Robust Visual-Inertial Navigation and Control of Fixed-Wing and Multirotor Aircraft

Jerel Bendt Nielsen
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/7584

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Robust Visual-Inertial Navigation and Control
of Fixed-Wing and Multirotor Aircraft

Jerel Bendt Nielsen

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

Randal W. Beard, Chair
Timothy W. McLain
Michael A. Goodrich
Brian A. Mazzeo
Dah Jye Lee

Department of Electrical and Computer Engineering
Brigham Young University

Copyright © 2019 Jerel Bendt Nielsen
All Rights Reserved
ABSTRACT

Robust Visual-Inertial Navigation and Control of Fixed-Wing and Multirotor Aircraft

Jerel Bendt Nielsen
Department of Electrical and Computer Engineering, BYU
Doctor of Philosophy

With the increased performance and reduced cost of cameras, the robotics community has taken great interest in estimation and control algorithms that fuse camera data with other sensor data. In response to this interest, this dissertation investigates the algorithms needed for robust guidance, navigation, and control of fixed-wing and multirotor aircraft applied to target estimation and circumnavigation.

This work begins with the development of a method to estimate target position relative to static landmarks, deriving and using a state-of-the-art EKF that estimates static landmarks in its state. Following this estimator, improvements are made to a nonlinear observer solving part of the SLAM problem. These improvements include a moving origin process to keep the coordinate origin within the camera field of view and a sliding window iteration algorithm to drastically improve convergence speed of the observer. Next, observers to directly estimate relative target position are created with a circumnavigation guidance law for a multirotor aircraft.

Taking a look at fixed-wing aircraft, a state-dependent LQR controller with inputs based on vector fields is developed, in addition to an EKF derived from error state and Lie group theory to estimate aircraft state and inertial wind velocity. The robustness of this controller/estimator combination is demonstrated through Monte Carlo simulations. Next, the accuracy, robustness, and consistency of a state-of-the-art EKF are improved for multirotors by augmenting the filter with a drag coefficient, partial updates, and keyframe resets. Monte Carlo simulations demonstrate the improved accuracy and consistency of the augmented filter. Lastly, a visual-inertial EKF using image coordinates is derived, as well as an offline calibration tool to estimate the transforms needed for accurate, visual-inertial estimation algorithms. The imaged-based EKF and calibrator are also shown to be robust under various conditions through numerical simulation.

Keywords: nonlinear systems, nonlinear observer, nonlinear controller, observability, graph optimization, state estimation, Kalman filter, LQR control, vector fields, target tracking, optical flow, fixed-wing control, multirotor control
ACKNOWLEDGMENTS

Working through the past several years to complete this dissertation has been a sometimes frustrating but highly rewarding journey. I am first and foremost grateful for the consistent guidance of my Father in Heaven, who provided me with needed inspiration on numerous occasions throughout this work. I am also grateful for the support of my parents, siblings, and especially my wife throughout this process. I thank my graduate committee and especially my advisor, Randy Beard, for continued support, ideas, and correction. The many discussions and joint efforts on various projects with my fellow members of the MAGICC lab have also been invaluable to my happiness and intellectual development.
# Table of Contents

List of Tables ................................................................. ix

List of Figures ............................................................... x

Chapter 1  Introduction .................................................... 1
  1.1 Multiple Target Tracking ........................................... 1
  1.2 Relative Target Estimation ....................................... 2
  1.3 Aircraft Guidance and Control ................................. 3
  1.4 Visual-Inertial Estimation ....................................... 4
  1.5 Problem Statement ................................................ 5

Chapter 2  Contributions .................................................. 6
  2.1 Ground Target Localization .................................... 6
  2.2 Relative Nonlinear Observer SLAM ............................. 7
  2.3 Relative Target Tracking and Aircraft Guidance ............ 8
  2.4 Robust Fixed-Wing Estimation and Control ................. 8
  2.5 Visual-Inertial Estimation in GPS-Denied Environments .... 9
  2.6 Additional Improvements to Visual-Inertial Estimation in GPS-Denied Environments 10
  2.7 Summary ........................................................... 11

Chapter 3  Relative Target Estimation Using a Cascade of Extended Kalman Filters . 12
  3.1 Introduction ....................................................... 12
  3.2 Landmark Estimation ............................................. 14
# Chapter 3  Target Estimation Methods

3.2.1 Propagation ........................................... 16  
3.2.2 Update ............................................. 17  
3.2.3 Measurement Model .................................. 18  
3.3 Target Estimation Methods ................................. 19  
3.4 Simulation Results ....................................... 21  
3.5 Conclusions ............................................. 26  
3.6 Appendix ................................................ 26  
   3.6.1 Nomenclature ..................................... 26  
   3.6.2 Conventions ...................................... 28  
   3.6.3 Common Identities and Key Derivatives .......... 32  
   3.6.4 Time Derivative of Feature Distance .......... 32  
   3.6.5 Time Derivative of a Bearing Vector .......... 33  
   3.6.6 Bearing Vector Tangent Space Mapping Identities ................. 34  
   3.6.7 Landmark Kinematics Relative to Camera .... 36  
   3.6.8 Camera Kinematics ................................ 37  
   3.6.9 Jacobians of State Kinematics ................. 39  

# Chapter 4  Nonlinear Observer SLAM and Ground Target Tracking  

4.1 Introduction ............................................ 41  
4.2 Nomenclature .......................................... 44  
4.3 Nonlinear Observer Slam ................................ 46  
   4.3.1 Observer Overview ................................ 46  
   4.3.2 Origin Shifting .................................... 48  
   4.3.3 Sliding Window Observer Iteration ........... 49  
4.4 Simulation Results ..................................... 51  
4.5 Conclusions ............................................ 57

# Chapter 5  Relative Moving Target Tracking and Circumnavigation  

5.1 Introduction ............................................ 59
7.1 Introduction .................................................. 117
7.2 Notation ....................................................... 119
    7.2.1 Quaternions ............................................. 120
    7.2.2 ⊞ and ⊟ Operators ....................................... 121
    7.2.3 Feature Bearing Parameterization .................. 122
7.3 Derivation ................................................... 124
    7.3.1 State Definition and Kinematics .................... 125
    7.3.2 Camera Measurement Model ......................... 126
    7.3.3 Accelerometer Measurement Model ................. 127
    7.3.4 Keyframe Reset ....................................... 127
    7.3.5 Partial Update ........................................ 128
7.4 Results ...................................................... 130
7.5 Conclusions ............................................... 133

Chapter 8  A Visual-Inertial Extended Kalman Filter Using Image Coordinates . . . 137
8.1 Introduction ................................................ 137
8.2 Mathematical Preliminaries ............................... 139
    8.2.1 Quaternions ............................................. 140
    8.2.2 Rigid Transforms ....................................... 142
    8.2.3 ⊞ and ⊟ Operators ....................................... 144
8.3 Image-Based EKF ........................................... 146
    8.3.1 Optical Flow ........................................... 146
    8.3.2 EKF Propagation ....................................... 149
    8.3.3 Measurement Models ................................... 153
    8.3.4 EKF Update and Keyframe Reset .................... 156
8.4 Calibration ................................................. 159
    8.4.1 Stage 1: Pose and Motion Capture Calibration .... 161
    8.4.2 Stage 2 - Camera Extrinsicis Calibration .......... 163

vii
8.4.3 Stage 3 - Drag Constant and Body Frame Calibration . . . . . . . . . . . . 167

8.5 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169

8.5.1 Calibration Performance . . . . . . . . . . . . . . . . . . . . . . . . . . . 170

8.5.2 EKF Performance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171

8.6 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177

8.7 Appendix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

8.7.1 2-DOF Quaternions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

8.7.2 Error State Kinematics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

8.7.3 EKF Propagation Jacobians . . . . . . . . . . . . . . . . . . . . . . . . . . 191

8.7.4 IMU Preintegration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193

Chapter 9 Conclusions and Future Work . . . . . . . . . . . . . . . . . . . . . . . . . 199

9.1 Image-Frame and Fixed-Frame Tracking Analysis and Comparison . . . . . . . . 199

9.2 Globally Stable SLAM in Real-Time . . . . . . . . . . . . . . . . . . . . . . . . . 200

9.3 Guidance for Clustered MTT and Maximized Observability . . . . . . . . . . . . 201

9.4 System Identification of Fixed-Wing Aircraft Using Local Optimization . . . . . 201

9.5 Robust Initialization of Visual-Inertial Systems . . . . . . . . . . . . . . . . . . . 202

References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 209
List of Tables

8.1  List of states and calibration parameters. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160

9.1  Advantages of fixed-frame and imaged-based tracking. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 200
## List of Figures

3.1 Ground target tracking data pipeline. ............................................. 15
3.2 Plot of a single landmark estimate versus truth over time. ............... 22
3.3 Target depth estimate versus truth over time using landmark estimator directly. . 23
3.4 Target NED estimate versus truth over time using landmark estimator directly. . 24
3.5 NED position estimates of two targets over time using average bearing method. . 25
3.6 NED position estimates of two targets over time using planar approximation method. 27

4.1 Ground target tracking data pipeline. ............................................. 45
4.2 Body position estimates with origin shifting. ................................. 53
4.3 Body velocity estimates with origin shifting. .................................. 54
4.4 Landmark position estimates with origin shifting. ........................... 55
4.5 Components of velocity bias estimation over time. ........................... 56
4.6 Moving target position estimates with origin shifting. .................... 57

5.1 Relative target position estimation over a one minute flight using the bearing-only observer defined in (5.2). ............................................. 74
5.2 Relative target position and velocity estimation over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9). .............. 75
5.3 Relative target radius and altitude over a one minute flight using the bearing-only observer defined in (5.2). ............................................. 76
5.4 Relative target radius and altitude estimation over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9). .............. 77
5.5 Commanded versus true multirotor velocity over a one minute flight using the bearing-only observer defined in (5.2). ................................. 78
5.6 Commanded versus true multirotor velocity over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9). ......................... 79

6.1 Altitude versus time of all Monte Carlo runs for LQR parameter sweep. Nominal path is shown by the dashed red line. ................................. 112
6.2 Top-down view of all Monte Carlo runs for LQR parameter sweep. Nominal path is shown by the dashed red line. ................................. 113
6.3 Absolute value of EKF error in position, velocity, roll, pitch, and heading for all Monte Carlo runs. ................................................................. 114
6.4 Absolute value of EKF error in accelerometer bias, rate gyro bias, and wind velocity for all Monte Carlo runs. .................................................. 115

7.1 A sample trajectory from the Monte Carlo simulation experiment. The multirotor flies a random path and observes landmarks on the ground. ................. 118
7.2 Illustration of feature bearing vector geometry including error on the plane orthogonal to the bearing vector. .................................................. 124
7.3 Illustration of how node and keyframes are established. ......................... 128
7.4 Average RMSE of the transform from the most recent keyframe (top) and average NEES (bottom) for each filter over the entire simulation time over 2016 runs. .... 133
7.5 The $\chi^2$ distribution with six degrees of freedom compared against each filter at the final simulation time of 5 minutes using 2016 samples. ................. 134
7.6 Drag term estimates of a single MC iteration with and without the partial update compared against truth. .................................................. 134
7.7 Accelerometer biases of a single MC iteration with and without the partial update compared against truth. ............................................ 135
8.1 Motion capture and camera parameter calibration error results of 100 Monte Carlo simulations with significant initial errors. ......................................................... 171
8.2 Other parameter calibration error results of 100 Monte Carlo simulations with significant initial errors. .............................................................. 172
8.3 EKF error in position, velocity, and attitude over 100 Monte Carlo simulations with significant initial error, using depth measurements. ..................... 174
8.4 EKF error in IMU biases and drag coefficient over 100 Monte Carlo simulations with significant initial error, using depth measurements. ............... 175
8.5 EKF error in IMU biases and inverse $z$-depth estimates over 100 Monte Carlo simulations with small initial error in pose and velocity, without depth measurements. 176
8.6 Average error in error-state propagation over 1000 Monte Carlo runs for pixel-based and quaternion-based landmark direction representations. ....... 177
Chapter 1

Introduction

In recent years, small unmanned aircraft have rapidly gained popularity in military organizations, among hobbyists, and in academia. This has been primarily due to the availability, cost, and computational power, which makes them useful in a wide variety of applications, and new applications are emerging every year. Taking the technology a step further, autonomy has also begun to be more desirable, as it affords fewer people to accomplish large tasks with greater efficiency and precision. However, autonomy is a much larger step than just removing the human from the vehicle because computers and sensors interact with the environment differently than people, and the algorithms required to achieve the desired behavior may be complex, yet still imperfect.

1.1 Multiple Target Tracking

One application of autonomy regarding small unmanned aircraft is target tracking. Autonomously tracking multiple targets has been an active area of research for many years, due to its many applications, including law enforcement [1], air traffic control [2], collision avoidance [3], simultaneous localization and mapping (SLAM) [4], tracking space debris [5], and others. There are often three main difficulties in multiple target tracking (MTT): collecting sensor measurements of the targets, associating these measurements with the correct targets (data association), and filtering clutter from true target measurements. Measurement collection can be computationally expensive when image processing is required, for example, when tracking with video cameras. Data association also becomes more difficult as the number of targets and the amount of clutter in the measurements increases.
Many algorithms attempt to solve the MTT problem, such as multiple hypothesis tracking (MHT) [6], joint probabilistic data association (JPDA) [7], and probabilistic hypothesis density (PHD) [8] algorithms. These suffer from computational complexity, requirements of unknown prior information, poor track continuity, or large variance in target estimates. Recursive RANSAC (R-RANSAC) is a recently introduced MTT algorithm [9–13] that improves upon many of these issues. R-RANSAC extends the traditional random sample consensus (RANSAC) algorithm to recursively estimate multiple dynamic signals in clutter. It stores a set of track hypotheses and identifies the best hypothesis of each target’s track, and given a sliding window of measurements, the algorithm either updates the existing hypotheses with a Kalman update or generates new tracks with the set of measurements using RANSAC. R-RANSAC can run in real time on desktop processors or with GPU image processing, but efforts have been made to improve the real-time viability for aerial vehicles with low computational power to track targets in scenes with a large amount of clutter [14].

1.2 Relative Target Estimation

In contrast, this dissertation focuses on the relative estimation of a single target or groups of targets clustered into a single point of reference. This is where autonomy begins. Using R-RANSAC or any other target tracker provides the target information, and having received that information, the aircraft can accomplish some task. There is potentially one more step before computing a control output to perform a task, and that is relative target estimation.

Some works focus on image-based visual servoing [15, 16], in which case target tracking directly in the image would be used. This dissertation focuses on the relative estimation of targets and accomplishes this task in two different ways. The first method is to estimate the target position relative to nearby landmarks. Simultaneous localization and mapping (SLAM) [17, 18] or even visual-inertial odometry (VIO) [19, 20] algorithms estimate static landmark positions relative to the camera, which are then used to approximate target position. This is discussed in Chapter 3.
and demonstrated again in Chapter 4. The second method for estimating relative target position is to use the relative bearing [21, 22]. Chapter 5 estimates target depth using two different nonlinear observers: one assuming bearing only measurements, another assuming depth information is available.

### 1.3 Aircraft Guidance and Control

Having obtained a 3D estimate of the target, guidance laws can be implemented to make the aircraft do something useful. In the case of target tracking and following, something useful is often to just stay in the area and continue tracking the targets of interest. Fixed-wing aircraft are especially useful for this application, due to lift generated by their wings and their ability to loiter for extended periods of time. This idea motivates the need for a circumnavigation guidance law to guarantee that the aircraft can autonomously circumnavigate a target for any amount of time. Despite the advantages of fixed-wing aircraft, Chapter 5 explores this circumnavigation idea with multirotor aircraft as a first cut, being motivated by [21, 23]. One reason for initially using multicopters is that it would be more straightforward to demonstrate the concept in hardware in a motion capture room than outdoors with a fixed-wing. Another reason is that the guidance law computes a velocity command, and it is the job of the controller specific to the aircraft to follow the commanded velocity.

This dissertation also details controllers specific to multirotor and fixed-wing aircraft. In addition to a circumnavigation guidance law, Chapter 5 also develops a nonlinear controller for multirotor aircraft. The primary advantages of nonlinear control are computational efficiency and stability, which were also the motivations for the controller in Chapter 5. Works like [24] also detail nonlinear control of multicopters, but the advantage of the controller derived in Chapter 5 is that control is defined in the body-level reference frame, allowing the user to define a yaw rate command rather than an absolute heading command.
To enable fixed-wing aircraft to also accomplish the circumnavigation task, a robust controller is developed in Chapter 6. This is a state-dependent LQR controller [25], where the desired LQR state is computed with vector fields [26,27], providing excellent robustness in the face of parameter error and strong winds. This method also reduces linearization error by computing the gain matrix every iteration, while coupling the lateral and longitudinal dynamics of the aircraft, contrary to [26].

1.4 Visual-Inertial Estimation

In addition to the fixed-wing LQR controller, Chapter 6 also provides a complete derivation of an extended Kalman filter (EKF) that simultaneously estimates the aircraft state and inertial wind velocity, similar to [28] but without pseudo measurements. This filter takes advantage of several measurements, including IMU, GPS, barometer, and airspeed. It does not use camera measurements because wind velocity cannot be estimated without depth information and RGB-D cameras generally do not have the range needed for fixed-wing aircraft. While this EKF does not include camera measurements, they could be used with GPS to smooth velocity and attitude estimates, as well as provide relative landmark estimates. The LQR controller in Chapter 6 takes estimates from this EKF and successfully follows line paths.

In contrast to Chapter 6, Chapter 7 shifts focus from inertial with GPS estimation to purely visual-inertial estimation. This chapter takes a state-of-the-art visual-inertial EKF (VI-EKF) developed in [19] and improves the accuracy and consistency of the filter with relative resets [29], partial updates [30], and a linear drag term [31]. Chapter 7 demonstrates significant improvements in accuracy and consistency through Monte Carlo simulations when these three modifications to the EKF are used simultaneously. This chapter also compares performance with the original VI-EKF and with the modifications used individually.

Chapter 8 also focuses on the visual-inertial estimation problem, in addition to a calibration routine to estimate sensor transforms needed for visual-inertial estimation. This chapter derives
a VI-EKF using image coordinates and compares its linearity with the quaternion-based state-of-the-art version [19], showing comparable linearity without the requirement of iterated updates. The new EKF also incorporates relative resets [29] to improve consistency, partial updates [30] for better robustness, and a linear drag term [31] for better accuracy. For accurate estimation, several parameters are needed, such as the IMU to camera time offset, IMU to camera transform, and for drag-enabled estimators IMU to drag rotation and the drag coefficient. The calibration routine derived in Chapter 8 obtains these parameters with sufficient aircraft motion, given motion capture, IMU, and camera measurements. This is simpler than [32] but has the benefit of providing drag related information. Other approaches [33,34] estimate parameters online, but this reduces system stability with the added complexity and is only beneficial when parameters are expected to change.

1.5 Problem Statement

A successful autonomous mission is accomplished by solving three robotics problems: guidance, navigation, and control (GN&C). Guidance tells an agent where to go and what to do in a changing environment. Navigation determines what the environment or world is and where the agent is located within that world. Control gives the necessary inputs for the agent to perform the maneuvers that achieve the desired guidance. Often in the literature, we see partially autonomous missions, where assumptions are made to eliminate pieces of the problem, such as a known environment, no obstacles, or perfect navigation. This dissertation aims to robustly solve the full GN&C problem for fixed-wing and multirotor aircraft estimating and circumnavigating arbitrary targets.
Chapter 2

Contributions

This chapter describes the contributions of this dissertation, which is comprised of six technical papers. Papers 1, 2, 3, and 5 have been accepted for publication in three different conferences, while Papers 4 and 6 are currently being prepared for journal submissions. Following detailed descriptions of the contributions provided by each paper, this chapter concludes with a summary of these contributions.

2.1 Ground Target Localization

Paper 1
Relative Target Estimation Using a Cascade of Extended Kalman Filters
Published in the 30th International Technical Meeting of the Satellite Division of the Institute of Navigation [35]

The primary contribution proposed in Paper 1 is a novel method of estimating ground target positions relative to the camera. This method uses estimates of static landmarks near the target, which are then used to approximate a local ground plane. This is followed by the extension of the target bearing vector to intersect this local ground plane. Using static landmarks near the target to approximate a ground plane relaxes the traditional flat-earth assumption [36] but assumes locally planar ground in the immediate vicinity of the target. This produces good, 3D target measurements in the many ground tracking scenarios, where the aircraft flies at a high altitude or there are not many tall objects near the target. To estimate relative landmark positions with an RGB camera, the camera’s linear and angular velocities must be known or simultaneously estimated as in [19]. Only
a general framework of the visual-inertial EKF is provided in [19], so Paper 1 also contributes a thorough derivation of the EKF. This paper is given in Chapter 3.

2.2 Relative Nonlinear Observer SLAM

Paper 2
Nonlinear Observer SLAM and Ground Target Tracking
Published in the 2018 Annual American Control Conference [37]

One contribution of this paper is a practical implementation of the observer derived in [38]. The observer in [38] assumes that the inertial origin is always in view of the camera, which almost never occurs. Paper 2 shows that this can be overcome by updating the origin to a new inertial position that is within the camera’s field of view. This occurs whenever the current origin is about to leave the field of view and is followed by appropriate shifting of position and landmark estimates into the coordinates of the new origin. Our approach is similar to changing node frames as in [29] and also allows operation in a relative manner, while position from the first origin is estimated simultaneously by simply storing the positions associated with each origin update. Another contribution of this paper is greatly improved observer convergence speed via sliding window iteration. The estimates of the nonlinear observer in [38] do not converge quick enough for real-time implementation. As it is not desirable to fly around the starting point, waiting for estimates to converge over a long period of time, the observer runs backward and forward on windows of saved velocity and bearing measurements to more quickly get to the correct solution. Lastly, this paper also provides a demonstration of ground target tracking via the ground plane approximation method described in [35]. This paper is given in Chapter 4.
2.3 Relative Target Tracking and Aircraft Guidance

Paper 3
Relative Moving Target Tracking and Circumnavigation
Accepted to the 2019 Annual American Control Conference

The contributions of this paper include an relative target observer using bearing-only and range with bearing measurements, a circumnavigation guidance law, and a nonlinear controller for a multicopter derived in the body-level reference frame. This bearing only observer assumes a static target as in [21] but for the 3D case, while the range with bearing observer assumes constant velocity targets. Numerical simulation demonstrates convergence of the two observers and boundedness when the assumptions of zero or constant velocity are broken. The circumnavigation guidance law defines a command velocity roughly orthogonal to the direction of the target that pushes the aircraft to a desired relative radius and altitude. Ultimate boundedness of the radial and altitude errors is proven mathematically, providing confidence in this guidance law. The multicopter nonlinear controller has some similarity to [24] but is developed in the body-level frame, removing any dependence on heading. This is especially useful in flights where no absolute position or heading information is available. This paper is given in Chapter 5.

2.4 Robust Fixed-Wing Estimation and Control

Paper 4
LQR Control of a Fixed-Wing Aircraft Using Vector Fields
To be submitted to IEEE Transactions on Control Systems Technology

This paper contributes a state-dependent LQR controller for a fixed-wing aircraft, where desired flight path and course angles are determined using vector fields. There are several benefits to this formulation. State-dependent means that the LQR linearization is computed every control iteration, significantly reducing linearization error compared against traditional LQR applied to
nonlinear systems, where the gain matrix is computed once at some equilibrium point. Performing LQR on attitude, velocity, and angular rate simultaneously couples the lateral and longitudinal dynamics of the aircraft, theoretically giving more accurate and robust control than decoupled solutions, like [36]. Desired flight path and course angles are based on vector fields, which are then used to obtain the desired LQR state. This procedure allows for a sensible selection of the desired LQR state, while making the LQR controller very robust to wind and error in aircraft parameters. Paper 4 also derives an EKF using IMU, GPS, barometer, and airspeed sensors to estimate the aircraft state and inertial wind velocity. The primary contribution related to the EKF is its derivation using error state and Lie group concepts. This paper is given in Chapter 6.

