based estimator compares to a keyframe-based estimator.
References