2.5 Visual-Inertial Estimation in GPS-Denied Environments

Paper 5
Improving the Robustness of Visual-Inertial Extended Kalman Filtering
Accepted to the 2019 International Conference on Robotics and Automation

The significant contributions of this paper are improvements to a state-of-the-art VI-EKF [19] and verification of these improvements through Monte Carlo simulation. This paper augments the VI-EKF specifically for use on a multirotor with a drag coefficient for more accurate prediction and better velocity observability, partial updates to improve robustness in drag coefficient and IMU bias estimation, and keyframe resets for filter consistency. Adding a drag term to improve state prediction is an obvious choice when flying the VI-EKF on a multirotor but the use of partial updates and keyframe resets is less obvious. Linearization error in Kalman filters tends to correlate IMU bias estimation with attitude estimation, which is incorrect and results in larger than necessary updates being applied to IMU bias estimates. The partial update is a simple, yet principled method to manually limit the magnitude of the Kalman update to individual elements of the state, which is especially useful when dealing with nuisance states, such as IMU biases and the drag coefficient. The keyframe reset handles a completely different issue, that is consistency. Attempting to estimate absolute position and heading in a Kalman filter without absolute measurements can
lead to a covariance matrix that does not correctly represent uncertainty in the system, hence the filter becomes inconsistent. This is the case with purely visual-inertial estimation, but keyframe resets force the filter to estimate relative position and heading. Paper 5 demonstrates the theoretical advantages of these three features in a VI-EKF running on a multirotor. This paper is given in Chapter 7.

2.6 Additional Improvements to Visual-Inertial Estimation in GPS-Denied Environments

Paper 6
A Visual-Inertial Extended Kalman Filter Using Image Coordinates
To be submitted to IEEE Transactions on Robotics

The contributions of this paper include a visual-inertial EKF based on image coordinates and inverse $z$-depth, a graph-based calibration scheme to obtain key sensor transforms needed for drag-enabled estimation, and a complete derivation of the new filter and calibration scheme based on error state and Lie groups. Current state-of-the-art in visual-inertial extended Kalman filtering employs a quaternion representation of landmark directions and inverse distance to landmarks [19]. Paper 6 shows that the same task can be accomplished using pixel positions directly in the state with inverse $z$-depth for landmark distance, where the $z$-depth is the $z$ component of the 3D landmark vector in the camera frame. This paper shows that using pixel positions directly in the state results in the same or better linearity than the quaternion version, while simultaneously removing the need for an iterated update. The new filter also implements keyframe resets for better consistency, partial updates for robustness, and a linear drag coefficient for better accuracy. To achieve accurate results with a drag-based dynamic model, Paper 6 develops a calibration routine to compute several key transforms, like IMU to body where drag is applied and IMU to camera. The calibration scheme is an offline, batch optimization, where the aircraft is flown in a motion capture room, during which IMU, motion capture, and camera data are collected. Using this data, the calibrator computes the maximum likelihood estimate of the various sensor transforms given the sensor data. This is also
accomplished in three different stages to achieve better robustness in the face of highly uncertain initial estimates. This paper is given in Chapter 8.

2.7 Summary

In summary, this dissertation improves upon the literature regarding relative target tracking, circumnavigation guidance, and visual-inertial extended Kalman filtering. This work offers the following contributions:

- A ground target tracking framework with guarantees and bounds on target localization and circumnavigation when tracking moving ground targets from an aerial vehicle dependent on a minimal set of relative sensors, such as an IMU and camera.

- Novel control algorithms for fixed-wing and multirotor aircraft that operate independent of absolute position and heading with robustness to aircraft parameter error and external disturbances.

- A principled mixture of two control and estimation paradigms: the well-known error-state formulation and modern on-manifold formulation. The combination of these two paradigms simplifies derivation, implementation, and guarantees on-manifold operation of group elements.
Chapter 3

Relative Target Estimation Using a Cascade of Extended Kalman Filters

3.1 Introduction

Autonomous multiple target tracking (MTT) has been an area of interest for several decades, due to a broad list of applications, such as law enforcement [1], air traffic control [2], collision avoidance [3], simultaneous localization and mapping (SLAM) [4], tracking space debris [5], and others. There are typically three main difficulties in MTT: collecting sensor measurements of the targets, associating these measurements with the correct targets (data association), and filtering clutter or noise from true target measurements. Measurement collection is computationally expensive when image processing is required, for example, when tracking with a video camera. The difficulty of data association grows with the number of targets and the amount of clutter in the measurements.

Many algorithms have been developed to solve the MTT problem, such as the multiple hypothesis tracking (MHT) [6], joint probabilistic data association (JPDA) [7], and probabilistic hypothesis density (PHD) [8] algorithms. These suffer from computational complexity, requirements of prior information, poor track continuity, or large variance in target estimates. Recursive RANSAC (R-RANSAC) is a recently introduced MTT algorithm [9–13] that solves many of these issues. R-RANSAC extends the traditional random sample consensus (RANSAC) algorithm to recursively estimate multiple dynamic signals in clutter. It stores a set of track hypotheses and identifies the best hypothesis of each target’s track, and given a sliding window of measurements,

\[1\text{Originally published in [35]}\]
it either updates the existing hypotheses with a Kalman update or generates new tracks with the set of measurements using RANSAC. This algorithm can run in real time, even with a large number of targets and a significant amount of clutter.

For video-based tracking, the algorithm can operate directly in the undistorted, 2D image frame [12], or it can operate in the fixed, 3D fixed frame [39]. To track in the fixed frame, measurements are first projected from image coordinates into the camera frame via a perspective projection, followed by several rotations and translations to the fixed frame. This process and coordinate systems used in this chapter are described in detail in the appendix. Due to ease of implementation, MTT is typically done in the image frame using a nearly constant velocity, acceleration, or jerk model for motion propagation. Advantages of object tracking in the image frame include noise introduced into the measurement only comes from one sensor (the camera), image noise is typically minimal for modern cameras, and the tracker does not depend on the quality of the camera pose estimator. Advantages of fixed-frame tracking include the ability to use target specific motion propagation models, tracks are readily available for multi-vehicle problems, the tracker can account for geography, and poor homographies do not directly affect tracking. Calculation of the relative ground position of a target from a UAV is commonly done via the flat-earth model [36], which assumes the UAV flies above a flat plane. This will clearly experience large errors in real scenarios where the ground is uneven or the UAV altitude is not accurately known. The methods presented in this chapter are capable of 3D target localization without this flat-earth assumption.

The primary contribution proposed in this chapter is a novel method of estimating 3D target positions relative to the camera. Three methods to overcome the flat-earth assumption are presented and simulation results are discussed. Each method relies on a landmark estimator, which uses concepts discussed in the recently introduced robust visual-inertial odometry (ROVIO) [19] algorithm. The first method is to directly insert bearing vector measurements of targets into the landmark estimator with additional process noise added. The second estimates the target’s distance as a weighted average of the landmark estimator’s converged landmark distances. The third
method approximates a ground plane via least squares, given three or more converged landmark estimates, where the target distance is estimated as the intersection of its bearing vector and the approximate ground plane.

The data pipeline is shown in Figure 3.1. The UAV state estimator is an EKF responsible for estimating the position and velocity of the UAV relative to a fixed, local Euclidean reference frame parameterized in north-east-down (NED) coordinates. This estimator also allows us to compute the camera linear and angular velocities, which are needed inputs to the Landmark Estimator. The landmark estimator filters out the moving image measurements and target estimator filters out the static image measurements. As the landmark estimator’s estimates of static landmarks converge, the Target Estimator produces moving target measurements in the 3D camera frame. These can then be transformed into the fixed frame, where they are then processed by a multiple target tracker (R-RANSAC) to produce track estimates also in the fixed frame. The main focus of this chapter is the target estimator block, however, the landmark estimator is also outlined and details of its derivation is contained in the appendix.

3.2 Landmark Estimation

Landmarks are objects that are static relative to the fixed frame and are detected by a camera as features or corners. These features are direct measurements of landmark direction and can be used for tightly coupled visual-inertial odometry [19]. In this chapter, we assume that the UAV has a state estimator separate from a landmark estimator. The landmark estimator is a multiplicative extended Kalman filter (MEKF) that estimates the bearing vector and inverse distance to each landmark, given camera linear/angular velocity and landmark measurements. Camera velocities come from the state estimator and are used to propagate the state forward in time. The landmark bearing measurements update the landmark estimates.
Figure 3.1: The data pipeline for tracking multiple ground targets in the fixed frame. Blue blocks represent functions and the red blocks represent outputs of these functions.
3.2.1 Propagation

The landmark state is comprised of a set of quaternions associated with the bearing vectors of \( N \) landmarks and also the inverse distances to those same landmarks, and is given by

\[
x = \begin{bmatrix} q_0^\top & \rho_0 & \cdots & q_N^\top & \rho_N \end{bmatrix}^\top,
\]

where \( q_i \in SO(3) \) is the shortest rotation from the camera optical axis to the landmark bearing vector represented by a unit quaternion and \( \rho_i \in \mathbb{R} \) is the inverse-depth parameter associated with the \( i^{th} \) landmark. All landmarks are assumed to be static and have the same kinematics defined by

\[
\dot{q} = f_q(\omega, v) = T_\xi^\top \left( \omega_{c/i}^c + \rho_c (\xi_c)^\times v_{c/i}^c \right)
\]

\[
\dot{\rho} = f_\rho(v) = \rho^2 (\xi_c)^\top v_{c/i}^c,
\]

where \( v_{c/i}^c \) and \( \omega_{c/i}^c \) are the linear and angular velocities of the camera with respect to the fixed frame, \( \xi_c \in S^2 \) is the unit vector pointing at the landmark, defined in the camera frame, and \( T_\xi \) is a projection discussed in Section 3.6.

We use a quaternion as the underlying representation for the bearing to each landmark, thus, the state has \( 5N \) dimensions, while the kinematics of the state has \( 3N \) dimensions. This difference in dimensions is because elements of \( S^2 \subset \mathbb{R}^3 \) are minimally represented with two parameters (typically azimuth and elevation) and in order to keep the covariance minimal, changes to the state must be performed in a minimal manner.

The vector and quaternion components of the state, that is the inverse-depth and quaternion components, can be modified on each one’s particular manifold [40]. Euler integration of the kinematics is used to propagate the state forward in time. Propagation of the unit quaternions representing landmark unit vectors [41] and inverse depth parameters for a small time step \( \Delta t \) is
given by

\[
\dot{q}(t + \Delta t) = \dot{q}(t) \boxplus \hat{T}_{\zeta} \left( f_q(\omega, v) \Delta t \right)
\]

\[
\dot{\rho}(t + \Delta t) = \dot{\rho}(t) + f_\rho(v) \Delta t,
\]

where the \( \boxplus/\boxminus \) operators are discussed in Section 3.6.

The change in state covariance with respect to time is given by

\[
\dot{P} = FP + PF^\top + GQ_uG^\top + Q_x,
\]

where the Jacobians \( F \) and \( G \) are derived in the appendix. The state covariance is then also propagated forward with Euler integration by

\[
P(t + \Delta t) = P(t) + \left( FP + PF^\top + GQ_uG^\top + Q_x \right) \Delta t.
\]

### 3.2.2 Update

The measurement residual error for a general measurement with measurement model \( h \) is computed by

\[
r = z \boxminus h(\hat{x}),
\]

where \( z \) is a sensor measurement.

The measurement uncertainty and Kalman gain are computed by

\[
S = HPH^\top + R_z
\]

\[
K = PH^\top S^{-1}.
\]
The update is usually given by
\[ x^+ = x^- + Kr, \]
but the quaternion components cannot be correctly updated with addition. Separating \( Kr \) into a vector state \( \Delta \rho \) associated with the inverse depth components and an attitude state \( \Delta q \) associated with the bearing quaternions, we perform the update by
\[
\hat{q}^+ = \hat{q}^- \boxplus T_\varsigma \Delta q \\
\hat{\rho}^+ = \hat{\rho}^- + \Delta \rho.
\]
Lastly, the covariance is updated by
\[ P^+ = (I - KH) P^- . \]

### 3.2.3 Measurement Model

The landmark bearing vectors are measured directly in the camera frame, so the measurement model is given by
\[ h(\hat{x}) = q \boxplus \eta, \]
where \( \eta \) is Gaussian noise and the measurement Jacobian is then given by
\[
H = \frac{\partial h(\hat{x})}{\partial \mathbf{x}} \\
= \begin{bmatrix} I & 0 \end{bmatrix}.
\]
3.3 Target Estimation Methods

The first method for estimating a moving target’s 3D position relative to the UAV is to use the static landmark estimator directly. The problem with this method is that the kinematics of the estimator are derived from static landmark assumptions. Therefore, this works for slow moving targets to some degree by increasing the process noise parameters of the estimator, but as shown in Section 3.4, it is far from ideal.

The second method requires at least one landmark’s inverse distance state to converge. Assuming that this and other landmarks are near the target, we can take a weighted average of the landmarks’ distances to get the target’s estimated distance. The weights come from the inverse error in the target’s measured bearing vector against the landmarks’ estimated bearing vectors. This means that landmark bearing vectors closer to the target’s have more influence in the calculation of the target’s distance. Suppose we have a target’s bearing vector $\zeta_t$ and several landmark vectors $\zeta_0 \cdots \zeta_N$. The target’s inverse distance is estimated as

$$\hat{\rho}_t = \frac{\sum_{i=0}^{N} w_i \rho_i}{\sum_{i=0}^{N} w_i}, \quad w_i = \frac{1}{\|\zeta_t - \zeta_i\|}.$$ 

In the third target estimation method, we fit a ground plane to three or more landmarks whose inverse distance estimates have converged below some threshold and then find the distance required to scale the target’s bearing vector to intersect this approximated ground plane. It’s important to note that this is done in the camera reference frame. The equation of a plane is given by

$$\mathbf{n}^\top (\mathbf{r} - \mathbf{r}_0) = 0,$$

where $\mathbf{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^\top$ is the vector normal to the plane, $\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ is an arbitrary point on the plane, and $\mathbf{r}_0 = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^\top \delta$ is the offset vector to the center of the plane. Evaluating
this dot product and rearranging variables yields

\[
\frac{n_x}{n_z}x + \frac{n_y}{n_z}y + z = \frac{n_x}{n_z}x_0 + \frac{n_y}{n_z}y_0 + z_0,
\]

and if we let \( a = \frac{n_x}{n_z}, b = \frac{n_y}{n_z}, \) and \( c = \frac{n_x}{n_z}x_0 + \frac{n_y}{n_z}y_0 + z_0, \) we then have

\[
a x + b y + z = c.
\]

Given \( N \geq 3 \) points, a plane can be approximated by a least squares solution of the form

\[
A x = b,
\]

where

\[
A = \begin{bmatrix}
x_1 & y_1 & -1 \\
x_2 & y_2 & -1 \\
\vdots & \vdots & \vdots \\
x_N & y_N & -1
\end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -z_1 \\ -z_2 \\ \vdots \\ -z_N \end{bmatrix},
\]

and where the solution is \( \mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{b}. \) With the ground plane defined and assuming that the target bearing vector is not parallel to this plane, we can find the distance \( d(\rho_t) \) which scales the target bearing vector \( \zeta_t = [\zeta_x \ \zeta_y \ \zeta_z]^\top \) to intersect this ground plane. Inserting \( \mathbf{r} = d(\rho_t) \zeta_t \) into the general equation of a plane yields

\[
\mathbf{n}^\top (d(\rho_t) \zeta_t - \mathbf{r}_0) = 0,
\]

and performing the dot product and some algebra gives

\[
d(\rho_t) \left( \frac{n_x}{n_z} \zeta_x + \frac{n_y}{n_z} \zeta_y + \zeta_z \right) = \frac{n_x}{n_z}x_0 + \frac{n_y}{n_z}y_0 + z_0.
\]
Once again, letting \( a = \frac{n_x}{\bar{n}_x} \), \( b = \frac{n_y}{\bar{n}_y} \), and \( c = \frac{n_z}{\bar{n}_z} x_0 + \frac{n_y}{\bar{n}_y} y_0 + z_0 \), and solving for \( d(\rho) \) leaves us with

\[
d(\rho) = \frac{c}{a \xi_x + b \xi_y + \xi_z}.
\]

### 3.4 Simulation Results

Simulations were used to demonstrate each of the proposed methods for relative 3D target position estimation. A quadcopter flies to user-specified points in a 50 by 50 meter square, 50 meters above the ground. Landmarks are placed with a uniformly random distribution inside of this square area with a maximum vertical deviation from the flat ground of \( \pm 1 \) meter and targets are simply points moving in figure eight patterns at various frequencies. Quadcopter kinematics have become commonplace in the literature, so they are not detailed here but are used in simulation to generate UAV position and velocities. The landmark estimator receives linear and angular velocities of the quadcopter from the UAV State Estimator as shown in Figure 3.1. Since the focus of this chapter is not the UAV state estimator, we assume that it is perfect but then corrupt the measurements of landmarks and targets with Gaussian noise.

Because the bearing to each landmark and target is measured by the camera, it is considered known. Distance or depth, however, is not known and is recursively estimated by the landmark estimator, as shown in Figure 3.2. This shows a single landmark estimate over time as an example but is typical of most landmark estimates, where the distance estimates converge in about a second with very little uncertainty.

Results of the first target estimation method are shown in Figures 3.3 and 3.4. In Figure 3.3, we see that the uncertainty only shrinks to a certain point and the distance error is as much as 20 meters at times. The error in distance translates directly to the estimated north, east, and down components of the target shown in Figure 3.4, where we get errors of ten meters or more on each
Figure 3.2: An example of the convergence of a stationary landmarks over time under good observability conditions.
axis. This is expected because the kinematic model is derived for a stationary landmark. However, this method of target estimation may work for very slow moving targets with a sufficiently high camera frame rate.

Figure 3.3: An example of the convergence of a moving target tracked directly in the landmark estimator over time. The legend is the same as Figure 3.2.

The second target estimation method was used in Figure 3.5. We see that using the weighted average of static landmark errors to estimate the 3D target position works better than the first method. The estimates of both targets have error as high as five meters but roughly exhibit the correct trends. The estimate shows up as zero for the initial quarter of a second because it waits for at least one landmark estimate to converge.
Figure 3.4: Tracking positions of a target in the fixed frame over time using the filter directly. North, east, and down components are shown.
Figure 3.5: Tracking positions of two targets in the fixed frame over time using the average bearing method. North, east, and down components are shown.
The third method of target estimation, using ground plane fitting, is clearly the best, as shown in Figure 3.6. We see that the error is around a meter or less on all axes for both targets being tracked. Notice that the north and east components are estimated almost exactly. This is because the quadrotor is high above the targets, so these components are more affected by the bearing vector estimates of landmarks, which are easily estimated. The down component is subject to the distance estimates of landmarks, which is less precise than the bearings. Thus, the fitted ground plane tends to bounce up and down, causing the down estimate to wander.

3.5 Conclusions

We have developed a method for estimating 3D ground target positions from a UAV equipped with a monocular camera. It is clear that fitting a plane to estimates of landmarks on the ground is the best of the discussed methods. It assumes that the UAV state estimator is working properly and that most of the estimated landmarks are on the ground. Future work includes adding a constant-velocity model to the landmark estimator state to track moving targets directly with better accuracy, incorporating RANSAC in the ground plane fitting to eliminate landmarks such as trees or other tall objects, and running this algorithm in hardware on aerial platforms. Accuracy in a real tracking scenario would also be improved by selecting a region of interest around each target and creating a planar approximation based on landmark estimates nearby each target.

3.6 Appendix

This appendix contains useful derivations related to the landmark estimator, some of which can also be found in [41] using a different notation.

3.6.1 Nomenclature

\( R^b_a \) Rotation from reference frame \( a \) to \( b \)

\( \hat{a} \) Estimate of true variable \( a \)
Figure 3.6: Tracking positions of two targets in the fixed frame over time using the fitted plane method. North, east, and down components are shown.
\( \dot{a} \) Measurement of variable \( a \)

\( \dot{a} \) Time derivative variable \( a \)

\( \ddot{a} \) Error of variable \( a \), i.e., \( \ddot{a} = a - \dot{a} \)

**Superscript**

\( i \) Expressed in the fixed or inertial coordinate frame

\( b \) Expressed in the vehicle body coordinate frame

\( g \) Expressed in the gimbal coordinate frame

\( cb \) Expressed in the camera body coordinate frame

\( c \) Expressed in the camera coordinate frame

\( l \) Expressed in the landmark coordinate frame

\( p \) Expressed in the image coordinate frame

\( \top \) Matrix transpose

**Subscript**

\( a/b \) Velocity or angular rate of frame \( a \) with respect to frame \( b \)

\( ab \) Vector from \( a \) to \( b \)

### 3.6.2 Conventions

This section defines key mathematical operations used throughout the chapter. Quaternions take the form

\[
\mathbf{q} = q_w + q_x \mathbf{i} + q_y \mathbf{j} + q_z \mathbf{k} = \begin{bmatrix} q_w & \mathbf{\hat{q}}^\top \end{bmatrix}^\top,
\]

where \( \mathbf{\hat{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top \) and the standard Hamiltonian convention is used, which is defined by

\[
\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1.
\]
Quaternion multiplication is defined by

\[ p \otimes q = \begin{bmatrix} p_w & -\bar{p}^\top \\ \bar{p} & p_w I + \bar{p}^\times \end{bmatrix} \begin{bmatrix} q_w \\ q \end{bmatrix} = \begin{bmatrix} q_w & -\bar{q}^\top \\ \bar{q} & q_w I - \bar{q}^\times \end{bmatrix} \begin{bmatrix} p_w \\ \bar{p} \end{bmatrix}, \]

and the operator \( (\cdot)^\times \) is the skew-symmetric operator defined by

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

A 3 × 3 rotation matrix associated with \( q^b_a \) may be defined as

\[ R^b_a = R \left( q^b_a \right) = \left( 2q_w^2 - 1 \right) I - 2q_w q^\times + 2\bar{q}q^\top. \]

Landmarks in this chapter move with respect to the camera, similar to traditional, inertial kinematics, and therefore, the following definitions are used

\[ q \triangleq \hat{q} \otimes \bar{q} \]
\[ \bar{q} = \hat{q}^{-1} \otimes q \]
\[ R(q) = R(\bar{q}) R(\hat{q}). \]
We use the same representation of bearing vectors to landmarks as [41]. The bearing vector and its tangent space operator are given by

$$\zeta = R(q)e_z \in S^2 \subset \mathbb{R}^3$$

$$T_\zeta = R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \in \mathbb{R}^{3 \times 2},$$

where $e_{x/y/z} \in \mathbb{R}^3$ are basis vectors of an arbitrary, orthonormal coordinate system and the unit quaternion $q$ is defined by the shortest rotation from the camera optical axis to the bearing vector $\zeta$. Note that we now have two different quaternions for attitude and for a bearing vector, where the attitude quaternion has a superscript and subscript for the coordinate systems and the bearing vector quaternion has no coordinate system superscript or subscript. The bearing vector is over-parameterized because it is a three element, unit length vector, therefore only two parameters (such as azimuth and elevation) are needed to fully describe it. This leads to a singularity in the covariance associated with $\zeta$, so we will remove one element by mapping to the space orthogonal to $\zeta$. The matrix $T_\zeta$ spans this tangent space, where $T_\zeta : \mathbb{R}^2 \to \mathbb{R}^3$ and $T_\zeta^\top : \mathbb{R}^3 \to \mathbb{R}^2$. In other words, $T_\zeta^\top$ maps to the 2-dimensional tangent space of $\zeta$ spanned by $e_x$ and $e_y$.

The landmark estimator uses boxplus and boxminus operators [40], which were applied to unit vector kinematics in [41]. In this chapter, they are defined slightly differently than [41] by

$$\boxplus : SO(3) \times \mathbb{R}^2 \to SO(3),$$

$$q \boxplus \tilde{q} \mapsto q \otimes \exp(T_\zeta \tilde{q})$$

$$\boxtimes : SO(3) \times SO(3) \to \mathbb{R}^2$$

$$p \boxtimes q \mapsto T_\zeta^\top \theta(q, p),$$
where

$$
\theta(q, p) = \cos^{-1} \left( \frac{\xi_p^\top \xi_q}{\| \xi_p \times \xi_q \|} \right) \frac{\xi_p \times \xi_q}{\| \xi_p \times \xi_q \|}.
$$

Given an angular update $\delta \in \mathbb{R}^3$, the exponential map to a unit quaternion is given by

$$
\exp(\delta) = \begin{bmatrix} q_w \\ \vec{q} \end{bmatrix} = \begin{bmatrix} \cos\left( \frac{\|\delta\|}{2} \right) \\ \sin\left( \frac{\|\delta\|}{2} \right) \frac{\delta}{\|\delta\|} \end{bmatrix},
$$

and when $\|\delta\| \approx 0$,

$$
\exp(\delta) \approx \begin{bmatrix} 1 \\ \frac{\delta}{2} \end{bmatrix}.
$$

The corresponding logarithm is given by

$$
\log(q) = 2\arctan2 \left( \|\vec{q}\|, q_w \right) \frac{\vec{q}}{\|\vec{q}\|},
$$

and when $\|\vec{q}\| \approx 0$,

$$
\log(q) \approx \text{sign}(q_w) \vec{q}.
$$

Coordinate reference frames are similar to those defined in [36]. There is a fixed, inertial coordinate system. The vehicle frame is centered on the vehicle body with its axes aligned with the inertial frame. The body frame is the vehicle frame rotated so that its axes align with the body. The gimbal frame is the body frame translated to the gimbal and rotated to align with the gimbal. The camera body frame is the gimbal frame translated and rotated to align with the camera. Typically, camera coordinates have the $z$-axis aligned with the camera’s optical axis and $x$-axis to the right. Therefore, the rotation from the camera body frame to the camera frame is a simple reordering of the axes. Finally, from the camera frame, we can perform a perspective projection.
into the image frame via the intrinsic camera matrix, assuming scene depth is known. We can also rotate from the camera frame to the landmark frame, which aligns the optical axis with a landmark bearing vector via shortest axis-angle rotation.

### 3.6.3 Common Identities and Key Derivatives

Here are some common identities that prove useful in these derivations

\[
(Rv)^\times = Rv^\times R^T
\]

\[
\dot{R}(q_b^i) = R(q_b^i) (\omega_{b/i}^b)^\times
\]

\[
\dot{R}(q_i^b) = - (\omega_{b/i}^b)^\times R(q_i^b).
\]

The following derivatives are taken from [41],

\[
\frac{\partial}{\partial t} q_i^b = - \omega_{i/b}^b
\]

\[
\frac{\partial}{\partial q} (R(q) r) = - (R(q) r)^\times
\]

\[
\frac{\partial}{\partial q} \xi_c^c = (\xi_c^c)^\times T_\xi
\]

\[
\frac{\partial}{\partial q} (T_\xi^T r) = - T_\xi^T (r)^\times T_\xi.
\]

### 3.6.4 Time Derivative of Feature Distance

The distance to a feature in the camera frame is given by

\[
d(\rho) = \frac{1}{\bar{\rho}}.
\]
Differentiating with respect to time yields
\[
\frac{d}{dt}(d(\rho)) = \frac{d}{d\rho}(d(\rho)) \frac{d}{dt}(\rho)
\]
\[
= d'(\rho) \dot{\rho},
\]
where
\[
d'(\rho) = \frac{d}{d\rho}(d(\rho))
\]
\[
= \frac{d}{d\rho} \left( \frac{1}{\rho} \right)
\]
\[
= -\frac{1}{\rho^2}.
\]

### 3.6.5 Time Derivative of a Bearing Vector

A unit vectors pointing to a feature in the camera frame is over-parameterized by a quaternion representation of \( SO(3) \) or a vector in \( \mathbb{R}^3 \), since its minimal representation is in \( S^2 \), the unit sphere in \( \mathbb{R}^3 \). Therefore, its derivative should be represented in a minimal manner. As discussed in [41], the change in a unit vector can be described by an axis-angle rotation vector, \( \zeta \in \mathbb{R}^2 \), in the plane orthogonal to the unit vector. Thus, the time derivative of a unit vector may be given by

\[
\frac{\partial}{\partial t}(\zeta^c) = \frac{\partial}{\partial \mathbf{q}} \frac{\partial}{\partial t}(\zeta^c)
\]
\[
= \frac{\partial}{\partial \mathbf{q}} (\zeta^c) \frac{\partial}{\partial t} \mathbf{q}
\]
\[
= (\zeta^c) \times T_{\zeta} \dot{\zeta}^c.
\]

It’s important to note that \( \dot{\zeta}^c \) indicates a rotation from the new unit vector direction to the old, and thus, the above equation creates a vector pointing from the old direction to the new.
3.6.6 Bearing Vector Tangent Space Mapping Identities

Mapping from the unit vector tangent space and immediately back gives

\[ T_\zeta^T T_\zeta = \left( R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \right)^T R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \]

\[ = \begin{bmatrix} e_x^T \\ e_y^T \end{bmatrix} R^T(q) R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \]

\[ = \begin{bmatrix} e_x^T \\ e_y^T \end{bmatrix} \begin{bmatrix} e_x & e_y \end{bmatrix} \]

\[ = \begin{bmatrix} e_x^T e_x & e_x^T e_y \\ e_y^T e_x & e_y^T e_y \end{bmatrix} \]

\[ = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ = I. \]

Reversing the transformation order, we now have

\[ T_\zeta T_\zeta^T = R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \left( R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \right)^T \]

\[ = R(q) \begin{bmatrix} e_x & e_y \end{bmatrix} \begin{bmatrix} e_x^T \\ e_y^T \end{bmatrix} R^T(q) \]

\[ = R(q) \left( e_x e_x^T + e_y e_y^T \right) R^T(q) \]

\[ = R(q) \left( e_x e_x^T + e_y e_y^T + e_z e_z^T - e_z e_z^T \right) R^T(q) \]

\[ = R(q) \left( I - e_z e_z^T \right) R^T(q) \]

\[ = R(q) R^T(q) - R(q) e_z e_z^T R^T(q) \]

\[ = I - R(q) e_z (R(q) e_z)^T \]
= I − ζζ⊤.

This last identity is needed in the derivation of bearing vector kinematics. We have

\[
T_ζ^T (ζc)^× (ζc)^× a = \left( R(q) \left[ \begin{array}{c} e_x \\ e_y \end{array} \right] \right)^T (ζc)^× (ζc)^× a \\
= \left[ \begin{array}{c} e_x^T \\ e_y^T \end{array} \right] R^T(q) (ζc)^× (ζc)^× a \\
= \left[ \begin{array}{c} e_x^T \\ e_y^T \end{array} \right] \left( R^T(q) ζc \right)^× \left( R^T(q) ζc \right)^× R^T(q) a \\
= \left[ \begin{array}{c} e_x^T \\ e_y^T \end{array} \right] (e_z)^× (e_z)^× R^T(q) a \\
= \left[ \begin{array}{c} e_x^T \\ e_y^T \end{array} \right] \left[ \begin{array}{ccc} -e_x & -e_y & 0 \\ -e^T_y & -e^T_x & 0 \end{array} \right] R^T(q) a \\
= \left[ \begin{array}{ccc} -e_x^T e_x & -e_x^T e_y & 0 \\ -e_y^T e_x & -e_y^T e_y & 0 \end{array} \right] R^T(q) a \\
= \left[ \begin{array}{ccc} -1 & 0 & 0 \\ 0 & -1 & 0 \end{array} \right] R^T(q) a \\
= - \left[ \begin{array}{c} e_x^T \\ e_y^T \end{array} \right] R^T(q) a \\
= -T_ζ^T a,
\]

which means that

\[
T_ζ^T (ζc)^× (ζc)^× = -T_ζ^T.
\]
3.6.7 Landmark Kinematics Relative to Camera

We will derive the feature dynamics relative to the camera. Given a bearing vector $\zeta^c$ pointing from the camera to a landmark, the landmark location can be defined in the inertial frame by

$$p^i_{ln} = p^i_{lc} + R(q^i_l) \zeta^c d(\rho).$$

Taking the time derivative of the entire equation yields

$$0 = v^i_{c/i} + \frac{d}{dt} (R(q^i_l)) \zeta^c d(\rho) + R(q^i_l) \frac{d}{dt} (\zeta^c) d(\rho) + R(q^i_l) \zeta^c \frac{d}{dt} (d(\rho))$$

$$= v^i_{c/i} + R(q^i_l) \left( \omega^c_{c/i} \right) \times \zeta^c d(\rho) + R(q^i_l) (\zeta^c) \times T_\zeta \zeta^c d(\rho) + R(q^i_l) \zeta^c d'(\rho) \dot{\rho},$$

and rotating into the camera frame gives

$$0 = v^c_{c/i} + \left( \omega^c_{c/i} \right) \times \zeta^c d(\rho) + (\zeta^c) \times T_\zeta \zeta^c d(\rho) + \zeta^c d'(\rho) \dot{\rho}$$

$$= v^c_{c/i} - (\zeta^c) \times \omega^c_{c/i} d(\rho) + (\zeta^c) \times T_\zeta \zeta^c d(\rho) + \zeta^c d'(\rho) \dot{\rho}. \quad (3.1)$$

In order to isolate $\dot{\zeta}^c$, multiply by $d(\rho)^{-1} T_\zeta^T (\zeta^c) \times$ to get

$$0 = d(\rho)^{-1} T_\zeta^T (\zeta^c) \times v^c_{c/i} - d(\rho)^{-1} T_\zeta^T (\zeta^c) \times (\zeta^c) \times \omega^c_{c/i} d(\rho) +$$

$$d(\rho)^{-1} T_\zeta^T (\zeta^c) \times (\zeta^c) \times T_\zeta \zeta^c d(\rho) + d(\rho)^{-1} T_\zeta^T (\zeta^c) \times \zeta^c d'(\rho) \dot{\rho}$$

$$= d(\rho)^{-1} T_\zeta^T (\zeta^c) \times v^c_{c/i} - T_\zeta^T (\zeta^c) \times (\zeta^c) \times \omega^c_{c/i} + T_\zeta^T (\zeta^c) \times (\zeta^c) \times T_\zeta \zeta^c,$$

and using the identity $T_\zeta^T (\zeta^c) \times (\zeta^c) \times = -T_\zeta^T$ followed by $T_\zeta^T T_\zeta = I$ gives us

$$0 = d(\rho)^{-1} T_\zeta^T (\zeta^c) \times v^c_{c/i} + T_\zeta^T \omega^c_{c/i} - T_\zeta^T T_\zeta \zeta^c$$

$$= d(\rho)^{-1} T_\zeta^T (\zeta^c) \times v^c_{c/i} + T_\zeta^T \omega^c_{c/i} - \zeta^c.$$
Solving for $\dot{\zeta}^c$ yields

$$\dot{\zeta}^c = T_\zeta^T \left( \omega_{c/i}^c + (\zeta^c)^\times \frac{v_{c/i}^c}{d(\rho)} \right)$$

$$= T_\zeta^T \left( \omega_{c/i}^c + \rho (\zeta^c)^\times v_{c/i}^c \right).$$

To isolate $\dot{\rho}$, multiply (3.1) by $d'(\rho)^{-1}(\zeta^c)^\top$ to get

$$0 = d'(\rho)^{-1}(\zeta^c)^\top v_{c/i}^c - d'(\rho)^{-1}(\zeta^c)^\top (\zeta^c)^\times \omega_{c/i}^c d(\rho) +$$

$$d'(\rho)^{-1}(\zeta^c)^\top (\zeta^c)^\times T_\zeta \dot{\zeta}^c d(\rho) + d'(\rho)^{-1}(\zeta^c)^\top \zeta^c d'(\rho) \dot{\rho}$$

$$= d'(\rho)^{-1}(\zeta^c)^\top v_{c/i}^c + \dot{\rho},$$

and solving for $\dot{\rho}$ yields;

$$\dot{\rho} = \frac{-(\zeta^c)^\top v_{c/i}^c}{d'(\rho)}$$

$$= \rho^2 (\zeta^c)^\top v_{c/i}^c.$$

### 3.6.8 Camera Kinematics

The position of the camera in the inertial frame is given by

$$p_{ic}^j = p_{ib}^j + R(q_{ib}^j) p_{bg}^b + R(q_{ib}^j) R(q_{bg}^b) p_{gc}^g,$$

and assuming that the camera frame is centered on and aligned with the gimbal frame, we are left with

$$p_{ic}^j = p_{ib}^j + R(q_{ib}^j) p_{bc}^b.$$
Differentiating gives

\[ \dot{p}_i^c = \dot{p}_{ib} + \dot{R}(q_b^i) \dot{p}_{bc} + R(q_b^i) \dot{p}_{bc}, \]

and assuming a fixed camera, yields

\[ \dot{p}_i^c = \dot{p}_{ib} + \dot{R}(q_b^i) \dot{p}_{bc} \]

\[ \dot{v}_{c/i}^i = \dot{v}_{b/i}^i + R(q_b^i) (\omega_{b/i}^b) \times \dot{p}_{bc}^b. \]

Rotating both sides into the camera frame gives

\[ R(q_{cb}^c) R(q_b^{cb}) R(q_i^c) \dot{v}_{c/i}^i = R(q_{cb}^c) R(q_b^{cb}) R(q_i^c) \dot{v}_{b/i}^b + R(q_{cb}^c) R(q_b^{cb}) R(q_i^c) (\omega_{b/i}^b) \times \dot{p}_{bc}^b \]

\[ \dot{v}_{c/i}^c = R(q_{cb}^c) (\dot{v}_{b/i}^b + \omega_{b/i}^b) \times \dot{p}_{bc}^b \]

\[ \dot{v}_{c/i}^c = R(q_b^c) \left( \dot{v}_{b/i}^b + (\omega_{b/i}^b) \times \dot{p}_{bc}^b \right) \]

\[ \dot{v}_{c/i}^c = R(q_b^c) \left( \dot{v}_{b/i}^b + (\omega_{b/i}^b) \times \dot{p}_{bc}^b \right). \]

The angular velocity of the camera is the combination of body rotation and gimbal rotation given by

\[ \omega_{c/i}^c = R(q_{cb}^c) R(q_{gb}) \omega_{b/i}^b + R(q_{cb}^c) R(q_{gb}) \omega_{gb}^g \]

\[ = R(q_b^c) \omega_{b/i}^b + R(q_g^c) \omega_{gb}^g, \]

and since we are assuming a fixed camera, this becomes

\[ \omega_{c/i}^c = R(q_b^c) \omega_{b/i}^b. \]
3.6.9 Jacobians of State Kinematics

The state and input noise for a single landmark estimate is given by

\[
\mathbf{x} = \begin{bmatrix} \mathbf{q}^\top & \mathbf{\rho} \end{bmatrix}^\top
\]
\[
\mathbf{u} = \begin{bmatrix} \eta_\omega^\top & \eta_a^\top \end{bmatrix}^\top.
\]

The true body-relative state kinematics are given by

\[
\dot{\mathbf{q}} = T_\zeta^\top \left( \omega_{c/i}^c + \rho (\zeta^c)^\times \mathbf{v}_{c/i}^c \right)
\]
\[
\dot{\mathbf{\rho}} = \rho^2 (\zeta^c)^\top \mathbf{v}_{c/i}^c,
\]

where \( \omega_{b/i}^b \) and \( \mathbf{v}_{b/i}^b \) are inputs from the tracking vehicle’s state estimator and

\[
\mathbf{v}_{c/i}^c = R(\mathbf{q}_b^c) \left( \mathbf{v}_{b/i}^b + (\omega_{b/i}^b)^\times \mathbf{p}_{bc}^b \right)
\]
\[
\omega_{c/i}^c = R(\mathbf{q}_b^c) \omega_{b/i}^b.
\]

The covariance of the state kinematics with respect to the state is then given by

\[
\mathbf{F} = \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{\rho}} \\ \frac{\partial \dot{\mathbf{\rho}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{\rho}}}{\partial \mathbf{\rho}} \end{bmatrix},
\]

where

\[
\frac{\partial \mathbf{q}}{\partial \mathbf{q}} = -T_\zeta^\top \left( (\omega_{c/i}^c + \rho (\zeta^c)^\times \mathbf{v}_{c/i}^c)^\times + \rho (\mathbf{v}_{c/i}^c)^\times (\zeta^c)^\times \right) T_\zeta
\]
\[
\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{\rho}} = T_\zeta^\top (\zeta^c)^\times \mathbf{v}_{c/i}^c
\]
\[
\frac{\partial \dot{\rho}}{\partial q} = \rho^2 \left( v_{c/i}^c \right)^\top (\zeta^c)^\times T_\zeta \\
\frac{\partial \dot{\rho}}{\partial \rho} = 2\rho (\zeta^c)^\top v_{c/i}^c.
\]

The covariance of the state dynamics with respect to the input noise is given by

\[
G = \frac{\partial \dot{x}}{\partial u} = \begin{bmatrix} \frac{\partial q}{\partial \eta_\omega} & 0 \\ \frac{\partial \rho}{\partial \eta_\omega} & 0 \end{bmatrix},
\]

where

\[
\frac{\partial q}{\partial \eta_\omega} = T_\zeta^\top \left( -R(q_b^c) + \rho (\zeta^c)^\times R(q_b^c) \left(p_{bc}^b\right)^\times \right)
\]

\[
\frac{\partial \rho}{\partial \eta_\omega} = \rho^2 (\zeta^c)^\top R(q_b^c) \left(p_{bc}^b\right)^\times.
\]
Chapter 4

Nonlinear Observer SLAM and Ground Target Tracking

4.1 Introduction

Autonomously tracking multiple targets has remained an active area of research for many years, due to the many applications, including law enforcement [1], air traffic control [2], collision avoidance [3], simultaneous localization and mapping (SLAM) [4], tracking space debris [5], and others. There are often three main difficulties in multiple target tracking (MTT): collecting sensor measurements of the targets, associating these measurements with the correct targets (data association), and filtering clutter from true target measurements. Measurement collection can be computationally expensive when image processing is required, for example, when tracking with video cameras. Data association also becomes more difficult as the number of targets and the amount of clutter in the measurements increases.

Many algorithms attempt to solve the MTT problem, such as multiple hypothesis tracking (MHT) [6], joint probabilistic data association (JPDA) [7], and probabilistic hypothesis density (PHD) [8] algorithms. These suffer from computational complexity, requirements of unknown prior information, poor track continuity, or large variance in target estimates. Recursive RANSAC (R-RANSAC) is a recently introduced MTT algorithm [9–13] that improves upon many of these issues. R-RANSAC extends the traditional random sample consensus (RANSAC) algorithm to recursively estimate multiple dynamic signals in clutter. It stores a set of track hypotheses and identifies the best hypothesis of each target’s track, and given a sliding window of measurements,

\[\text{Originally published in [37]}\]
the algorithm either updates the existing hypotheses with a Kalman update or generates new tracks with the set of measurements using RANSAC. R-RANSAC can run in real time on desktop processors or with GPU image processing, but efforts have been made to improve real-time viability for aerial vehicles with low computational power, to track targets in scenes with a large amount of clutter [14].

The R-RANSAC algorithm can represent tracks directly in the image plane [12], or it can represent tracks in the inertial reference frame [39]. To track in the inertial frame, measurements are first projected from pixel coordinates into the inertial frame via a perspective projection. However, due to ease of implementation, MTT is typically done in the image frame using nearly constant velocity, acceleration, or jerk models for motion propagation. Object tracking in the image frame has several advantages, including noise introduced into the measurement only comes from one sensor (the camera), image noise is typically minimal for modern cameras, and the tracker does not depend on camera pose estimation. Advantages of inertial tracking include the ability to use target specific motion propagation models, tracks are readily available for multi-vehicle problems, the tracker can account for geography, and poor homography estimates do not directly affect the tracker.

To track targets in the inertial reference frame, inertial measurements must be provided to the target tracker. These inertial measurements can be obtained with a SLAM or some visual odometry (VO) algorithms. Many state-of-the-art SLAM [18] and VO [42] algorithms use nonlinear optimization techniques to estimate camera pose as well as 3D landmark positions. Others, such as [17, 19], take a filter-based approach and may include all desired estimates in, for example, an extended Kalman filter, unscented Kalman filter, or particle filter. These filter-based SLAM algorithms lack stability guarantees, recently however, others have begun to emerge [38, 43] with provable stability under persistent excitation constraints. In [43], the authors derive a linear, time-varying Kalman filter, where a measured velocity is required for input as well as an accurate attitude estimate. One concern with this filter is that its estimates converge but to an unknown inertial
origin location. Additionally, velocity measurements for a multi-copter require an accurate drag model with no wind or a downward-facing camera to run optical flow with well-estimated scene depth. Similarly, the algorithm in [38] requires an accurate attitude estimate but doesn’t require a velocity measurement. This is because it directly integrates accelerometer measurements for position and velocity, while simultaneously estimating a velocity bias.

The primary contributions of this chapter are two modifications to enable a practical implementation of the observer in [38], that is, sliding-window observer iteration and origin shifting. In [38], the observer assumes that the inertial origin is viewed by the camera at all times. This guarantees that position estimation converges to something that is measured, i.e. the landmark. However, it is not practical to assume that a single landmark will always be observed. This is overcome by occasionally updating the origin to a new inertial position that is in the camera’s field of view whenever the previous origin is about to leave the current field of view. Position and landmark estimates are then shifted appropriately. This approach is similar to changing node frames as in [29] and also allows us to operate in a relative manner, while estimating the vector from the first origin simultaneously by simply storing the vectors associated with each origin update. Another issue with the nonlinear observer in [38] is that estimates do not usually converge quick enough for real time implementation. It is not desirable to fly around the starting point to force landmark and position estimates to converge over a ten to twenty second period. To compensate for this, we allow the observer to iterate on a sliding window of saved velocities and bearing measurements. We store the estimated velocity and bearing measurements for a time window, integrate the position estimate backward in time to the beginning of the window, and then re-run the observer over this window, using the saved data. Each time step, data older than the window size is dropped, new data is added, and the observer processes the new time window. While, this iterative procedure only runs for a short time on initialization and after origin updates, it greatly decreases the time to convergence. Finally, to estimate the position of a maneuvering target, we take three or more esti-
mates of stationary landmarks close to the target to approximate a plane and then scale the bearing vector to the moving target so that it intersects this plane [35].

The data pipeline for tracking targets in an inertial frame is shown in Figure 4.1. The nonlinear observer recently introduced in [38] is used to localize the vehicle and landmarks simultaneously in the inertial frame. Given these landmark estimates and the bearing vectors to the targets, we produce inertial measurements of the ground targets, which are then passed in a multiple target tracker (R-RANSAC) to create ground tracks for each of the targets. The pipeline requires four coordinate frames. The inertial frame is a fixed, Euclidean reference frame parameterized in north-east-down coordinates. The body frame defines the translation and attitude of the vehicle body with respect to the inertial frame. The camera body frame is the translation and rotation of the camera with respect to the vehicle body. Finally, the camera frame’s z-axis is aligned with the camera body frame’s x-axis and its x-axis aligned with the camera body frame’s y-axis, where the camera frame’s z-axis aligns with the camera’s optical axis with the y-axis down and x-axis to the right.

### 4.2 Nomenclature

- $R^b_a$: Rotation from reference frame $a$ to $b$
- $\hat{a}$: Estimate of true variable $a$
- $\bar{a}$: Measurement of $a$
- $\dot{a}$: Time derivative $a$

**Superscript**

- $i$: Expressed in the inertial coordinate frame
- $b$: Expressed in the vehicle body coordinate frame
- $cb$: Expressed in the camera body coordinate frame
- $c$: Expressed in the camera coordinate frame

44
Figure 4.1: The data pipeline for multiple target tracking in an inertial frame. Blue blocks represent functions and the red blocks represent outputs of these functions.
Matrix transpose

Subscript

$a/b$ Velocity or angular rate of frame $a$ with respect to frame $b$

$ab$ Vector from $a$ to $b$

4.3 Nonlinear Observer Slam

4.3.1 Observer Overview

There are four different observers derived in [38], but we are interested in the fourth one, which assumes an unknown velocity bias and unknown landmark positions. Let the state be defined by

$$x = \begin{bmatrix} p_{ib}^i & p_{il,1}^i & \cdots & p_{il,N-1}^i \end{bmatrix}$$

for estimating $N$ landmark positions. The inertial kinematics of the vehicle body and $j^{th}$ landmark are given by

$$\dot{p}_{ib}^i = R_{ib}^i v_{b/i}^b,$$

$$\dot{v}_{b/i}^i = R_{ib}^i \ddot{a}_{b/i}^b + g^i,$$

$$\ddot{p}_{it,j} = 0,$$  \hspace{1cm} (4.1)

where $j = 0, 1, \ldots, N-1$, $\ddot{a}_{b/i}^b$ is the acceleration measured by the IMU, $g^i$ is the gravity vector in the inertial frame, and $v_{b/i}^b$ is the velocity of the vehicle in the body frame. The camera provides bearing vector measurements to each landmark given by

$$y_{ij}^c = p_{cl,j}^c / \left\| p_{cl,j}^c \right\|,$$

which are transformed into the inertial frame for use in the observer by

$$p_{bl,j}^i = R_b^i \left( p_{bc}^b + R_{eb}^h R_{ec}^b p_{cl,j}^c \right)$$

$$y_{ij}^c = \frac{p_{bl,j}^i}{\left\| p_{bl,j}^i \right\|}.$$  \hspace{1cm} (4.2)
In (4.2), we note that $p^c_{el,j}$ is not available. Therefore, bearing measurements are simply rotated into the inertial frame by $y_j^i = R^i_b R^b_{cb} R^c_{cb} y_j^c$. This introduces error into the bearing measurements that increases inversely with the distance to the landmarks, however, placing the camera and IMU near the body center, minimizes this error.

Because the multi-copter is constantly in motion, the IMU cannot correctly measure the direction of gravity for attitude estimation, as in [44], and therefore an alternative form of attitude estimation is needed. Velocity-aided attitude observers, e.g. [45], show great promise for the camera-IMU sensor combination but throughout this chapter, it is assumed that attitude is known.

Having dropped superscripts for simplicity, the nonlinear observer derived in [38] is then given by

$$\dot{\hat{x}} = \begin{bmatrix} \hat{v} \\ 0 \end{bmatrix} - k M \Pi \hat{x}$$

$$\hat{v} = R^i_b \tilde{a}^b + g$$

$$\dot{\hat{\beta}} = -M \Pi M \hat{\beta} - M \Pi \hat{x}$$

$$\dot{M} = I - k M \Pi M$$

$$\hat{x}^\beta = \hat{x} + M \hat{\beta},$$

where $\hat{x}$ is the biased state estimate, $\hat{x}^\beta$ is the unbiased state estimate, $\hat{\beta}$ is the estimated velocity bias, $k > 0$ is the observer gain, $M$ is a real valued matrix, and

$$\Pi = \begin{bmatrix}
\sum_{j=0}^{N-1} \pi_{y,j} & -\pi_{y,1} & -\pi_{y,2} & \cdots & -\pi_{y,N-1} \\
-\pi_{y,1} & \pi_{y,1} & 0 & \cdots & 0 \\
-\pi_{y,2} & 0 & \pi_{y,2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\pi_{y,N-1} & 0 & 0 & 0 & \pi_{y,N-1}
\end{bmatrix}$$

$$\pi_{y,j} = I - y_j y_j^\top,$$
We initialize $\hat{\beta}$ with zeros because the velocity error due to acceleration integration is initially small and $M$ as identity, although it could be any symmetric, positive definite matrix.

### 4.3.2 Origin Shifting

The nonlinear observer defines the inertial origin as $p_{il,0} = \mathbf{0}$, which means that the observer state is defined relative to the landmark $p_{il,0}$. Note that the landmark $p_{il,0}$ is not contained in the state because it is the origin and is always zero. The observer requires a bearing measurement to the origin landmark at all times, and therefore we must select a new origin whenever $p_{il,0}$ is about to leave the camera’s field of view. Since the observer is globally stable, we could select any landmark, even a newly acquired one but this would not allow us to easily track change in position from the first origin. Therefore, we select the new origin to be one of the landmark estimates that has already converged and is expected to leave the frame later than other current landmark estimates.

Assume that we have a history of origins $O_0 \ldots O_p$, where $O_0$ is the initial origin and $O_p$ is the origin at the current time step. Let $\mathbf{d}$ represent the vector pointing from $O_0$ to $O_p$. The estimate $\hat{\mathbf{d}}$ is initialized to a zero vector and every time the origin changes, we have $\hat{\mathbf{d}} = \hat{\mathbf{d}} + \hat{\mathbf{p}}_{il,j}$, where we have selected the $j^{th}$ landmark $\hat{\mathbf{p}}_{il,j}$ to be the new origin. Additionally, the position estimate and each landmark estimate has to be shifted to account for the change in origin by

$$\hat{\mathbf{x}}_{new} = \hat{\mathbf{x}}_{old} - \begin{bmatrix} \hat{\mathbf{p}}_{il,j} \\ \vdots \\ \hat{\mathbf{p}}_{il,j} \end{bmatrix}.$$  

It is easy to see that error in $\hat{\mathbf{d}}$ accumulates over time because imperfect estimates $\hat{\mathbf{p}}_{il,j}$ are continually added to it, however, this is true of many SLAM algorithms operating without some global measurement, such as GPS.
4.3.3 Sliding Window Observer Iteration

The nonlinear observer struggles to converge quickly on startup. Increasing the gain helps to a limited extent but causes a larger error in steady state, in addition to $M$ growing rapidly beyond machine precision, due to the large error at startup when the gain is too large. Another possible method of improving convergence is to save all measurements and state data over a time window and guess the initial conditions iteratively, then select the guess that produced the least amount of error over the window. Guess-and-check methods are costly. Therefore, we could improve this by integrating backward in time from the end of the window to the beginning, use this as the new initial guess, re-run the observer over the window using the saved data, and repeat. The problem with this method is that measurement windows with insufficient excitation drive the estimates away from truth, so we improve upon this by allowing the window to slide with time. A sliding window gives us the ability to intelligently estimate the initial state iteratively, while continually using new measurements that have varying levels of excitation for each of the landmarks. Thus, to reduce the convergence time of the observer without introducing instability, we allow the observer to iterate on a sliding window of saved bearing measurements and velocities. This process is outlined in Algorithm 1 for a window of size $S$ time steps.

In line 8 of Algorithm 1, we use Euler integration to propagate the position backward in time to the beginning of the sliding window. We do not backward integrate $M$ and $\hat{\beta}$. This is because $\hat{\beta}$ depends on $M$ and $M$ must be a solution to the Ricatti equation given in (4.3), which means that $M$ must also be positive definite. Backward integration of $M$ is given by $M(t - 1) = M(t) - I\Delta t$, which removes the positive definite guarantee on $M$. Therefore, we simply save the values of $M$ and $\hat{\beta}$ at the beginning of each window and use these initial values for forward integration. Another subtlety of this algorithm is that landmark estimates are not backward integrated. Recall that the observer is operating in the inertial frame, hence static landmark kinematics given in (4.1) do not change the landmark estimates.
Algorithm 1 Sliding Window Observer Iteration

1: for each \( t \) do
2:   Save \( y_0...N-1 \) and \( \hat{v} \) to history window.
3:   if history size > \( S \) then
4:     Trim oldest values of \( y_0...N-1 \) and \( \hat{v} \).
5:   end if
6:   if history size = \( S \) then
7:     for \( j = 0 \) to \( T-1 \) do
8:       for \( m = S-2 \) to 0 do
9:         \( \hat{p}_{ib}^j = \hat{p}_{ib}^j - \hat{v}_m \Delta t \)
10:     end for
11:     \( M = M_{\text{saved}} \)
12:     \( \hat{\beta} = \hat{\beta}_{\text{saved}} \)
13:     for \( n = 0 \) to \( S-2 \) do
14:       \( M = M + (I - kM \Pi_n M) \Delta t \)
15:       \( \hat{\beta} = \hat{\beta} + \left(-M \Pi_n M \hat{\beta} - M \Pi_n \hat{x}\right) \Delta t \)
16:       \( \hat{x} = \hat{x} + \left[\begin{array}{c} \hat{v}_n \\ 0 \end{array}\right] - kM \Pi_n \hat{x} \Delta t \)
17:     end for
18:   end for
19:   \( M_{\text{saved}} = M \)
20:   \( \hat{\beta}_{\text{saved}} = \hat{\beta} \)
21: end if
22: end for

The observer iteration procedure reduces convergence time of the observer but also accelerates observer divergence from truth when observability is low. In practice, observer iteration also
hinders the observer’s ability to track the velocity bias, which causes the velocity error to slowly grow, so iteration should only be used briefly on initialization and after origin changes. In general, observer iteration improves state error reduction, given a time window where observations are persistently exciting and velocity error is small. Because global, exponential stability of the nonlinear observer, assuming persistent excitation, is proven in [38], any set of initial conditions are driven toward the truth by the observer, resulting in less error at the end of a time window than at the beginning, given a large enough window. While the error in velocity is small over a small window, backward integration of the position results in a new initial condition with less error than the original initial condition.

4.4 Simulation Results

To investigate the performance of the nonlinear observer and its modifications in a SLAM setting, we setup a multi-copter simulation, where the multi-copter uses a PID controller to fly a user-defined 3D path. Control inputs are computed based on the multi-copter’s true state, so that we can easily compare results on the same observed state. This work is done in simulation prior to hardware implementation so that observer performance can easily be separated from the quality of data association of camera measurements. Therefore, the supplied measurements are created by computing unit vectors pointing to landmarks in the vehicle body frame. This is less realistic in that there is no simulated camera or restrictions from having a finite field of view, but we can still demonstrate the validity and shortcomings of the proposed methods in an ideal scenario. In the following simulations, the multi-copter takes off near the initial origin and flies a figure-eight path at a nearly constant altitude of ten meters. The simulated accelerometer has added Gaussian noise set to a standard deviation of 0.1 m/s² with a small, constant bias. It is assumed that a separate observer provides accurate attitude estimates, therefore we simply assume it is known. Sliding window iteration has parameters $T = 1$, $S = 20$, and only runs for two seconds after initialization and origin shifts. The landmark of the first origin is located at zero to simplify plotting.
The figures in this section demonstrate the performance of this observer, in addition to running it with sliding window iteration, origin shifting, and using it for target tracking in the inertial frame. Figures 4.2 and 4.3 show the position and velocity estimation performance, respectively, of the observer with and without sliding window iteration enabled. Figure 4.4 is just one of ten inertial landmark estimates over time, shown as an example of landmark convergence. We see that sliding window iteration of the observer greatly reduces the convergence time on startup. Figure 4.5 shows that the observer is able to track the general trend of the velocity bias over time with and without sliding window iteration.

Results for origin shifting are also shown in Figures 4.2-4.4, where we shift the inertial origin to the location of the second landmark \( \mathbf{p}_{il,1} \) every five seconds and replace \( \mathbf{p}_{il,1} \) with a random, new landmark until the thirty second mark. We do this because there is no field of view constraint in the simulation, but we still need to show how the observer behaves when the origin changes. This results in small, sudden changes in the position and velocity estimates due to the poor initial estimation of the new landmark. In Figure 4.4, we see that the new landmark’s north and east estimates converge quickly, while coverage of the down estimate is delayed by a second or two. This happens because the observer projects the estimate onto the measurement direction, decreasing the landmark vector’s length significantly, which is then increased as more observations arrive. An improved method of initializing landmark estimates would likely reduce this initial error, and due to the consensus style of estimation, diminish the jumps in the other estimates. Additionally, origin drift is apparent in Figure 4.4, but the drift of the iterated observer’s estimates is reduced because of the fast convergence of landmark estimates.

Finally, Figure 4.6 shows the inertial position estimation of a maneuvering target over time using the planar approximation method presented in [35], where it is assumed that target bearing vector is measured. This requires the landmark estimates to be defined with respect to the vehicle body, which is accomplished for the \( j^{th} \) landmark by \( \mathbf{p}_{bl,j} = \mathbf{p}_{il,j} - \mathbf{p}_{ib,j} \). This remains in the inertial frame and creates vectors pointing from the vehicle body to the landmarks. Now, we see
Figure 4.2: Position components of the observer with origin shifting every five seconds compared against truth. Solid blue is ground truth, dash-dot red is the estimate without sliding window iteration, and dashed green is the estimate with sliding window iteration.
Figure 4.3: Velocity components of the observer with origin shifting every five seconds compared against truth. Solid blue is ground truth, dash-dot red is the estimate without sliding window iteration, and dashed green is the estimate with sliding window iteration.
Figure 4.4: Landmark position components of the observer with origin shifting every five seconds compared against truth. Solid blue is ground truth, dash-dot red is the estimate without sliding window iteration, and dashed green is the estimate with sliding window iteration.
Figure 4.5: Velocity bias components of the observer with and without sliding window iteration enabled on startup compared against truth. Solid blue is ground truth, dash-dot red is the estimate without sliding window iteration, and dashed green is the estimate with sliding window iteration.
from Figure 4.6 that the tracker works about as well as the position estimation, being affected by
the changing origin and like the other estimates, quickly converging toward the truth after each
origin shift.

![Target Estimation Components (m)](image)

**Figure 4.6:** Target position components with origin shifting every five seconds compared against truth using
sliding window iteration. Solid blue is ground truth and dashed red is the estimate.

### 4.5 Conclusions

We presented two methods necessary for a practical implementation of the observer from [38],
namely, origin shifting and sliding window observer iteration. Origin shifting suffered from large
errors immediately after the change in origin, but due to the global stability of the observer and
sliding window observer iteration, this error quickly diminishes. Smoothing the estimation during
origin changes is to be carefully examined in future work. One possibility for smoothing is to
use a second observer for the initial estimation new landmarks only, so that newly added landmarks do not negatively affect current position and landmark estimates. Sliding window iteration dramatically improves observer convergence speed during sequences of persistent excitation and is necessary for hardware implementation of this observer. We require further analysis of this method to better specify optimal conditions of its operation, such as how long it should run after newly established origins. Poor initial estimates combined with suboptimal observability of landmarks can cause observer iteration to degrade filter performance, therefore, detection of these scenarios should also be investigated. Lastly, we showed that this observer could be used to track a maneuvering ground target with accuracy comparable to its own position estimation.
Chapter 5

Relative Moving Target Tracking and Circumnavigation

5.1 Nomenclature

\[ R_a^b \] Rotation from reference frame \( a \) to \( b \)

\( \dot{a} \) Estimate of true variable \( a \)

\( \ddot{a} \) Measurement of \( a \)

\( \dot{a} \) Time derivative \( a \)

Superscript

\( i \) Expressed in the inertial coordinate frame

\( l \) Expressed in the aircraft’s body-level coordinate frame

\( b \) Expressed in the aircraft’s body coordinate frame

\( \top \) Matrix transpose

Subscript

\( a/b \) The state of frame \( a \) with respect to frame \( b \) (e.g. position or velocity)

---

1Accepted to the 2019 American Control Conference
The skew symmetric operator is defined by $\cdot \wedge$, such that $\mathbf{a} \wedge \mathbf{b} = \mathbf{a} \times \mathbf{b}$. We also make use the basis vectors

\[
\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top, \\
\mathbf{e}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top, \\
\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top,
\]

and denote other unit vectors by $\mathbf{e}_*$. 

### 5.2 Introduction

Target tracking and surveillance from an unmanned air vehicle (UAV) has been an area of interest in the research community for many years, primarily aimed at military applications, due to the cost of sensors and aerial platforms. In recent years however, the development and proliferation of increasingly smaller sensors, such as inertial measurement units (IMU) and video cameras has caused the emergence of small unmanned aircraft systems (sUAS). This phenomenon has greatly multiplied the possibilities for aerial target tracking and brought forth many works related to the circumnavigation of targets [21–23, 46–49].

This proliferation of low-cost sensors has also led to a significant effort to fuse IMU measurements with camera measurements. The IMU provides high rate measurements of linear acceleration and angular rate, while the camera provides direction or full-vector measurements to landmarks, depending on the type of camera. These types of measurements are ideal for continuous-discrete Kalman filtering because the mechanization of IMU measurements provides high-rate prediction, while the camera provides low-rate corrections. The fusion of these measurements has been thoroughly demonstrated through Kalman filtering, optimization, and nonlinear techniques in many recent visual-inertial odometry (VIO) and simultaneous localization and mapping (SLAM) works [18, 19, 42, 50–52].
Many of the existing circumnavigation algorithms assume a known inertial state [21, 46]. These will not perform as well when the following agent is only equipped with a camera and IMU for state estimation because the global position and heading are not observable [53]. Others have developed algorithms for GPS-denied target tracking and following [47, 54] with successful hardware demonstrations. Some even follow targets using image-based visual servoing (IBVS) techniques [55]. However, these assume a known object size or appearance and do not address the observability required for the state estimator to estimate IMU biases, enabling accurate attitude estimation.

This chapter develops observers and controllers for relative estimation and circumnavigation of a moving ground target, assuming that the aircraft is equipped with only a camera and IMU for state estimation. While cameras typically provide only bearing information, some modern cameras are capable of providing depth information. When a depth measurement is not available, we can create pseudo-depth measurements by approximating target position relative to static landmark position estimates [35]. Therefore, we develop two observers in this chapter, where the first uses bearing-only information and the second takes advantage of full vector information. Following the target observers, we also use nonlinear system theory to define a commanded velocity vector to drive the aircraft to a circumnavigating path about the target. Finally, a nonlinear controller for a multirotor aircraft is derived in the body-level reference frame to drive the multirotor to a commanded velocity. While we do not develop multirotor state estimator in this chapter, we note that observability of the IMU biases is also guaranteed because of the persistent excitation of landmark bearing measurements [56] resulting from the circumnavigating motion.

We begin in Section 5.3 by developing the target observers using bearing-only and range-with-bearing measurements. This is followed by Section 5.4.1, where we derive the commanded body-level velocity needed to bring an arbitrary aircraft to a desired radius and altitude relative to the target. Next in Section 5.4.2, we design a controller specific to a multirotor aircraft to drive the multirotor’s velocity to a commanded body-level velocity. Lastly, Section 5.5 demonstrates
the effectiveness of these observers and controllers under various target motions and discusses the results.

5.3 Target Estimation

5.3.1 Bearing-Only Measurements

The position of the target relative to the UAS and its time derivative are given by

\[
\begin{align*}
p_{t/b}^i &= R_i^b (p_{i/i}^i - p_{b/i}^i) \\
\dot{p}_{t/b}^b &= \dot{R}_i^b (p_{i/i}^i - p_{b/i}^i) + R_i^b \left( \dot{p}_{i/i}^i - \dot{p}_{b/i}^i \right).
\end{align*}
\]

Assuming a stationary target \( \dot{p}_{i/i}^i = 0 \) and inserting \( \dot{R}_i^b = -\left( \mathbf{\omega}_{b/i}^b \right)^\wedge R_i^b \), this becomes

\[
\dot{p}_{t/b}^b = -\mathbf{\omega}_{b/i}^b \times p_{t/b}^i - v_{b/i}^b,
\]

where \( \mathbf{\omega}_{b/i}^b \) and \( v_{b/i}^b \) are the angular and linear rates of the UAS. Here, we have assumed that the aircraft’s visual-inertial estimator is working well, such that its observable states are known, and we have also assumed a stationary target. However, the error will remain bounded for slowly moving targets as shown in [21].

**Proposition 1** Given the dynamics in (5.1) with measurements \( e_t = \frac{p_{t/b}^i}{\left\| p_{t/b}^i \right\|}, \mathbf{\omega}_{b/i}^b, \) and \( v_{b/i}^b \), the bearing-only observer for relative target position is given by

\[
\dot{\hat{p}}_{t/b}^b = -\mathbf{\omega}_{b/i}^b \times \hat{p}_{t/b}^b - v_{b/i}^b - k_1 \left( I - e_t e_t^\top \right) \hat{p}_{t/b}^b,
\]

where a positive gain \( k_1 \) yields globally, asymptotically convergent estimation error \( \tilde{p}_{t/b}^b = \hat{p}_{t/b}^b - p_{t/b}^i \), if \( v_{b/i}^b \) is not aligned with \( e_t \).
**Proof** The estimation error evolves in time according to

\[
\dot{\tilde{p}}_{t/b} = \left( -\left( \omega_{b/i} \right)^\wedge - k_1 \left( I - e_t e_t^\top \right) \right) \tilde{p}_{t/b}.
\]

Define the Lyapunov function candidate \( \mathcal{L} = \frac{1}{2} \left( \tilde{p}_{t/b} \right)^\top \tilde{p}_{t/b} \) and differentiate to obtain

\[
\dot{\mathcal{L}} = -k_1 \left\| \tilde{p}_{t/b} \right\|^2 + k_1 \left( e_t^\top \tilde{p}_{t/b} \right)^2,
\]

which is negative semi-definite by the Cauchy-Schwarz inequality

\[
\left\langle e_t, \tilde{p}_{t/b} \right\rangle \leq \left\| e_t \right\| \left\| \tilde{p}_{t/b} \right\| = \left\| \tilde{p}_{t/b} \right\|.
\]

Let the set \( S = \left\{ (e_t, \tilde{p}_{t/b}) \mid \dot{\mathcal{L}}(e_t, \tilde{p}_{t/b}) = 0 \right\} \) contain only trajectories, where \( \tilde{p}_{t/b} = 0 \) or the directions of \( e_t \) and \( \tilde{p}_{t/b} \) are aligned. Suppose that we begin with a trajectory in the set \( S \) and \( \tilde{p}_{t/b} \neq 0 \). When \( e_t \) and \( \tilde{p}_{t/b} \) are aligned, \( e_t \) and \( \tilde{p}_{t/b} \) are also aligned. We then have the relationship

\[
e_t \times \tilde{p}_{t/b} = \frac{d}{dt} \left[ e_t \times \tilde{p}_{t/b} \right] = 0,
\]

and evaluating the time derivative in (5.3) yields

\[
e_t \times \dot{\tilde{p}}_{t/b} + e_t \times \tilde{p}_{t/b} = 0,
\]

where

\[
\dot{e}_t = -\omega_{b/i} \times e_t - \left( I - e_t e_t^\top \right) \frac{v_{b/i}}{\left\| \tilde{p}_{t/b} \right\|}.
\]

Using \( e_t \times \tilde{p}_{t/b} = 0 \) simplifies (5.4) to

\[
v_{b/i} \times \dot{\tilde{p}}_{t/b} = 0.
\]
Therefore, when $v_{b/i}^h$ is not aligned with $e_t$, we have $v_{b/i}^h \times \ddot{p}_{t/b}^b \neq 0$, and the trajectory will leave the set $S$, resulting in $\dot{L} < 0$ and $\ddot{p}_{t/b}^b \rightarrow 0$.

5.3.2 Range-with-Bearing Measurements

The position and velocity of the target relative to the agent are given by

$$p_{t/b}^h = R_t^b \left( p_{t/i}^i - p_{b/i}^i \right)$$

$$v_{t/b}^h = R_t^b \left( v_{t/i}^i - v_{b/i}^i \right). \quad (5.5)$$

Assuming a constant velocity target $\dot{v}_{t/i}^i = 0$ and inserting $\dot{R}_t^b = -\left( \omega_{b/i}^b \right)^\wedge R_t^b$, this becomes

$$p_{t/b}^h = v_{t/b}^h - \omega_{b/i}^b \times p_{t/b}^b \quad (5.6)$$

$$v_{t/b}^h = -\omega_{b/i}^b \times v_{t/b}^h - v_{b/i}^b, \quad (5.7)$$

where $\dot{v}_{b/i}^h$ is the aircraft’s acceleration. This may be written in terms of the accelerometer measurement as

$$\dot{v}_{b/i}^h = \ddot{a}_{b/i}^b + gR_t^b e_3 - \omega_{b/i}^b \times v_{b/i}^h, $$

where $g$ is gravity’s magnitude and $\ddot{a}_{b/i}^h$ is the measured specific acceleration of the IMU.

**Proposition 2** Given the dynamics in (5.6) and (5.7) with measurements $p_{t/b}^b$, $\omega_{b/i}^b$, and $\dot{v}_{b/i}^h$ the observer for relative target position and velocity is given by

$$\dot{\hat{p}}_{t/b}^h = \hat{v}_{t/b}^h - \omega_{b/i}^b \times \hat{p}_{t/b}^h - k_1 \ddot{p}_{t/b}^b \quad (5.8)$$

$$\dot{\hat{v}}_{t/b}^h = -\omega_{b/i}^b \times \hat{v}_{t/b}^h - \dot{v}_{b/i}^h - k_2 \ddot{p}_{t/b}^b, \quad (5.9)$$
where positive gains $k_1$ and $k_2$ yield globally, asymptotically convergent estimation errors $\dot{p}_t^b = \dot{p}_t^b - p_t^b$ and $\dot{v}_t^b = \dot{v}_t^b - v_t^b$.

**Proof**  The estimation errors evolve in time according to

$$
\begin{align*}
\dot{\hat{p}}_t^b &= -\omega_{b/i}^b \times \hat{p}_t^b + \hat{v}_t^b - k_1 \hat{p}_t^b, \\
\dot{\hat{v}}_t^b &= -\omega_{b/i}^b \times \hat{v}_t^b - k_2 \hat{p}_t^b.
\end{align*}
$$

Define the Lyapunov function candidate $L = \frac{1}{2} \left( (\hat{p}_t^b)^\top \hat{p}_t^b + \frac{1}{k_2} (\hat{v}_t^b)^\top \hat{v}_t^b \right)$ and differentiate to obtain

$$
\dot{L} = -k_z \|\hat{p}_t^b\|^2,
$$

which is negative semi-definite.

Let the set $S = \left\{ (\hat{p}_t^b, \hat{v}_t^b) \mid \dot{\hat{p}}_t^b, \hat{v}_t^b = 0 \right\}$ contain only trajectories, where $\hat{p}_t^b = 0$. Suppose that we begin in the set $S$ and $\hat{v}_t^b \neq 0$. Under this condition, the errors evolve according to

$$
\begin{align*}
\dot{\hat{p}}_t^b &= \hat{v}_t^b, \\
\dot{\hat{v}}_t^b &= -\omega_{b/i}^b \times \hat{v}_t^b,
\end{align*}
$$

indicating that this trajectory will leave the set $S$, except when $\hat{p}_t^b = \hat{v}_t^b = 0$. Thus, by the LaSalle Invariance Lemma, $\hat{p}_t^b \to 0$ and $\hat{v}_t^b \to 0$. 

$\blacksquare$
5.4 Circumnavigation

This section makes use of the body-level reference frame which is the body frame with roll and pitch removed. We derive the following velocity and multirotor controllers in this body-level reference frame to remove any dependence on heading.

5.4.1 Velocity Control

In this section, we derive the commanded velocity in the body-level reference frame needed to bring an arbitrary aircraft to a circumnavigating orbit about a target at some constant, desired relative radius \( r_d \) and altitude \( h_d \). The relative radius and altitude in the body-level frame are computed by rotating the body-fixed relative target position into the body-level frame and projecting onto the horizontal plane and vertical axis. These are written in terms of the relative target position by

\[
\begin{align*}
  r &= \left\| \left( I - e_3 e_3^\top \right) R^l_{b\text{p}_{t/b}} \right\| \\
  h &= e_3^\top R^l_{b\text{p}_{t/b}}.
\end{align*}
\]

Differentiating these with respect to time yields

\[
\begin{align*}
  \dot{r} &= e_r^\top \left( v^l_{t/i} - v^l_{b/i} \right) \quad \text{(5.10)} \\
  \dot{h} &= e_3^\top \left( v^l_{t/i} - v^l_{b/i} \right), \quad \text{(5.11)}
\end{align*}
\]

where \( e_r \) is the horizontal target direction in the body-level frame and \( e_r \perp e_3 \). The horizontal, radial target direction may be written in terms of \( e_t \) or \( p^b_{t/b} \) by

\[
\begin{align*}
  e_r &= \frac{(I - e_3 e_3^\top) R^l_{b} e_t}{\left\| (I - e_3 e_3^\top) R^l_{b} e_t \right\|} \quad \text{(5.12)} \\
  &= \frac{(I - e_3 e_3^\top) R^l_{b} p^b_{t/b}}{\left\| (I - e_3 e_3^\top) R^l_{b} p^b_{t/b} \right\|}. \quad \text{(5.13)}
\end{align*}
\]
Proposition 3  Given the dynamics in (5.10) and (5.11) with measurements $e_r, R^l_p$, and (5.12), the commanded body-level velocity for bearing-only target estimation from Proposition 1 is given by

$$v^l_c = k_r \tilde{r} e_r + v_t \frac{e_3 \times e_r}{\|e_3 \times e_r\|} + k_h \tilde{h} e_3,$$

(5.14)

where positive gains $k_r, k_h$ and user-defined tangential velocity $v_t$ yield globally, asymptotically bounded estimation errors $\tilde{r} = r - r_d$ and $\tilde{h} = h - h_d$.

Proof  The relative radial and altitude errors evolve in time according to

$$\dot{\tilde{r}} = e_r^\top \left( v^l_{i/i} - v^l_{b/i} \right)$$

(5.15)

$$\dot{\tilde{h}} = e_3^\top \left( v^l_{i/i} - v^l_{b/i} \right).$$

(5.16)

Selecting $v^l_{b/i} = v^l_c$ from (5.14), these become

$$\dot{\tilde{r}} = -k_r \tilde{r} + e_r^\top v^l_{i/i}$$

$$\dot{\tilde{h}} = -k_h \tilde{h} + e_3^\top v^l_{i/i}.$$

Define the Lyapunov function candidate $L = \frac{1}{2} (\tilde{r}^2 + \tilde{h}^2)$ and differentiate with respect to time to obtain

$$\dot{L} = -k_r \dot{\tilde{r}}^2 - k_h \dot{\tilde{h}}^2 + \left( \tilde{r} e_r^\top + h e_3^\top \right) v^l_{i/i}.$$  

(5.17)

Define $x = \left[ \begin{array}{c} \tilde{r} \\ \tilde{h} \end{array} \right]^\top$ and (5.17) becomes

$$\dot{L} = -x^\top A x + b^\top x,$$
where

\[ A = \begin{bmatrix} k_r & 0 \\ 0 & k_h \end{bmatrix} \]

\[ b = \begin{bmatrix} e_r^\top v_{t/i}^l \\ e_3^\top v_{t/i}^l \end{bmatrix}. \]

With \( \sigma_{\text{min}} = \min(k_r, k_h) \) and \( \sigma_{\text{max}} = \max(k_r, k_h) \), we then have

\[ \sigma_{\text{min}} \| x \|^2 \leq x^\top A x \leq \sigma_{\text{max}} \| x \|^2, \]

which yields the relation

\[ \dot{\mathcal{L}} \leq \sigma_{\text{min}} \| x \| \left( \frac{\| b \|}{\sigma_{\text{min}}} - \| x \| \right). \]

Therefore, \( \dot{\mathcal{L}} < 0 \) if \( \| x \| > \frac{\| b \|}{\sigma_{\text{min}}} \) and the solution is ultimately bounded in finite time. After convergence to the bounded region, we have

\[ \| x \| \leq \frac{\| b \|}{\sigma_{\text{min}}}, \]

which depends on target velocity but can be made arbitrarily small by choosing \( k_r, k_h \) to be arbitrarily large.

**Proposition 4**  Given the dynamics in (5.10) and (5.11) with measurements \( p_{t/b}^b, R_{l/b}^l \), and (5.13), the commanded body-level velocity for range-with-bearing target estimation from Proposition 2 is given by

\[ v_c^l = k_r \tilde{e}_r + v_t \frac{e_3 \times e_r}{\| e_3 \times e_r \|} + k_h \tilde{e}_3 + R_{l/b}^l \dot{v}_{t/i}^b. \]  (5.18)
where positive gains $k_r$, $k_h$ and user-defined tangential velocity $v_t$ yield globally, asymptotically convergent estimation errors $\tilde{r} = r - r_d$ and $\tilde{h} = h - h_d$.

**Proof** Inserting $v_{b/i}^l = v_{c}^l$ from (5.18) into (5.15) and (5.16) gives

$$
\dot{\tilde{r}} = -k_r \tilde{r} + e_r^T R_b^l \left( v_{t/i}^b - \dot{v}_{t/i}^b \right),
$$

$$
\dot{\tilde{h}} = -k_h \tilde{h} + e_3^T R_b^l \left( v_{t/i}^b - \dot{v}_{t/i}^b \right),
$$

and using (5.5), these become

$$
\dot{\tilde{r}} = -k_r \tilde{r} + e_r^T R_b^l \left( -\tilde{v}_{t/b}^b + R_i^b \left( v_{b/i}^i - \dot{v}_{b/i}^i \right) \right),
$$

$$
\dot{\tilde{h}} = -k_h \tilde{h} + e_3^T R_b^l \left( -\tilde{v}_{t/b}^b + R_i^b \left( v_{b/i}^i - \dot{v}_{b/i}^i \right) \right).
$$

Assuming that the multirotor’s state estimator works well, $\dot{v}_{b/i}^i \to v_{b/i}^i$, and we are left with

$$
\dot{\tilde{r}} = -k_r \tilde{r} - e_r^T R_b^l \tilde{v}_{t/b}^b,
$$

$$
\dot{\tilde{h}} = -k_h \tilde{h} - e_3^T R_b^l \tilde{v}_{t/b}^b.
$$

Define the Lyapunov function candidate $\mathcal{L} = \frac{1}{2} \left( \tilde{r}^2 + \tilde{h}^2 \right)$ and differentiate with respect to time to obtain

$$
\dot{\mathcal{L}} = -k_r \tilde{r}^2 - k_h \tilde{h}^2 - \left( \tilde{r} e_r^T + \tilde{h} e_3^T \right) R_b^l \tilde{v}_{t/b}^b.
$$

The proof of Proposition 2 showed that $\tilde{v}_{t/b}^b \to 0$. Therefore, $\dot{\mathcal{L}} < 0$, ensuring that $\tilde{r} \to 0$ and $\tilde{h} \to 0$.

To reduce the likelihood of the target moving out of the camera’s field of view, the desired radius and altitude may be chosen according to the direction of the camera optical axis during level
flight. For a known optical axis angle from vertical $\theta$, we may choose a desired relative altitude and let the desired radius be chosen mathematically to align the level optical axis with the target by

$$r_d = h_d \tan \theta.$$  

5.4.2 Multirotor Velocity Control

Neglecting wind and drag, velocity dynamics of a multirotor aircraft in the body-level frame are given by

$$\dot{v}_{b/i}^l = ge_3 - \frac{T}{m} R_{ib}^l e_3 - \omega_{i/i}^l \times v_{b/i}^l, \quad (5.19)$$

where $g$ gravitational magnitude, $T$ is thrust, $m$ is vehicle mass, and the yaw rate of the body-level frame is

$$\omega_{i/i}^l = e_3 e_3^\top R_{ib}^l \omega_{b/i}^l.$$  

Assuming a known hover throttle signal $s_h$, we approximate thrust with $T \approx mg \frac{s}{s_h}$ and (5.19) becomes

$$\dot{v}_{b/i}^l = g \left( I - \frac{s}{s_h} R_{ib}^l \right) e_3 - \omega_{i/i}^l \times v_{b/i}^l.$$  

Defining the velocity error $\tilde{v} = v_{c}^l - v_{b/i}^l$ and holding the command constant, the time derivative is given by

$$\dot{\tilde{v}} = -g \left( I - \frac{s}{s_h} R_{ib}^l \right) e_3 + \omega_{i/i}^l \times v_{b/i}^l, \quad (5.20)$$

Now, let

$$u = s R_{ib}^l e_3, \quad (5.21)$$
and (5.20) becomes

\[ \dot{\tilde{v}} = \frac{g}{s_h} u - g e_3 + \omega_{l/i}^l \times v_{b/i}^l. \]

This allows us to select \( u \) as a vector input that drives the velocity error to zero. Define the Lyapunov function candidate \( \mathcal{L} = \frac{1}{2} \tilde{v}^\top \tilde{v} \) and differentiate to obtain

\[ \dot{\mathcal{L}} = \tilde{v}^\top \left( \frac{g}{s_h} u - g e_3 + \omega_{l/i}^l \times v_{b/i}^l \right). \]

Choose

\[ u = s_h \left( e_3 - \frac{1}{g} \left( \omega_{l/i}^l \times v_{b/i}^l + K_v \tilde{v} \right) \right), \]

and (5.22) reduces to

\[ \dot{\mathcal{L}} = -\tilde{v}^\top K_v \tilde{v}, \]

which is negative definite for positive definite \( K_v \). Equating (5.21) and (5.23) gives

\[ s R_b^l e_3 = s_h \left( e_3 - \frac{1}{g} \left( \omega_{l/i}^l \times v_{b/i}^l + K_v \tilde{v} \right) \right). \]

Using the current aircraft attitude estimate and solving for the commanded thrust signal yields

\[ s_c = s_h e_3^\top \left( R_b^l \right)^\top \left( e_3 - \frac{1}{g} \left( \omega_{l/i}^l \times v_{b/i}^l + K_v \tilde{v} \right) \right). \]

Now using the commanded thrust signal, we need to solve for the commanded attitude. Substituting \( s = s_c, R_b^l = (R_b^l)_c \) in (5.24) and dividing by \( s_c \), we have

\[ \left( R_b^l \right)_c e_3 = \frac{s_h}{s_c} \left( e_3 - \frac{1}{g} \left( \omega_{l/i}^l \times v_{b/i}^l + K_v \tilde{v} \right) \right). \]
We cannot directly solve for \((R^l_b)_c\), but this equation tells us that the commanded body-down axis in the body-level frame should point in the direction
\[
(e^b_3)_c = \frac{e_3 - \frac{1}{g} \left( \omega^l_{l/j} \times v^l_{b/j} + K_v \hat{v} \right)}{\left\| e_3 - \frac{1}{g} \left( \omega^l_{l/j} \times v^l_{b/j} + K_v \hat{v} \right) \right\|}.
\]

Having the body-down axis pointed in this direction relative to the body-level frame will drive the multirotor to the commanded velocity, regardless of its heading. The commanded body-forward and body-right axes in the body-level frame may then be given by
\[
\begin{align*}
(e^b_2)_c &= \frac{(e^b_2)_c \times e_1}{\left\| (e^b_2)_c \times e_1 \right\|}, \\
(e^b_1)_c &= (e^b_2)_c \times (e^b_3)_c,
\end{align*}
\]
and the commanded attitude is now given by
\[
(R^l_b)_c = \begin{bmatrix}
(e^b_1)_c & (e^b_2)_c & (e^b_3)_c
\end{bmatrix}.
\]

However, many flight controllers require roll and pitch angles along with a commanded yaw rate. Roll and pitch angles are easily computed from \((R^l_b)_c\) because it has the form
\[
(R^l_b)_c = \begin{bmatrix}
\cos \theta & \sin \phi \sin \theta & \cos \phi \sin \theta \\
0 & \cos \phi & -\sin \phi \\
-\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta
\end{bmatrix},
\]
which leads to the commanded roll and pitch angles
\[
\phi_c = \tan^{-1} \left( \frac{e^\top_2 (R^l_b)_c e_3}{e^\top_2 (R^l_b)_c e_2} \right)
\]
\[
\theta_c = \tan^{-1}\left(-\frac{e_3^T (R_b^l)_c e_1}{e_1^T (R_b^l)_c e_1}\right).
\]

Commanded yaw rate can be given by the angular difference about the body z axis between the camera optical axis and the target direction vector. Given the fixed optical axis defined in the multirotor’s body reference frame \(e_{opt}^b\), the commanded body yaw rate is given by

\[
r_c = k_r e_3^T \left( e_{opt}^b \times e_r \right),
\]

where \(k_r\) is a positive gain. This does not drive the angular error in yaw exactly to zero but a high gain can reduce the error close to zero.

5.5 Simulation Results

To demonstrate the performance of the observers and controllers derived in Sections 5.3 and 5.4, we designed a simulation of a multirotor and a nonholonomic ground vehicle. The multirotor is modeled with nonlinear aerodynamic drag and collects measurements of the ground vehicle in the body reference frame each time step. Measurements of the target are also corrupted with random Gaussian noise that is zero mean and has a standard deviation of magnitude \(10^{-1}\). The ground vehicle uses a bicycle steering model, has variable elevation, and follows a cycled list of four waypoints that form the shape of a square about the inertial origin. The ground vehicle also moves with a constant forward velocity of one meter per second, while the multirotor orbits with a velocity saturated at five meters per second and is commanded to face the ground vehicle at all times. We assume that a visual-inertial estimator is properly operating onboard the multirotor, providing the true linear and angular velocities, true IMU biases, as well as the true roll and pitch of the aircraft. The desired relative radius and altitude is set to five meters in all experiments.

Figure 5.1 shows relative target position estimation using bearing-only measurements, while Figure 5.2 shows relative position and velocity estimates using range-with-bearing measurements.
Figure 5.1: Relative target position estimation over a one minute flight using the bearing-only observer defined in (5.2).
Figure 5.2: Relative target position and velocity estimation over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9).
We see that flying at five times the velocity of the ground vehicle, the multirotor is able to correctly estimate the ground vehicles relative position using the range-with-bearing method and converges to a bounded region for the bearing-only method. However, about every 15 seconds, we observe larger deviations from truth, which is due to the sharp turns of the ground vehicle as it moves to the next desired waypoint. This happens to both observers because neither one accounts for turns in their motion models, but because the turns are brief, the observers’ estimates do not significantly deviate from truth.

Figure 5.3: Relative target radius and altitude over a one minute flight using the bearing-only observer defined in (5.2).

Figures 5.3 and 5.4 show each observer’s relative radius and altitude estimates over a one minute flight. The estimates quickly converge to the truth at the beginning of the simulation and experience larger deviations during sharp turns of the ground vehicle because that is where the observers’ estimates experience a more significant deviation from truth. The velocity controller
Figure 5.4: Relative target radius and altitude estimation over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9).
successfully drives the aircraft to the desired relative radius and altitude according to target estimation estimates, but we see that the true relative radius and altitude using the bearing-only observer deviates to a bounded region due to observer error.

![Bearsings-only Commanded and True Velocities](image)

**Figure 5.5:** Commanded versus true multirotor velocity over a one minute flight using the bearing-only observer defined in (5.2).

Finally, Figures 5.5 and 5.6 depict the multirotor’s commanded velocity versus its true velocity for both the bearing-only and range-with-bearing observers. The sensor noise applied to the measurements bleeds through into the commanded velocity most notably along the \(x\) axis, but we see that the true velocity generally lines up with the commanded velocity throughout the flight with the exception of the \(y\) axis. This is primarily due to multirotor drag not being modeled in the derivation of the multirotor velocity controller, as we see a mostly constant offset between commanded velocity and true velocity.
Figure 5.6: Commanded versus true multirotor velocity over a one minute flight using the range-with-bearing observer defined in (5.8) and (5.9).
5.6 Conclusions and Future Work

We have developed nonlinear observers for relative target position and velocity estimation using bearing-only and range-with-bearing measurements. We also designed a controller to bring a multirotor aircraft to some desired body-level frame velocity and computed the velocity command that puts the aircraft into a circumnavigating orbit about the target. We have also shown mathematically and through simulation that a multirotor does indeed approach the desired relative radius and altitude using our controller and observers. This work has better enabled GPS-denied target tracking and circumnavigation from a multirotor aerial platform. Future work includes applications to fixed-wing aircraft, a hardware implementation of the observers and controller, development of a decision process during loss of target tracking, exploration of observers that account for higher order motion, and possibly accounting for aerodynamic drag and wind in the multirotor controller.
Chapter 6

LQR Control of a Fixed-Wing Aircraft Using Vector Fields

6.1 Introduction

The robust control and estimation of fixed-wing aircraft has been an important problem of interest for many decades, beginning with large, manned aircraft and recently for small unmanned air vehicles. Autonomy of these small aircraft is also increasingly growing in demand due to the numerous applications, such as search and rescue [57], disaster relief [58], mapping/surveying [59], and others. The key to automating or simplifying these tasks for users is robust state estimation and control that continues to function as intended, in spite of unknown environment or hardware-related conditions.

The first problem to overcome for any kind of aircraft control is state estimation. Most often, small fixed-wing aircraft are equipped with an IMU, GPS, barometer, and a pitot tube (airspeed sensor) because these sensors can be found in low-cost form and the data output is easily handled. It is relatively straightforward to fuse information from these sensors in a Kalman filter, which is one of the most common schemes used in state estimation. One challenge when using this suite of sensors is the estimation of wind velocity. This is challenging because the pitot tube can only provide one component of airspeed. While varying aircraft attitude makes wind velocity observable using a single pitot tube, equipping the aircraft with vanes attached to encoders to directly measure angle of attack and sideslip angle provides the airspeed vector measurement, but off-the-shelf sensors of this type are not common. Estimation in [28] is accomplished via

---

1To be submitted to IEEE Transactions on Control Systems Technology
Kalman filter, where they also apply pseudo-measurements of steady-state angle of attack and sideslip under the assumption that the aircraft returns to its trim values after disturbances. A complementary filter for attitude is developed in [60] uses an IMU and a pitot tube to estimate the attitude of the aircraft under coordinated turn conditions. An unscented Kalman filter (UKF) is developed in [61] and compared against an extended Kalman filter (EKF) for attitude estimation. In [36], an EKF that estimates all of the states needed for control is derived using Euler angles for attitude and a typical suite of low-cost sensors. Note that to perform any meaningful task when using these attitude-only estimators, additional estimators are needed to make aircraft position, velocity, and wind estimates available to the controller.

Assuming that estimation is functioning properly, control may then be applied to the aircraft. Longitudinal and lateral controllers based on PID and vector fields are designed in [36], primarily due to their simplicity. In [62], a nonlinear guidance law is developed and the authors demonstrate its superior accuracy compared to traditional PID control, as well as theoretical stability. The authors of [26, 27, 63] explore and demonstrate the concept of using vector fields as guidance for the underlying controllers, ensuring stable trajectories under various disturbances like wind gusts. The approach in [63] is generalized to time-varying curves in $n$-dimensional space demonstrate successful paths with ground robots, while [27] focus their efforts on proving global stability with Lyapunov-based methods mathematically and numerically. Experimental results in [26] show successful orbit and line following of a fixed-wing aircraft in the presence of some wind using low-cost sensors for estimation.

This chapter tackles both the estimation and control problems associated with the autonomy of fixed-wing aircraft. Primary contributions are a robust control scheme that couples longitudinal and lateral dynamics, in addition to a modern and efficient full-state estimator verified numerically on a fixed-wing aircraft. Accurate state estimation is accomplished with an EKF, taking advantage of IMU, GPS, barometer, and pitot tube sensors. Attitude is represented with a unit quaternion, preventing the possibility of gimbal lock and improving performance of the estimator, due to the
lower computational load associated with this representation [64]. The filter is also based on error state techniques and Lie group theory. This results in a minimal representation of attitude covariance and some derivations being simplified versions of those found in [29]. While wind is estimated in the EKF using a single pitot tube, we must highlight that the aircraft must continually change direction to observe the full wind vector because the pitot tube only measures a single component of this vector. The controller uses vector field concepts for guidance and achieves the desired objective via a linear quadratic regulator (LQR) operating simultaneously on aircraft velocity, attitude, and angular rate. Fixed-wing aircraft must fly like nonholonomic vehicles to maintain lift by minimizing sideslip, therefore using position error directly in the controller can make it difficult to get stable control outputs. Instead, we use position error in the vector-field calculation. This feeds into the calculation of desired aircraft velocity, attitude and angular rate, which are more directly tied to its control outputs. Typical use of LQR is to linearize about some nominal state, such as trim, and calculate the gain matrix one time prior to flight. As seen in Section 6.3, aircraft dynamics are highly nonlinear, so linearizing about a single trim state does not result in a robust solution. Therefore, we linearize and compute the gain matrix every control iteration, resulting in a state-dependent LQR formulation [25] that handles well the high degree of system nonlinearity in any flight regime.

The chapter is organized as follows. Sections 6.2 discusses the notation and mathematics used throughout the chapter. This is followed by a description of aircraft dynamics in Section 6.3. Section 6.4 derives the vector-field guidance laws and briefly describes the use of these laws in the LQR controller. Next, the EKF used for state estimation is derived in Section 6.5, followed by simulation results in Section 6.6.

6.2 Mathematical Preliminaries

6.2.1 Nomenclature

\( \hat{a} \) Estimate of true variable \( a \)
\( \dot{a} \)  Desired value of \( a \)
\( \bar{a} \)  Measurement of \( a \)
\( \hat{a} \)  Error of \( a \)
\( \dot{a} \)  Time derivative \( a \)

**Superscript**

\( I \)  Expressed in the inertial coordinate frame
\( l \)  Expressed in the line path coordinate frame
\( b \)  Expressed in the aircraft’s body coordinate frame
\( \top \)  Matrix transpose

**Subscript**

\( a/b \)  The state of frame \( a \) with respect to frame \( b \) (e.g. position or velocity)

The skew symmetric operator is defined by \([ \cdot ]_\times\), such that \([a]_\times b = a \times b\). Reversing the order of cross products gives

\[
[a]_\times b = -[b]_\times a. \tag{6.1}
\]

We also denote basis vectors and other unit vectors by \( e_\ast \).

### 6.2.2 Rotations

This chapter employs unit quaternions (Hamilton convention) to represent rotations

\[
\mathbf{q} = \begin{bmatrix}
q_0 \\
\mathbf{q}
\end{bmatrix} \in S^3,
\]
where \( \bar{q} \) is the complex portion of \( q \) and \( q_0 \) is the real part. Quaternion multiplication is defined by

\[
p \otimes q = \begin{bmatrix}
p_0q_0 - \bar{p}^\top \bar{q} \\
q_0\bar{p} + p_0\bar{q} + \bar{p} \times \bar{q}
\end{bmatrix}.
\]

We often use rotation matrices when rotating vectors. The construction of a rotation matrix from a unit quaternion is denoted as \( R(q_b^a) = R_b^a \), where \( R_b^a \) rotates a vector expressed in frame \( a \) to a vector expressed in frame \( b \). The construction is defined by

\[
R(q) = (2q_0^2 - 1)I - 2q_0[\bar{q}]_\times + 2\bar{q}q^\top.
\]

The construction of a unit quaternion from roll \( \phi \), pitch \( \theta \), and yaw \( \psi \) Euler angles is given by

\[
q(\phi, \theta, \psi) = \begin{bmatrix}
\cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \\
\cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\
\cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\
\sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2}
\end{bmatrix}.
\]

We also make use of the \( \boxplus \) and \( \boxminus \) operators [40], which are overloaded for different manifolds. For vectors, these operators consist of typical addition and subtraction of vector elements. For unit quaternions in this chapter, we define the \( \boxplus/\boxminus \) operators as

\[
\boxplus : S^3 \times \mathbb{R}^3 \to S^3,
\]

\[
q \boxplus \delta \mapsto q \otimes \exp(\delta)
\]

\[
\boxminus : S^3 \times S^3 \to \mathbb{R}^3,
\]

\[
q_2 \boxminus q_1 \mapsto \log\left(q_1^{-1} \otimes q_2\right),
\]
where the exponential and logarithmic maps are given by

\[
\exp(\delta) = \begin{bmatrix}
\cos\left(\frac{\|\delta\|}{2}\right) \\
\sin\left(\frac{\|\delta\|}{2}\right) \frac{\delta}{\|\delta\|}
\end{bmatrix}
\]

\[
\log(q) = 2\text{atan2}(\|\tilde{q}\|, q_0) \frac{q}{\|\tilde{q}\|}.
\]

When \(\|\delta\| \approx 0\) or \(\|\tilde{q}\| \approx 0\), these maps are approximated as

\[
\exp(\delta) \approx \begin{bmatrix} 1 \\ \frac{\delta}{2} \end{bmatrix}
\]

(6.6)

\[
\log(q) \approx 2\text{sign}(q_0) \tilde{q}.
\]

Aligning with the theory of manifold operators \(\oplus/\oslash\), which we have just described, we also represent unit quaternion derivatives and error in the Lie Algebra, such that \(\dot{\tilde{q}} \in \mathbb{R}^3\) and \(\tilde{\tilde{q}} \in \mathbb{R}^3\). It is clear from (6.5) that \(\tilde{q} \in \mathbb{R}^3\), but the derivative is less obvious. Derivatives are fundamentally defined by limits, so let’s take a limit to verify that \(\dot{\tilde{q}} \in \mathbb{R}^3\). Let \(q_{t+\Delta t} = q_t \oplus \omega \Delta t\) for a constant angular rate \(\omega \in \mathbb{R}^3\) and a time step \(\Delta t\). Taking a limit with appropriate addition and subtraction operators given in (6.4) and (6.5), we have

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

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

\[
= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left(\log\left(q_t^{-1} \otimes q_t \otimes \exp\left(\omega \Delta t\right)\right)\right)
\]

\[
= \omega.
\]
6.3 Aircraft Dynamics

The aircraft’s position, velocity, attitude, and angular rate evolve in time according to

\[
\dot{p}_{b/I} = (R_{I}^{b})^\top v_{b/I}^{b} \\
\dot{v}_{b/I}^{b} = \frac{1}{m} F^{b} - \omega_{b/I}^{b} \times v_{b/I}^{b} \\
\dot{q}_{I}^{b} = \omega_{b/I}^{b} \\
\dot{\omega}_{b/I}^{b} = J^{-1} \left( \tau^{b} - \omega_{b/I}^{b} \times J \omega_{b/I}^{b} \right),
\]

where \( m \) is the aircraft’s mass, \( J \) is the aircraft’s inertia matrix, and \( F^{b} \) and \( \tau^{b} \) are the force and torque applied to the aircraft body [36].

We assume that the aircraft is equipped with the four control inputs: aileron, elevator, throttle, and rudder. The aircraft receives a throttle signal \( s_{t} \in [0, 1] \) and signals for aileron, elevator, and rudder given by

\[
s_{a} = \frac{\delta_{a}}{\delta_{a_{\text{max}}}} \in [-1, 1] \\
s_{e} = \frac{\delta_{e}}{\delta_{e_{\text{max}}}} \in [-1, 1] \\
s_{r} = \frac{\delta_{r}}{\delta_{r_{\text{max}}}} \in [-1, 1],
\]

where \( \delta_{\ast} \) denotes deflection angle in radians and \( \delta_{\ast_{\text{max}}} \) is the physically defined, maximum angle of deflection.

Vehicle air velocity, air speed, angle of attack, and side slip angle are defined by

\[
v_{a/I}^{b} = v_{b/I}^{b} - R_{I}^{b} v_{w/I}^{I} \\
V_{a} = \left\| v_{a/I}^{b} \right\|,
\]
\[ \alpha = \tan^{-1}\left( \frac{\mathbf{e}_3^b \mathbf{v}_{a/l}^b}{\mathbf{e}_1^b \mathbf{v}_{a/l}^b} \right) \]
\[ \beta = \sin^{-1}\left( \frac{\mathbf{e}_2^b \mathbf{v}_{a/l}^b}{V_a^b} \right) \]

where \( \mathbf{v}_{w/l}^f \) is the wind velocity expressed in the inertial frame.

Nondimensionalized coefficients of lift and drag are defined by

\[ C_L(\alpha) = (1 - \sigma(\alpha)) [C_{L_0} + C_{L_\alpha} \alpha] + \sigma(\alpha) [2 \text{sign}(\alpha) \sin^2 \alpha \cos \alpha] \]
\[ C_D(\alpha) = C_{D_p} + \frac{S(C_{L_0} + C_{L_\alpha} \alpha)^2}{\pi b^2}, \]

where

\[ \sigma(\alpha) = \frac{1 + e^{-M(\alpha - \alpha_0)} + e^{M(\alpha + \alpha_0)}}{(1 + e^{-M(\alpha - \alpha_0)}) (1 + e^{M(\alpha + \alpha_0)})}. \]

Nondimensionalized coefficients of force in the body \( x \) and \( z \) axes are therefore given by

\[ C_X(\alpha) = -C_D(\alpha) \cos \alpha + C_L(\alpha) \sin \alpha \]
\[ C_{Xq}(\alpha) = -C_{Dq} \cos \alpha + C_{Lq} \sin \alpha \]
\[ C_{X\delta_e}(\alpha) = -C_{D\delta_e} \cos \alpha + C_{L\delta_e} \sin \alpha \]
\[ C_Z(\alpha) = -C_D(\alpha) \sin \alpha - C_L(\alpha) \cos \alpha \]
\[ C_{Zq}(\alpha) = -C_{Dq} \sin \alpha - C_{Lq} \cos \alpha \]
\[ C_{Z\delta_e}(\alpha) = -C_{D\delta_e} \sin \alpha - C_{L\delta_e} \cos \alpha. \]

Consequently, the force and torque expressed in the body frame are given by

\[ \mathbf{f}^b = mR^b I^g + \frac{\rho V_a^2 S}{2} \left( C_F(\alpha, \beta) + \ldots \right) \]
\[
\frac{1}{2V_a} C_{F_\omega} (\alpha) \omega_{b/I} + C_{F_u} (\alpha) u + \\
\rho S_{\text{prop}} C_{\text{prop}} e_3^\top u \left( V_a + e_3^\top u (k_{\text{motor}} - V_a) \right) \cdot \\
(k_{\text{motor}} - V_a) e_1 \\
\tau^b = \frac{\rho V_a^2 S}{2} C_{bc} \left( C_\tau (\alpha, \beta) + \frac{1}{2V_a} C_{\tau_0} \omega_{b/I} + \\
C_{\tau_0} u \right) - k_{T_p} (k_\Omega e_3^\top u)^2 e_1,
\]

where

\[
u = \begin{bmatrix} s_d & s_e & s_l & s_r \end{bmatrix}^\top
\]

\[
C_F (\alpha, \beta) = \begin{bmatrix} C_X (\alpha) \\ C_{Y_0} + C_{Y_\beta} \beta \\ C_Z (\alpha) \end{bmatrix}
\]

\[
C_{F_\omega} (\alpha) = \begin{bmatrix} 0 & C_{X_d} (\alpha) c & 0 \\ C_{Y_p} b & 0 & C_{Y_l} b \\ 0 & C_{Z_d} (\alpha) c & 0 \end{bmatrix}
\]

\[
C_{F_u} (\alpha) = \begin{bmatrix} 0 & C_{X_d} (\alpha) \delta_{\epsilon_{\text{max}}} & 0 & 0 \\ C_{Y_\delta} \delta_{\epsilon_{\text{max}}} & 0 & 0 & C_{Y_{\delta}} \delta_{\epsilon_{\text{max}}} \\ 0 & C_{Z_d} (\alpha) \delta_{\epsilon_{\text{max}}} & 0 & 0 \end{bmatrix}
\]

\[
C_{bc} = \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & b \end{bmatrix}
\]

\[
C_\tau (\alpha, \beta) = \begin{bmatrix} C_{l_0} + C_{l_\beta} \beta \\ C_{m_0} + C_{m_\alpha} \alpha \\ C_{n_0} + C_{n_\beta} \beta \end{bmatrix}
\]
$$C_{\tau_0} = \begin{bmatrix} C_{l_p}\ b & 0 & C_{l_r}\ b \\ 0 & C_{m_q}\ c & 0 \\ C_{n_p}\ b & 0 & C_{n_r}\ b \end{bmatrix}$$

$$C_{\tau_u} = \begin{bmatrix} C_{l_{\delta a}}\ \delta_{\text{max}} & 0 & 0 & C_{l_{\delta r}}\ \delta_{\text{max}} \\ 0 & C_{m_{\delta e}}\ \delta_{\text{max}} & 0 & 0 \\ C_{n_{\delta a}}\ \delta_{\text{max}} & 0 & 0 & C_{n_{\delta r}}\ \delta_{\text{max}} \end{bmatrix}.$$

### 6.4 LQR Controller

We employ the commonly used infinite horizon LQR controller that minimizes the cost functional

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

where $x$ is the state, $u$ is the control input, and $Q = Q^\top \geq 0$ and $R = R^\top \geq 0$ are user defined weighting matrices. This assumes that $x$ evolves linearly according to

$$\dot{x} = Ax + Bu,$$  \hspace{1cm} (6.10)

so that with a gain matrix $K$, the feedback control law may be given by

$$u = -Kx.$$

The gain matrix $K$ is computed by

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

and $P$ is computed by solving the continuous algebraic Riccati equation

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0.$$
Unfortunately, the formulation of LQR control we just described assumes a linear dynamic model and also drives the state $x$ to zero. As shown in Section 6.3, aircraft dynamics are highly nonlinear, and we do not want to drive the aircraft state to zero. To use LQR to drive the aircraft to a desired state, we define a state error $\tilde{x}$ between a desired state $\tilde{x}$ and the current state $x$, then redefine the cost functional as

$$ J = \int_0^\infty \left( \tilde{x}^\top Q \tilde{x} + u^\top R u \right) dt. $$

Error state dynamics are a nonlinear function of the form $\dot{\tilde{x}} = \tilde{f}(\tilde{x}, \dot{x}, u)$, but LQR requires a linear equation in the same form as (6.10). Fortunately, we can linearize $\tilde{f}(\tilde{x}, \dot{x}, u)$ about the state error $\tilde{x}$ and a reference control input $u_{ref}$. The input $u_{ref}$ is usually set to level flight trim conditions or to the control input from the previous control calculation.

Having chosen $u_{ref}$, error state dynamics are linearized as

$$ \dot{\tilde{x}} = A(\tilde{x}) \tilde{x} + B(\tilde{x}) u, $$

where

$$ A(\tilde{x}) = \frac{\partial \tilde{f}}{\partial \tilde{x}}, \quad B(\tilde{x}) = \frac{\partial \tilde{f}}{\partial u} \bigg|_{u=u_{ref}}. $$

The control input and gain matrix are now calculated by

$$ u = -K(\tilde{x}) \tilde{x} $$

$$ K(\tilde{x}) = R^{-1} B(\tilde{x})^\top P(\tilde{x}), $$

and $P(\tilde{x})$ is computed by solving the state-dependent continuous algebraic Riccati equation

$$ A(\tilde{x})^\top P(\tilde{x}) + P(\tilde{x}) A(\tilde{x}) - $$

$$ P(\tilde{x}) B(\tilde{x}) R^{-1} B(\tilde{x})^\top P(\tilde{x}) + Q = 0. $$
We have defined several terms as functions of $\tilde{x}$ to indicate a non-traditional LQR controller, where $u$ is computed every control iteration. This results in more precise and robust control than traditional LQR because computing $A(\tilde{x})$ and $B(\tilde{x})$ yields less linearization error.

Another important factor in the robustness of an LQR controller is the choice of $x$ and $\tilde{x}$. A typical choice is to include position, linear velocity, attitude, and angular velocity in the state. However, a fixed-wing aircraft must minimize sideslip, flying like a nonholonomic vehicle, which makes it difficult to map position error to a good control input. Instead, we define the LQR state, along with the desired state as

$$x = \begin{bmatrix} v_{b/I}^b \\ q_l^b \\ \omega_{b/I}^b \end{bmatrix}, \quad \tilde{x} = \begin{bmatrix} \dot{v}_{b/I}^b \\ \dot{q}_l^b \\ \dot{\omega}_{b/I}^b \end{bmatrix},$$

and choose the desired states based on vector fields in the following sections. In the next section, we describe the straight-line path geometry and the vector fields associated with orthogonal path error. This is then followed by a thorough description of how $\tilde{x}$ is computed, before finally determining how to numerically compute $\tilde{f}(\tilde{x}, \tilde{x}, u), A(\tilde{x}),$ and $B(\tilde{x})$.

### 6.4.1 Straight-Line Following with Vector Fields

It is natural to control the aircraft to follow a straight-line path by adjusting its course and flight path angles until the aircraft is on the desired path. Additionally if time of flight is not a concern, the aircraft should fly at its most efficient airspeed. To accomplish this task, we must first define the variables related to the straight-line path. Assume the aircraft has a set of waypoints and is currently from the $i^{th}$ waypoint $p_{w_i/I}$ to the $j^{th}$ waypoint $p_{w_j/I}$. Define a straight-line between these two waypoints with the line origin at $p_{w_i/I}$ and direction pointing from $p_{w_i/I}$ to $p_{w_j/I}$. The
line direction is then given by
\[ \mathbf{e}_{w_j/w_i}' = \frac{\mathbf{p}_{w_j/I}' - \mathbf{p}_{w_i/I}'}{\|\mathbf{p}_{w_j/I}' - \mathbf{p}_{w_i/I}'\|}. \]

We also need the error orthogonal to this line path defined in the inertial frame, which is given by
\[ \hat{\mathbf{p}}_I' = \left( I - \mathbf{e}_{w_j/w_i}' \left( \mathbf{e}_{w_j/w_i}' \right)^\top \right) \left( \mathbf{p}_b/I' - \mathbf{p}_{w_i/I}' \right). \]

Just as in [36], we can set the desired course angle \( \bar{\chi} \) to drive the aircraft toward the desired straight-line path. Let the course angle of the straight-line path be given by
\[ \chi_l = \text{atan}2 \left( \mathbf{e}_{w_j/w_i}^\top \mathbf{e}_{w_j/w_i}'(\mathbf{e}_{w_j/w_i}')^\top, \mathbf{e}_{w_j/w_i}^\top \mathbf{e}_{w_j/w_i}' \right), \]
and define a rotation from inertial to the straight-line path reference frame \( l \) as
\[ R_I^l = \begin{bmatrix} \cos \chi_l & \sin \chi_l & 0 \\ -\sin \chi_l & \cos \chi_l & 0 \\ 0 & 0 & 1 \end{bmatrix}. \]

The desired course angle is then computed by
\[ \hat{\chi} = \chi_l - \chi_\infty \frac{2}{\pi} \tan^{-1} \left( k_\chi \mathbf{e}_2^\top R_I^l \hat{\mathbf{p}}_l \right), \]
where \( k_\chi \) is a positive gain and \( \chi_\infty \) is the maximum allowed difference between \( \hat{\chi} \) and \( \chi_l \).

To control flight path angle \( \gamma \), first define the flight path angle of the straight-line path by
\[ \gamma = \tan^{-1} \frac{-\mathbf{e}_3^\top \mathbf{e}_{w_j/w_i}'}{\sqrt{\left( \mathbf{e}_1^\top \mathbf{e}_{w_j/w_i}' \right)^2 + \left( \mathbf{e}_2^\top \mathbf{e}_{w_j/w_i}' \right)^2}}. \]
The desired flight path angle is then computed by

\[ \dot{\gamma} = \gamma + \gamma_0 \frac{2}{\pi} \tan^{-1} \left( k_\gamma e_3 R_l^T \tilde{p}_l \right), \]

where \( k_\gamma \) is a positive gain and \( \gamma_0 \) is the maximum allowed difference between \( \dot{\gamma} \) and \( \gamma_l \).

Using the vector fields \( \dot{\chi} \) and \( \dot{\gamma} \), we can now compute the elements of \( \dot{x} \).

### 6.4.2 Choosing the Desired States

#### 6.4.2.1 Desired Linear Velocity

It is important that the controller always provides sufficient throttle command to maintain lift. Therefore, the desired velocity is given by

\[ \dot{\tilde{v}}_{b/I} = \dot{\tilde{V}}_a e_1 + R_l^b \dot{v}_{w/I}, \]  

(6.11)

where desired airspeed \( \dot{\tilde{V}}_a \) is defined by the user, typically at the aircraft’s most efficient cruising speed. This forces LQR to more consistently choose a throttle command that maintains sufficient forward air velocity for lift, regardless of aircraft attitude.

#### 6.4.2.2 Desired Attitude

It is natural for the aircraft to roll to govern change in heading, while using the rudder to control sideslip. To this end, let the desired roll angle be given by

\[ \dot{\phi} = \text{sign} (\dot{\chi} - \chi) \min (|\dot{\chi} - \chi|, \phi_{\text{max}}), \]
where the difference in course angle $\chi - \chi$ is wrapped between $-\pi$ and $\pi$, and

$$
\chi = \text{atan2} \left( \mathbf{e}_2^\top \left( R_I^b \right) v_{b/1}^b, \mathbf{e}_1^\top \left( R_I^b \right) v_{b/1}^b \right).
$$

The desired pitch is similarly defined by

$$
\dot{\theta} = \text{sign} (\gamma - \gamma) \min (|\gamma - \gamma|, \theta_{\text{max}}),
$$

where

$$
\gamma = \tan^{-1} \left( \frac{-\mathbf{e}_3^\top \left( R_I^b \right) v_{b/1}^b}{\sqrt{\left( \mathbf{e}_1^\top \left( R_I^b \right) v_{b/1}^b \right)^2 + \left( \mathbf{e}_2^\top \left( R_I^b \right) v_{b/1}^b \right)^2}} \right).
$$

Because changes in heading are managed by the roll angle, we simply leave the desired heading at the current value $\dot{\psi} = \psi$. Using (6.3), the desired unit quaternion is then computed from desired Euler angles as

$$
\dot{q}_I^b = q(\dot{\phi}, \dot{\theta}, \dot{\psi}).
$$

### 6.4.2.3 Desired Angular Rate

Desired angular rate can be chosen to satisfy the coordinated turn condition, which helps LQR choose more appropriate commands. Let $\dot{\phi} = \dot{\theta} = 0$ and the desired heading is then given by

$$
\dot{\psi} = \frac{g}{V_{\alpha}} \tan \dot{\phi}.
$$

These Euler angles must be rotated appropriately to get the desired angular rate vector in the aircraft’s body frame. Applying rotations in a 3-2-1 order, the desired body angular rate is given
by

\[ \dot{\omega}_{b/I} = \begin{bmatrix} \dot{\phi} \\ 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_{\dot{\phi}} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + R_{\dot{\theta}} R_{\dot{\theta}} \begin{bmatrix} 0 \\ \dot{\psi} \end{bmatrix} \]

(6.13)

where

\[ R_{\dot{\phi}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \]
\[ R_{\dot{\theta}} = \begin{bmatrix} \cos \dot{\theta} & 0 & -\sin \dot{\theta} \\ 0 & 1 & 0 \\ \sin \dot{\theta} & 0 & \cos \dot{\theta} \end{bmatrix} \]

### 6.4.3 LQR Error State Kinematics

Having fully define the desire LQR state, we now define the aircraft LQR error state as

\[ \tilde{x} = \bar{x} \oslash x = \begin{bmatrix} v^b_{b/I} - v^b_{b/I} \\ q^b_I \oslash q^b_I \\ \omega^b_{b/I} - \omega^b_{b/I} \end{bmatrix} \]

where the \( \oslash \) operator implies the appropriate subtraction operation per element of \( \tilde{x} \) and \( x \).
Prior to defining how $\dot{x}$ evolves in time, we must define how $x$ evolves. The state changes according to $\dot{x} = f(x, u)$, where

$$f(x, u) = \begin{bmatrix} \dot{v}^b_{b/I} \\ \dot{q}^b_I \\ \dot{\omega}^b_{b/I} \end{bmatrix},$$

and $\dot{v}^b_{b/I}$, $\dot{q}^b_I$, and $\dot{\omega}^b_{b/I}$ are given in (6.7), (6.8), and (6.9), respectively.

The state error evolves in time according to $\dot{\tilde{x}} = \tilde{f}(\tilde{x}, \tilde{x}, u)$, where $\tilde{f}(\tilde{x}, \tilde{x}, u) = \frac{d}{dt}(\tilde{x} \square x)$.

Computing the error state dynamics of fixed-wing aircraft is an error-prone and tedious process, due to complex dynamics. Therefore, we numerically approximate $\tilde{f}(\tilde{x}, \tilde{x}, u)$ up to second order accuracy by a central differencing scheme. Note that we pass both $\tilde{x}$ and $\tilde{x}$ into $\tilde{f}(\tilde{x}, \tilde{x}, u)$ because we need the ability to perturb error state for Jacobian calculations.

The function $\tilde{f}(\tilde{x}, \tilde{x}, u)$ starts by computing the current state as a function of error state and desired state by $x = \tilde{x} \boxplus (-\tilde{x})$. Next, perturb $x$ forward and backward in time by

$$x^+ = x \boxplus (f(x, u) \varepsilon)$$
$$x^- = x \boxplus (-f(x, u) \varepsilon),$$

where $\varepsilon$ is a small number. We then compute the perturbed desired states $\tilde{x}^+$ and $\tilde{x}^-$ from (6.11), (6.12), and (6.13) using the corresponding elements of $x^+$ and $x^-$. Finally, the error state kinematics are given by

$$\tilde{f}(\tilde{x}, \tilde{x}, u) = \frac{1}{2\varepsilon} (\tilde{x}^+ - \tilde{x}^-),$$

97
where

\[ \dot{\mathbf{x}}^+ = \ddot{\mathbf{x}}^+ \square \mathbf{x}^+ \]
\[ \dot{\mathbf{x}}^- = \ddot{\mathbf{x}}^- \square \mathbf{x}^- . \]

Using \( \tilde{f}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}) \) allows us to easily perturb \( \mathbf{x} \) and calculate the resulting error state dynamics, which is useful for numerically computing Jacobians. We compute the Jacobian of error dynamics with respect to state error and input by

\[ A(\mathbf{x}) = \frac{\partial \tilde{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \tilde{f}}{\partial x_1} & \cdots & \frac{\partial \tilde{f}}{\partial x_N} \end{bmatrix} \]
\[ B(\mathbf{x}) = \frac{\partial \tilde{f}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial \tilde{f}}{\partial u_1} & \frac{\partial \tilde{f}}{\partial u_2} & \frac{\partial \tilde{f}}{\partial u_3} & \frac{\partial \tilde{f}}{\partial u_4} \end{bmatrix} , \]

where for \( j = 1, 2, \ldots, N \) and \( k = 1, 2, 3, 4 \),

\[ \frac{\partial \tilde{f}}{\partial x_j} = \frac{1}{2\epsilon} \left[ \tilde{f}(\mathbf{x} + e_j\epsilon, \tilde{\mathbf{x}}, \mathbf{u}) - \tilde{f}(\mathbf{x} - e_j\epsilon, \tilde{\mathbf{x}}, \mathbf{u}) \right] \]
\[ \frac{\partial \tilde{f}}{\partial u_k} = \frac{1}{2\epsilon} \left[ \tilde{f}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u} + e_4\epsilon) - \tilde{f}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u} - e_4\epsilon) \right] . \]

### 6.5 State Estimation

We employ an extended Kalman filter to estimate the state of the aircraft, given an IMU, GPS, barometer, and pitot tube. Note that definitions given in Section 6.4 do not apply here, as we use many of the same variables. In this section, we first discuss the propagation of EKF state and covariance, followed by a discussion of the filter update step and individual sensor updates.
6.5.1 Propagation

In this chapter, we derive the EKF based on error state because we represent attitude with a quaternion. The reasons we use error state are that subtraction is not valid for quaternions and the covariance \( P = E\left[(x - \hat{x})(x - \hat{x})^\top\right] \) is never full rank treating the four quaternion elements like a vector [29]. We also take advantage of Lie group theory to simplify the derivation of error-state kinematics.

The EKF state is comprised of position, aircraft velocity, attitude, accelerometer bias, gyro bias, wind velocity, and barometer bias. The filter is also mechanized on the IMU, so the input of acceleration and angular rate predicts new states in time. Mathematically, we have

\[
x = \begin{bmatrix}
p^I_{b/I} \\
v^I_{b/I} \\
q^b_I \\
b_a \\
b_{\omega} \\
v^I_{w/I} \\
b_{\text{baro}}
\end{bmatrix}, \quad z_{imu} = \begin{bmatrix}
a^b_{b/I} \\
\Omega^b_{b/I}
\end{bmatrix},
\]
where $a^b_{b/l} = \bar{a}^b_{b/l} - b_a - \eta_a$ is true IMU acceleration, $\omega^b_{b/l} = \bar{\omega}^b_{b/l} - b_\omega - \eta_\omega$ is true IMU angular rate, and $\eta_*$ represents Gaussian noise. The state $x$ evolves in time according to

$$
\dot{x} = \begin{bmatrix}
    \dot{v}^l_{b/I} \\
    (R^b_I)^\top (a^b_{b/l} + g^I) \\
    \dot{\omega}^b_{b/I} \\
    \eta_a \\
    \eta_\omega \\
    \eta_w \\
    \eta_{baro}
\end{bmatrix},
$$

and here, we model IMU biases and wind as random walks whose derivatives are zero-mean and Gaussian.

The estimated state evolves in time according to

$$
\dot{\hat{x}} = g(\hat{x}, z_{imu}) = \begin{bmatrix}
    \dot{\hat{v}}^l_{b/I} \\
    (\hat{R}^b_I)^\top (\bar{a}^b_{b/I} - \hat{b}_a) + g^I \\
    \dot{\hat{\omega}}^b_{b/I} - \hat{b}_\omega \\
    0 \\
    0 \\
    0 \\
    0
\end{bmatrix},
$$

where $g^I$ is gravitational acceleration in the inertial frame. The state estimate $\hat{x}$ and its kinematics $g(\hat{x}, z_{imu})$ are the state-related terms that actually get computed when using the filter. In practice,
\( \hat{x} \) is propagated forward in time according to

\[
\hat{x}(t + \Delta t) = \hat{x}(t) \boxplus g(\hat{x}, z_{imu}) \Delta t,
\]

where the \( \boxplus \) operator is applied element-wise using the appropriate addition operator.

Having defined the true and estimated states along with their kinematics, we can now define the error state and its kinematics. The EKF error state is defined by

\[
\tilde{x} = \begin{bmatrix}
\tilde{p}_{b/I}^l \\
\tilde{q}_{l}^b \\
\tilde{b}_a \\
\tilde{b}_\omega \\
\tilde{v}_{w/I}^l \\
\tilde{b}_{baro}
\end{bmatrix} = \begin{bmatrix}
\hat{p}_{b/I}^l - \hat{p}_{b/I}^l \\
\hat{q}_{l}^b - \hat{q}_{b/I}^l \\
\hat{b}_a - \hat{b}_a \\
\hat{b}_\omega - \hat{b}_\omega \\
\hat{v}_{w/I}^l - \hat{v}_{w/I}^l \\
\hat{b}_b - \hat{b}_b
\end{bmatrix},
\]

(6.16)

but before deriving the kinematics of each element in \( \tilde{x} \), we require the following approximation of the true attitude represented by a rotation matrix as a function of the estimated attitude \( \hat{R}_l^b \) and attitude error \( \tilde{q}_l^b \in \mathbb{R}^3 \). Making use of (6.2) and (6.6), this approximation is given by

\[
R_l^b = R\left( \exp\left( \tilde{q}_l^b \right) \right) \hat{R}_l^b \\
\approx R\left( \begin{bmatrix}
1 \\
\frac{1}{2} \tilde{q}_l^b
\end{bmatrix} \right) \hat{R}_l^b \\
= \left( I - \begin{bmatrix} \tilde{q}_l^b \end{bmatrix}_x + \frac{1}{2} (\tilde{q}_l^b) (\tilde{q}_l^b)^\top \right) \hat{R}_l^b \\
\approx \left( I - \begin{bmatrix} \tilde{q}_l^b \end{bmatrix}_x \right) \hat{R}_l^b.
\]

(6.17)
We now proceed to derive the time evolution of each element in (6.16), making use of (6.1), (6.14), (6.15), (6.17), and dropping nonlinear error terms. We also briefly abuse notation by dropping most superscripts and subscripts to improve readability.

Differentiating position error yields

\[
\dot{\tilde{p}} = \tilde{p} - \dot{\hat{p}} = \mathbf{v} - \hat{\mathbf{v}} = \tilde{\mathbf{v}}.
\]

Differentiating velocity error yields

\[
\dot{\tilde{v}} = \tilde{v} - \dot{\hat{v}}
\approx \left( I - [\tilde{\mathbf{q}}] \times \right) \dot{\hat{\mathbf{R}}} \left( \tilde{\mathbf{a}} - \hat{\mathbf{b}} + \tilde{\mathbf{b}} + \eta \right) - \dot{\hat{\mathbf{R}}} \left( \tilde{\mathbf{a}} - \hat{\mathbf{b}} + \tilde{\mathbf{b}} + \eta \right)
\approx -\dot{\hat{\mathbf{R}}} [\tilde{\mathbf{a}} - \hat{\mathbf{b}}] \times \mathbf{q} - \dot{\hat{\mathbf{R}}} \left( \tilde{\mathbf{b}} + \eta \right).
\]

Rather than trying to differentiate through the $\square$ operation of the attitude error definition, we highlight that attitude is representing with an element of a Lie group and its derivative lies in its tangent space. Noting that the true and estimated derivatives lie on different tangent spaces, we move $\dot{\hat{\mathbf{q}}}$ from the tangent space of $\hat{\mathbf{q}}$ to the tangent space of $\mathbf{q}$ and perform regular subtraction because the tangent space is a vector space. We have

\[
\dot{\mathbf{q}} = \mathbf{q} - R \dot{\hat{\mathbf{R}}} \mathbf{q}.
\]
\[
\begin{align*}
\vec{\omega} - \mathbf{b}_\omega - \eta_\omega - R \hat{R}^\top (\vec{\omega} - \hat{\mathbf{b}}_\omega) \\
\approx \vec{\omega} - (\mathbf{b}_\omega + \hat{\mathbf{b}}_\omega) - \eta_\omega - \\
(I - [\hat{\mathbf{q}}]_\times) \hat{R} \hat{R}^\top (\vec{\omega} - \hat{\mathbf{b}}_\omega) \\
= [\hat{\mathbf{q}}]_\times (\vec{\omega} - \hat{\mathbf{b}}_\omega) - \hat{\mathbf{b}}_\omega - \eta_\omega \\
= - [\vec{\omega} - \hat{\mathbf{b}}_\omega]_\times \hat{\mathbf{q}} - \hat{\mathbf{b}}_\omega - \eta_\omega.
\end{align*}
\]

The errors in IMU biases, wind velocity, and barometer bias evolve according to

\[
\begin{align*}
\dot{\mathbf{b}}_a &= \eta_{ba} \\
\dot{\mathbf{b}}_\omega &= \eta_{b\omega} \\
\dot{\mathbf{v}}_w &= \eta_w \\
\dot{\mathbf{b}}_b &= \eta_{baro}.
\end{align*}
\]

Now, we define the evolution of state error as \( \dot{\mathbf{x}} = \ddot{g} (\mathbf{x}, \eta) \), where

\[
\ddot{g} (\mathbf{x}, \eta) = \begin{bmatrix}
\dot{\mathbf{p}}_{b/1}^I \\
\dot{\mathbf{q}}_{b/1}^p \\
\dot{\mathbf{q}}_f^p \\
\dot{\mathbf{b}}_a \\
\dot{\mathbf{b}}_\omega \\
\dot{\mathbf{v}}_{w/1}^I \\
\dot{\mathbf{b}}_{baro}
\end{bmatrix}, \quad \eta = \begin{bmatrix}
\eta_a \\
\eta_\omega
\end{bmatrix}.
\]
Having computed the error-state kinematics, we can compute the continuous-time derivative of covariance. This is given by

\[ \dot{P} = FP + PF^\top + GQ_uG^\top + Q_x, \]  

(6.18)

where \( Q_u \) is the IMU noise covariance, \( Q_x \) is the process noise covariance, and

\[
F = \frac{\partial \tilde{g}}{\partial \tilde{x}} =
\begin{bmatrix}
0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & \left( \hat{R}^b_{li} \right)^\top \left[ \hat{a}^b_{bi} - \hat{b}_a \right] \times & - \left( \hat{R}^b_{li} \right)^\top & 0 & 0 \\
0 & 0 & \left( \hat{\omega}^b_{bi} \right)^\top \left[ \hat{a}^b_{bi} - \hat{b}_a \right] \times & 0 & -I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
G = \frac{\partial \tilde{g}}{\partial \eta} =
\begin{bmatrix}
0 & 0 \\
- \left( \hat{R}^b_{li} \right)^\top & 0 \\
0 & -I \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

However, integrating (6.18) does not guarantee that the covariance matrix remains positive definite, and so we propagate covariance forward in time by a discrete process. This discrete propagation is defined up to second order by

\[ P_{k+1} = F_k P_k F_k^\top + G_k Q_u G_k^\top + Q_x \Delta t^2, \]
where

\[ F_k \approx I + F\Delta t + \frac{1}{2} FF\Delta t^2 \]
\[ G_k \approx \left( I + F\frac{\Delta t}{2} \right) G\Delta t. \]

### 6.5.2 Update

While the IMU predicts the state of the EKF, other sensors provide information to constrain the state, ideally close to truth. To perform an update for any given sensor, we must first model what we believe the measurement looks like as a function of our state estimate. But before defining how the update is performed, we first derive the measurement model for each sensor given as \( h_\ast \) or \( h_\ast \) and the corresponding observation matrix \( H_\ast \). Because of our error-state formulation, we must also compute the error model \( \tilde{h}_\ast \) in order to compute \( H_\ast \). We begin with the barometer, followed by the pitot tube, and lastly GPS.

The barometer measures absolute atmospheric pressure with some bias and noise. To measure aircraft altitude with this sensor, we record an average pressure at ground level of the flight location and subtract this from each incoming measurement. The pressure reading on the aircraft is then given approximately as

\[ h_{\text{baro}} = P_{\text{abs}} - P_{\text{ground}} + b_{\text{baro}} + \eta_{\text{baro}}, \]

where \( P_{\text{abs}} \) is the measured atmospheric pressure, \( P_{\text{ground}} \) is pressure at ground level, \( b_{\text{baro}} \) is the bias, and \( \eta_{\text{baro}} \) is noise. This can be written in terms of the state by first computing air density at ground level using the ideal gas law by

\[ \rho = \frac{P_{\text{ground}}M}{RT}, \]
where $M$ is the molar mass of air, $R$ is the universal gas constant, and $T$ is the air temperature in Kelvin.

The measurement model is then given in terms of the state by

$$h_{\text{baro}} = -\rho g e^T_3 p_{b/1} + b_{\text{baro}} + \eta_{\text{baro}},$$

and the estimated model is given by

$$\hat{h}_{\text{baro}} = -\rho g e^T_3 p_{b/1} + \hat{b}_{\text{baro}}.$$

Using (6.16), the error model is given by

$$\tilde{h}_{\text{baro}} = h_{\text{baro}} - \hat{h}_{\text{baro}} = -\rho g e^T_3 \tilde{p}_{b/1} + \tilde{b}_{\text{baro}} + \eta_{\text{baro}},$$

with the corresponding measurement Jacobian

$$H_{\text{baro}} = \frac{\partial h_{\text{baro}}}{\partial \tilde{x}} = \begin{bmatrix} -\rho g e^T_3 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The pitot tube is a differential pressure sensor that measures air velocity projected along the sensor’s measurement axis, which is typically aligned with the aircraft’s body $x$ axis. As it is impossible to perfectly align the sensor axis with the body $x$ axis, there exists some relative azimuth $\theta_a$ and elevation $\theta_e$ offset. The rotation from the body $x$ axis to the pitot tube’s measurement axis is given by

$$R^{\text{pt}}_b = R^{\text{pt}}_{ac} R^{\text{az}}_b$$
\[
\begin{bmatrix}
\cos \theta_e & 0 & -\sin \theta_e \\
0 & 1 & 0 \\
\sin \theta_e & 0 & \cos \theta_e
\end{bmatrix}
\begin{bmatrix}
\cos \theta_a & \sin \theta_a & 0 \\
-\sin \theta_a & \cos \theta_a & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \theta_e \cos \theta_a & \cos \theta_e \sin \theta_a & -\sin \theta_e \\
-\sin \theta_a & \cos \theta_a & 0 \\
\sin \theta_e \cos \theta_a & \sin \theta_e \sin \theta_a & \cos \theta_e
\end{bmatrix}
\]

Defining the differential pressure output of the pitot tube as \( P_{\text{diff}} \), along with bias \( b_{pt} \) and noise \( \eta_{pt} \), the measurement model is then given by

\[
h_{pt} = P_{\text{diff}} + b_{pt} + \eta_{pt}.
\]

We can approximate the pressure differential as a function of airspeed and density, where it only measures airspeed along the \( x \) axis in its own reference frame. Let \( \mathbf{v}_{a/I}^l \) denote airspeed in the inertial frame and also recall the wind triangle \( \mathbf{v}_{a/I} = \mathbf{v}_{b/I} - \mathbf{v}_{w/I} \). The measurement model as a function of the state is then given by

\[
h_{pt} = \rho \frac{1}{2} \left( \mathbf{e}_1^\top \mathbf{v}_{a/I}^{pt} \right)^2 + b_{pt} + \eta_{pt}
\]

and assuming that \( R_{b}^{pt} \) is known, the estimated model given as

\[
\hat{h}_{pt} = \rho \frac{1}{2} \left( \mathbf{e}_1^\top R_{b}^{pt} \hat{R}_{I}^{b} \left( \mathbf{v}_{b/I}^{l} - \mathbf{v}_{w/I}^{l} \right) \right)^2.
\]

Note that \( b_{pt} \) cannot be observed with the given sensor suite, so it is not part of the estimated measurement model. Using (6.1), (6.16), (6.17), and dropping nonlinear terms, the error model is
given by

\[
\tilde{h}_{pt} = h_{pt} - \hat{h}_{pt}
\]

\[
= \frac{\rho}{2} \left( e_1^\top R_{b}^{pt} R_{l} \left( v_{b/l}^l - v_{w/l}^l \right) \right)^2 + b_{pt} + \eta_{pt} - \frac{\rho}{2} \left( e_1^\top \hat{v}_{a/l}^{pt} \right)^2
\]

\[
= \frac{\rho}{2} \left( \left( e_1^\top R_{b}^{pt} \left( I - [\tilde{q}_{l}] \times \right) \right) \hat{R}_{l}^{b} \left( \tilde{v}_{b/l}^l + \tilde{v}_{b/l}^l - v_{b/l}^{l/w} \right) \right)^2 - \left( e_1^\top \hat{v}_{a/l}^{pt} \right)^2 \right) + b_{pt} + \eta_{pt}
\]

\[
= \frac{\rho}{2} \left( \left( e_1^\top \hat{v}_{a/l}^{pt} + e_1^\top R_{b}^{pt} \left[ \hat{q}_{a/l}^{b} \right] \times \hat{q}_{l}^{b} \right) + b_{pt} + \eta_{pt}
\]

\[
= \rho \left( e_1^\top \hat{v}_{a/l}^{pt} \right) e_1^\top R_{b}^{pt} \left[ \hat{q}_{a/l}^{b} \times \hat{q}_{l}^{b} \right]
\]

with the corresponding Jacobian

\[
H_{pt} = \frac{\partial \tilde{h}_{pt}}{\partial \tilde{x}}
\]

\[
= \left[ 0 \quad \frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{b/l}^l} \quad \frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{l}^{l/w}} \quad 0 \quad 0 \quad \frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{w/l}^l} \quad 0 \right]
\]

\[
\frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{b/l}^l} = \rho \left( e_1^\top \hat{v}_{a/l}^{pt} \right) e_1^\top R_{b}^{pt} \hat{R}_{l}^{b}
\]

\[
\frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{l}^{l/w}} = \rho \left( e_1^\top \hat{v}_{a/l}^{pt} \right) e_1^\top R_{b}^{pt} \left[ \hat{R}_{l}^{b} \left( \tilde{v}_{b/l}^l - \tilde{v}_{w/l}^l \right) \right] \times
\]

\[
\frac{\partial \tilde{h}_{pt}}{\partial \tilde{v}_{w/l}^l} = -\rho \left( e_1^\top \hat{v}_{a/l}^{pt} \right) e_1^\top R_{b}^{pt} \hat{R}_{l}^{b}.
\]
The GPS sensor provides position and velocity measurements in the earth-centered-earth-fixed (ECEF) coordinate system. Our estimator operates in a local, fixed, north-east-down (NED) coordinate frame, so the aircraft can be expressed in ECEF coordinates using a constant rotation $R_{ecef}^{I}$ and translation $p_{ecef}^{I}$. We define the latitude, longitude, and altitude of the inertial frame origin at the flight location and then compute $R_{ecef}^{I}$ and $p_{ecef}^{I}$ using coordinate transformations described in [65]. The measurement model is given in terms of the state by

$$h_{gps} = \begin{bmatrix} R_{ecef}^{I} p_{b/I}^{I} + p_{ecef}^{I} \\ R_{ecef}^{I} v_{b/I}^{I} \end{bmatrix},$$

and the estimated model is given by

$$\hat{h}_{gps} = \begin{bmatrix} R_{ecef}^{I} \hat{p}_{b/I}^{I} + p_{ecef}^{I} \\ R_{ecef}^{I} \hat{v}_{b/I}^{I} \end{bmatrix}.$$

Using (6.16), the error model is given by

$$\tilde{h}_{gps} = h_{gps} - \hat{h}_{gps} = \begin{bmatrix} R_{ecef}^{I} p_{b/I}^{I} \\ R_{ecef}^{I} v_{b/I}^{I} \end{bmatrix},$$

with the corresponding measurement Jacobian

$$H_{gps} = \frac{\partial \tilde{h}_{gps}}{\partial \hat{x}} = \begin{bmatrix} R_{ecef}^{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_{ecef}^{I} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Having defined all of the measurement models and observation matrices, we can now perform the EKF update step. When a non-IMU measurement $z_{*}$ is received, we compute the Kalman gain
by $K_s = PH_s^\top \left( R_s + H_s PH_s^\top \right)^{-1}$, where $R_s$ is the measurement covariance. Next, we perform a discrete update according to

$$\hat{x} = \hat{x} \oplus K_s (z_s \ominus h_s),$$

where the $\oplus/\ominus$ operators are applied element-wise with the appropriate addition or subtraction operation for the given element. The covariance is simultaneously updated according to the numerically stable Joseph form update, which is given by

$$P = (I - K_s H_s) P (I - K_s H_s)^\top + K_s R_s K_s^\top.$$

The full EKF, including the propagation and update steps, is summarized in Algorithm 2.

**Algorithm 2 EKF Summary**

1: Initialize $\hat{x}$ and $P$
2: for each $z_{imu}$ do
3: \hspace{1em} $F = \frac{\partial g}{\partial \hat{x}}$
4: \hspace{1em} $G = \frac{\partial g}{\partial \eta}$
5: \hspace{1em} $F_k = I + F \Delta t + \frac{1}{2} FF \Delta t^2$
6: \hspace{1em} $G_k = \left( I + F \frac{\Delta t}{2} \right) G \Delta t$
7: \hspace{1em} $\hat{x}_{k+1} = \hat{x}_k \oplus g(\hat{x}_k, z_{imu}) \Delta t$
8: \hspace{1em} $P_{k+1} = F_k P_k F_k^\top + G_k Q_k G_k^\top + Q_k \Delta t^2$
9: end for
10: if $z_s \neq z_{imu}$ then
11: \hspace{1em} $H_s = \frac{\partial h}{\partial \hat{x}}$
12: \hspace{1em} $K_s = PH_s^\top \left( R_s + H_s PH_s^\top \right)^{-1}$
13: \hspace{1em} $\hat{x} = \hat{x} \oplus K_s (z_s \ominus h_s)$
14: \hspace{1em} $P = (I - K_s H_s) P (I - K_s H_s)^\top + K_s R_s K_s^\top$
15: end if

### 6.6 Simulation Results

To investigate the performance and robustness of the proposed controller and estimator, we built a fixed-wing simulator in C++ that contains aircraft dynamics, sensor models, and random wind generation. We model the My Twin Dream airframe, where its aerodynamic parameters were
obtained with the open source software MachUp [66] and assume thrust parameters that result in a cruise throttle at level flight experienced in real flight experiments. In all simulations, the controller output is computed using estimates from the EKF. The aircraft is commanded to fly at its nominal cruise velocity of 16.38 m/s and is also commanded to follow line paths connected by four waypoints in a 400 by 400 meter square shape.

The Jacobians and gain matrix $A$, $B$, and $K$ are typically computed once about some equilibrium state and input, however, we compute these each time the controller runs to ensure a good linearization. This is much more computationally expensive, but our C++ implementation of the solution to the Continuous Algebraic Riccati Equation [67] uses LAPACK’s [68] sorted Schur Decomposition to solve the Riccati equation. This results in a total control calculation time averaging 140 microseconds with a standard deviation of 25 microseconds on an Intel i7-6700K processor clocked at 4.2 GHz.

Figures 6.1 and 6.2 show the commanded path in red and blue flight paths of the aircraft for 100 different simulations. The LQR portion of each simulation was initialized with uniformly random errors in aircraft parameters having a maximum of 30% error for any parameter. The EKF initial position, velocity, and attitude states were also initialized randomly with a maximum error of 10 meters for position, 2 m/s for velocity, 20° for roll/pitch, 40° for heading, and the remaining states start at zero. We attempted larger errors but at around 40%, roughly one out of 100 runs would result in aircraft instability, likely due to certain parameters being much more sensitive to error than others. Wind was initialized at a magnitude of 8.5 m/s directed toward the south-east direction with a downward component of 1 m/s. Wind also experienced a random walk where Gaussian noise was added to the current wind vector components every millisecond with standard deviations of 0.005 m/s in north/east directions and 0.001 m/s in the vertical direction. We linearized the LQR controller about the current state and previous input because we found that these linearization points give much better performance than linearizing about some trimmed state.

111
and input (e.g. level flight trim). To prevent unstable behavior, the desired roll is saturated at 30°, pitch at 30°, and state error is also saturated prior to the LQR gain matrix calculation.

![Figure 6.1: Altitude versus time of all Monte Carlo runs for LQR parameter sweep. Nominal path is shown by the dashed red line.](image)

We can see from Figure 6.1 that control of flight path angle maintains aircraft altitude fairly well, in spite of the aircraft parameter errors. There are dips in altitude mostly around corners, due to the sudden roll to compensate for lateral error. Lateral control is also successful, looking at Figure 6.2. The variance in these runs is not only caused by varying parameter error but also varying wind conditions and EKF convergence rates. It is also important to note that integrators in the LQR state are not needed because derived desired states from the error-based vector fields drives the aircraft toward the correct path, regardless of model error.

Figures 6.3 and 6.4 demonstrate successful state estimation and some robustness to initialization error. Because position and velocity are directly measured by GPS, the respective errors rapidly decrease to the noise and bias level of the GPS sensor. Looking closer, velocity experiences a jump in error after an initial decrease while the IMU bias and wind velocity estimates converge. Roll and pitch estimation is more accurate than heading, since they are more directly affected by the other EKF states.
Figure 6.2: Top-down view of all Monte Carlo runs for LQR parameter sweep. Nominal path is shown by the dashed red line.
Figure 6.3: Absolute value of EKF error in position, velocity, roll, pitch, and heading for all Monte Carlo runs.
From Figure 6.4, it is clear that the gyro bias is easily observable in the presence of other state errors. Accelerometer bias, however, struggles more than any other state to converge, due to the lack of constant accelerated flight. Similarly, wind velocity has difficulty converging initially because the pitot tube only measures one component of the wind vector. Therefore, we see that it tends to have step reductions in error as the aircraft makes its turns around the square trajectory.

![Graphs of accelerometer bias, gyro bias, and wind velocity over time](image)

**Figure 6.4:** Absolute value of EKF error in accelerometer bias, rate gyro bias, and wind velocity for all Monte Carlo runs.

### 6.7 Conclusions

This chapter has developed and numerically verified the performance of an LQR controller for a fixed-wing aircraft, using vector fields to compute the desired LQR state. Additionally, we also designed an error state EKF to estimate the aircraft state, given IMU, GPS, barometer, and
pitot tube data. The results presented in the chapter indicate robustness to model error in the LQR
controller and initialization error in the EKF. This robust behavior bodes well for use of these
algorithms on hardware platforms.
Chapter 7

Improving the Robustness of Visual-Inertial Extended Kalman Filtering

7.1 Introduction

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

Recent results in this area have demonstrated remarkable performance and capability [19, 69–73]. While smoothing methods and nonlinear batch optimization-based methods [74–76] have demonstrated significant advantages in terms of accuracy and consistency, they can be too computationally intense for many low-cost platforms. Filtering approaches have the advantage of being computationally efficient but can struggle in certain situations, due to significant nonlinearities and unobservability [53, 77, 78]. This chapter discusses filtering techniques for VI estimation that significantly increase robustness to these issues.

One major source of unobservability in VI filtering is the parameterization of feature locations. Feature locations parameterized in an inertial coordinate frame typically assume observability of

\[\text{Accepted to the 2019 International Conference on Robotics and Automation}\]
the transform to that frame. In many situations, however, this transform is unobservable, and estimation becomes inconsistent [53, 77, 78].

![Figure 7.1: A sample trajectory from the Monte Carlo simulation experiment. The multirotor flies a random path and observes landmarks on the ground.](image)

Recent methods have shown how to estimate features in the camera frame, rather than an inertial frame [19]. These parameterizations partition the states cleanly into observable and non-observable states, with global position and heading being completely unobservable. The unobservability of position and heading can be handled using the method proposed by [53], where the position and heading states are periodically reset, so that they remain observable and consistent. The global state and uncertainty are then calculated using other methods such as batch optimization, which are external to the Kalman filter.

Finally, many visual-inertial estimation approaches assume no knowledge about certain aspects of the system dynamics. In some applications, knowledge of specific parts of the system dynamics can help improve estimation accuracy and prevent divergence in certain modes at the expense of becoming less portable to other systems. [31, 79] For example, information regarding the speed capabilities of a multirotor aircraft can bound changes in estimates of depth to visual features. Leishman et al. [31] showed that including a linear model of drag on a multirotor significantly improves estimation accuracy. We will use this model to improve estimator robustness in this work.
Another source of nonlinearity and unobservability is the presence of filter states that are only partially observable or unobservable given specific vehicle motion. Examples of these states include IMU biases, depth to features and the above mentioned linear drag term. Brink [30] has shown that using a partial update can improve filter robustness to these so-called nuisance states, while maintaining consistency.

In this chapter, we extend the robocentric visual-inertial Kalman filtering approach described in [19] with the principles of relative navigation described in [29]. We also show that improving the dynamic model can significantly improve estimation accuracy of VI estimation applied to a multirotor and use the partial update formulation to deal with the additional nuisance state used in modeling drag. The chapter is organized as follows. In section 7.2, we describe several mathematical concepts and notation used throughout the chapter. In section 7.3, we briefly discuss the derivation of our baseline filter [19] with the improved dynamic model. Sections 7.3.2 and 7.3.3 discuss the measurement models used and sections 7.3.4 and 7.3.5 detail the keyframe reset and partial update steps, respectively. Finally, section 7.4 details a Monte Carlo simulation experiment and compares the performance of the proposed improvements in terms of accuracy and consistency.

7.2 Notation

The following definitions are used throughout the chapter.

- $e_i$: Unit vector with a one in the $i^{th}$ element
- $p_{b/I}$: Position of the body, with respect to the world frame, expressed in the world frame
- $v_{b/I}$: Velocity of the body frame, with respect to the world frame, expressed in the body frame
- $q_{b/I}$: Quaternion describing rotation from the world frame to the body frame
- $\beta_a$: Accelerometer bias
- $\beta_{\omega}$: Rate gyro bias
- $b$: Linear drag coefficient
\( \zeta^c_{i/c} \) Unit vector directed at the \( i^{th} \) feature from the camera origin, expressed in the camera frame

\( \mathbf{q}^c_i \) Quaternion which describes the rotation from the camera \( \mathbf{e}_3 \) axis to the unit vector \( \zeta^c_{i/c} \)

\( \rho_i \) Inverse distance to the \( i^{th} \) feature

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

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

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

\[
\mathbf{v} \times \mathbf{w} = \mathbf{v}^\wedge \mathbf{w}.
\]

To convert back to a vector from a skew-symmetric matrix, we use the \( \leftarrow \) operator, so that

\[
(\mathbf{v}^\wedge)^\leftarrow = \mathbf{v}.
\]

### 7.2.1 Quaternions

We will use Hamiltonian notation for unit quaternions \( \in S^3 \)

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

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

\[
R(\mathbf{q}) = (2q_0^2 - 1)I - 2q_0 \mathbf{q}^\wedge + 2\mathbf{q} \mathbf{q}^\top \in SO(3).
\]
This definition results in $R^b_a \mathbf{r}^a$ being interpreted as the original vector $\mathbf{r}^a$ expressed in the new coordinate frame $b$.

The exponential mapping for a unit quaternion is defined as

$$
\exp : \mathfrak{so}(3)^\vee \sim \mathbb{R}^3 \rightarrow S^3
$$

$$
\exp(\delta) \triangleq \begin{bmatrix}
\cos \left( \frac{\|\delta\|}{2} \right)

\sin \left( \frac{\|\delta\|}{2} \right) \frac{\delta}{\|\delta\|}
\end{bmatrix},
$$

with the corresponding logarithmic map defined as

$$
\log : S^3 \rightarrow \mathfrak{so}(3)^\vee \cong \mathbb{R}^3
$$

$$
\log(\mathbf{q}) \triangleq 2 \, \operatorname{atan2} \left( \|\bar{\mathbf{q}}\|, q_0 \right) \frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}.
$$

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

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

Eq. (7.1) is significant because the covariance is parameterized in the Lie algebra $\mathfrak{so}(3)$ (which is a vector space) of $SO(3)$ and therefore, can be used in a Kalman filtering framework.

### 7.2.2 ⊗ and □ Operators

Hertzberg et al. [40] describe a new syntax that simplifies working with Lie groups in a filtering and optimization framework by introducing the ⊗ and □ operators. This syntax allows us to work with elements of Lie groups in a notation similar to that of vectors and will be used to describe
our filter derivation. The $\boxplus$ and $\boxminus$ operators are defined differently for different groups. For $\mathbb{R}^n$, they are simply defined as the typical addition and subtraction operations. For attitude quaternions $\in S^3$, these operators are defined by

$\boxplus : S^3 \times \mathbb{R}^3 \rightarrow S^3$

$q \boxplus \theta \triangleq q \otimes \exp(\theta)$

$\boxminus : S^3 \times S^3 \rightarrow \mathbb{R}^3$

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

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

$q^b_i(t + dt) = q^b_i(t) \boxplus \theta$

$\theta = q^b_i(t + dt) \boxminus q^b_i(t)$.

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

### 7.2.3 Feature Bearing Parameterization

As in [19], we parameterize the feature bearing states in the camera frame as rotations $q^c_i \in S^3 \sim R^c_i \in SO(3)$, which describe the rotation from the camera $e_3$ axis to the unit vector directed at the feature. The unit vector directed at feature $i$ with respect to the camera frame $c$ is then defined by

$\xi^c_{i/c} = (R^c_i)^\top e_3 \in S^2 \subset \mathbb{R}^3$,

where we can see that this simply expresses the direction of the feature in the camera frame.
The difference between two unit vectors $\zeta_i \sqcap \zeta_j$ can be described using axis-angle representation, where the direction of the axis of rotation is orthogonal to both of the unit vectors, and its length is scaled by the magnitude of rotation, as shown in Figure 7.2. There are actually only two degrees of freedom in this parameterization because rotation about either feature vector does not change unit vector direction. To remove the redundant degree of freedom, we note that the axis of shortest rotation is always in the plane normal to $\zeta_{i/c}$ and define a projection matrix

$$T_{\zeta_i} = \left( R_{\zeta_i}^c \right)^\top [e_1 \ e_2] \in \mathbb{R}^{3 \times 2},$$

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

We must then define the $\sqcap$ and $\sqcup$ operators associated with feature bearing vectors as

$$\sqcup : SO(3) \times \mathbb{R}^2 \rightarrow SO(3)
q_{\zeta_i} \sqcup \delta \triangleq \exp(T_{\zeta_i} \delta) \otimes q_{\zeta_i}$$

$$\sqcup : SO(3) \times SO(3) \rightarrow \mathbb{R}^2
q_{\zeta_i} \sqcup q_{\zeta_j} \triangleq \theta T_{\zeta_i}^\top s,$$

where the axis $s$ and angle $\theta$ between the two feature direction vectors are given by

$$\theta = \cos^{-1} \left( \zeta_i^\top \zeta_j \right)$$

$$s = \frac{\zeta_i \times \zeta_j}{\|\zeta_i \times \zeta_j\|}.$$

With only two degrees of freedom, and with all feature vectors referencing the camera $e_3$ axis, there are an infinite number of unit quaternions which can be used to represent the same unit
vector. The difference between these rotations is some angle of rotation about the bearing vector itself. This is removed by the projection operation and can therefore be neglected. Reference [69] explores more deeply the validity of the $\boxplus$ and $\boxdot$ operators under this assumption.

**Figure 7.2:** Illustration of feature bearing vector geometry including error on the plane orthogonal to the bearing vector.

### 7.3 Derivation

In this section, we derive the relevant geometry and dynamics to fully describe and implement the filter proposed in this chapter.
7.3.1 State Definition and Kinematics

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

\[
    x = \begin{bmatrix}
        p_{b/I}^I & v_{b/I}^b & q_{I}^b & \beta_{a} & \beta_{\omega} & b & q_{c}^{\xi} & \rho_{1} & \cdots & q_{c}^{\xi} & \rho_{n}
    \end{bmatrix},
\]

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

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

where \( \boxdot \) for objects composed of multiple group elements implies the use of the appropriate \( \boxdot \) operator for each element.

Given measured acceleration \( \bar{a}_{b/I}^b \) and measured angular velocity \( \bar{\omega}_{b/I}^b \), the state has kinematics \( \dot{x} = f(x, u + \eta) \) with the elements of \( f \) given by [31] and defined as

\[
    \begin{align*}
    p_{b/I}^I &= (R_{I}^b)^\top v_{b/I}^b \\
    v_{b/I}^b &= e_{3}e_{3}^\top a_{b/I}^b + R_{I}^b g^I - b \left( I - e_{3}e_{3}^\top \right) v_{b/I}^b - (\omega_{b/I}^b)^\wedge v_{b/I}^b \\
    q_{I}^b &= \omega_{b/I}^b \\
    \dot{\beta}_{a} &= 0 \\
    \dot{\beta}_{\omega} &= 0 \\
    \dot{b} &= 0 \\
    \dot{q}_{c}^{\xi} &= -T_{\xi}^c \left( \omega_{c/I}^c + \rho_{i} (\xi_{c}^{\xi})^\wedge v_{c/I}^c \right) \\
    \dot{\rho}_{i} &= \rho_{i}^2 (\xi_{c}^{\xi})^\top v_{c/I}^c,
    \end{align*}
\]

where \( b \) is a linear drag term [31], \( u = \begin{bmatrix} a_{b/I}^b & \omega_{b/I}^b \end{bmatrix} \) is the input, \( \eta = [\eta_{a} \quad \eta_{\omega}] \) is input noise, and

\[
    a_{b/I}^b = \bar{a}_{b/I}^b - \beta_{a} - \eta_{a}
\]
\[ \omega_{b/I}^b = \dot{\omega}_{b/I} - \beta \omega - \eta \omega. \]

Camera linear and angular velocities are also given by

\[
v_{c/I}^c = R_b^c \left( v_{b/I}^b + \left( \omega_{b/I}^b \right) \wedge \mathbf{p}_{c/b}^b \right),
\]

\[
\omega_{c/I}^c = R_b^c \omega_{b/I}^b,
\]

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

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

\[
\hat{x}(t + dt) = \hat{x}(t) \boxplus f(\hat{x}(t), u(t)) dt
\]

\[
\hat{x}^+ = \hat{x} \boxplus K(z \boxminus h(\hat{x})),
\]

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

**7.3.2 Camera Measurement Model**

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

\[
h_{\text{cam}}(x) = \frac{1}{e_{\overline{3}}^c} \zeta \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix} \zeta^c + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}. \tag{7.3}
\]
The Jacobian $\frac{\partial h_{\text{cam}}}{\partial \mathbf{x}}$ of the camera measurement model is given by

$$H_{\text{cam}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & H_1 & 0 & \cdots & H_n & 0 \end{bmatrix},$$

where using the chain rule, we have

$$H_i = \frac{1}{e_3^\top \xi_{i/c}^c} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix} \left( \begin{bmatrix} \xi_{i/c}^c \bar{e}_3^\top \\ \bar{e}_3 \xi_{i/c}^c - I_{3 \times 3} \end{bmatrix} \right) \wedge \left( \xi_{i/c}^c \right)^\top \left( T_{\xi_i} \right).$$

### 7.3.3 Accelerometer Measurement Model

Using the multirotor drag model from [31] in (7.2) provides the benefit that velocity becomes directly observed by the accelerometer (assuming a linear drag constant). It is assumed that the accelerometer measures total acceleration of the body, neglecting gravity, in addition to a constant bias $\beta_a$ and zero-mean white noise $\eta_a$. If we also assume that thrust $T$ acts only along the body $e_3$ axis, we can consider just the body $e_1$ and $e_2$ axes, removing any dependence of the measurement on $T$. The measurement model is then given by

$$h_{\text{acc}}(\mathbf{x}) = I_{2 \times 3} \left( -b v_{b/l}^b + \beta_a + \eta_a \right).$$

(7.4)

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

$$H_{\text{acc}} = \begin{bmatrix} 0 & -b I_{2 \times 3} & 0 & I_{2 \times 3} & 0 & -I_{2 \times 3} v_{b/l}^b & 0 & \cdots \end{bmatrix}.$$  

### 7.3.4 Keyframe Reset

As shown in [53] and [29], performing a keyframe reset when global states are unobservable can dramatically improve filter consistency and accuracy. A keyframe reset is performed by resetting the global position and heading states to zero and updating the covariance matrix appro-

127
priately. Each reset step results in a new node being declared in a pose graph structure, which can then incorporate loop closures and other measurements as part of a global optimization routine. Figure 7.3 shows an illustration of the coordinate frames involved in the keyframe reset. Here, we note that our setup slightly differs from [53] and [29] in that there is no altimeter measurement available, so altitude is also unobservable, and we must also reset that state. Therefore, we can see in Figure 7.3 that node frames are co-located with keyframes, instead of on a ground plane.

Figure 7.3: Keyframes $k_i$ are declared at periodic intervals along the trajectory flown by the MAV, while node frames $n_i$ are associated with each keyframe and are gravity-aligned but co-located with each keyframe. The current body frame $b$ is estimated with respect to the most recent keyframe. New keyframes are declared when less than 25 percent of the features present in the previous keyframe are still present. This promotes observability of the transform between $b$ and the most recent keyframe.

7.3.5 Partial Update

A common difficulty faced in visual-inertial navigation is the estimation of nuisance states which may only be partially observable during many maneuvers. In the filter derived in this chapter, these states include the inverse depth to each feature $\rho_i$, accelerometer and gyro biases $\beta_a$ and $\beta_\omega$, and the linear drag term $b$. As noted in [30], estimating these terms in the traditional manner can cause filter divergence but ignoring them or considering them as known constants may produce an overconfident estimate. Because of the abundance of these states in our system, we employ a
version of the partial-update Schmidt-Kalman filter proposed by [30]. This method allows the
designer to tune the effect of a measurement update on the $i^{th}$ state with a scalar gain $\gamma_i$, while cor-
rectly estimating uncertainty in these partially-updated states. While this method loses optimality
guarantees in estimating these states in a linear Kalman-filtering framework, it has been shown to
speed up convergence of these nuisance states by limiting the effect of linearization errors when
applied to the non-linear IMU-camera extrinsics estimation problem [30].

A drawback of the formulation given in [30] is the intermediate calculation of $\hat{x}^+$ and $P^+$. We
can manipulate these equations to remove this intermediate calculation and maintain algebraic
equivalence. Let us first define $\lambda_i = 1 - \gamma_i$, and for $N$ states, also define

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

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

$$\hat{x}^{++} = \hat{x}^- \boxplus (\lambda \circ K (z \boxdot h(\hat{x}^-)))$$

$$P^{++} = P^- + \Lambda \circ \left( (I - KH)P^- (I - KH)^\top + 
KRK^\top - P^- \right),$$

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

$$1 = [1 \quad 1 \quad \cdots \quad 1]^\top$$

$$\Lambda = 1\lambda^\top + \lambda 1^\top - \lambda\lambda^\top.$$
7.4 Results

To identify improvements to consistency and accuracy, we employed a Monte Carlo (MC) simulation of a MAV with a nonlinear aerodynamic model. The multirotor was commanded to fly approximately five meters above a simulated ground plane at a constant forward velocity of one meter per second. The commanded heading for each iteration evolved according to a random walk. A fourth-order Runge Kutta integration scheme was used for the truth comparison. A sample trajectory is shown in Figure 7.1.

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

We implemented four different filters for comparison. The baseline (BL) filter is the same filter derived in [19] except with the measurement model for features given as (7.3) rather than the patch-based model in the original work. This was primarily done to simplify modeling in the simulation environment and to guarantee that all filters received the same measurements. The second filter modifies the baseline with a linear drag term (DT) as shown in (7.4), while the third filter modifies the baseline with keyframe resets (KF) given in Section 7.3.4. The fourth filter augments the baseline with a drag term, keyframe reset, and a partial update (KF+DT+PU). Each of these filters were given identical inputs and measurements for each MC iteration, and the relevant pro-
cess and sensor noise covariance matrices used in each filter were derived from the corresponding simulation parameters.

Inverse depth to each feature was initialized using the recommended values in [80] of \( \rho_0 = \frac{1}{2d_{\text{min}}} \) and \( R_0 = \frac{1}{16d_{\text{min}}} \) with a minimum distance to each feature assumed to be \( d_{\text{min}} = 2 \) meters. To deal with negative depth estimates, we used the method in [81], where any negative depth estimates were immediately re-initialized to \( d_{\text{min}} \) and the covariance appropriately expanded to account for the additional uncertainty. Because keyframes are not tied to a specific image in this estimator (as opposed to the implementation in [29]) new keyframes were declared when more than one half of the features present at the declaration of the previous keyframe were lost.

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

\[
J_{\text{RMS}} = \left\| \mathbf{J}_n \mathbf{T}_n \right\| = \left\| \begin{bmatrix} \hat{\mathbf{p}}_{b/n} - \mathbf{p}_{b/n}^n \\ \hat{\mathbf{q}}_{b/n} - \mathbf{q}_{b/n}^n \end{bmatrix} \right\|.
\]

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

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

\[
\varepsilon = \begin{cases} 
(\hat{T}_n^b \boxprod T_n^b)^\top P_{T_n^b} (\hat{T}_n^b \boxprod T_n^b) & \text{if KF} \\
(\hat{T}_I^b \boxprod T_I^b)^\top P_{T_I^b} (\hat{T}_I^b \boxprod T_I^b) & \text{otherwise}
\end{cases}.
\]

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

We performed 2016 MC iterations of a five-minute simulation study and calculated the RMSE and NEES at each time step (250 Hz). The average RMSE and NEES over time for each filter in the MC simulation study are shown in Figure 7.4. In this plot, we see that the RMSE of each filter decreases as each filter evolves in time and converges on the unknown biases. A histogram of the RMSE and NEES for each estimator at the final time is given in Figure 7.5.

It is clear from the results of this study that using keyframe resets dramatically affects RMSE and NEES, resulting more accurate and consistent pose estimates. In filters without a keyframe reset step, the unobservable position and heading states cause the filter to become increasingly inconsistent over time, resulting in large linearization errors and suboptimal sensor fusion [53].

It appears that while the drag term improves pose accuracy, it degrades consistency. This is not altogether unexpected as the drag term is only partially observable and the resulting linearization error on the drag term measurement update (7.4) causes the filter to become overconfident. The improved accuracy, however comes from better state integration which arises from the improved dynamic model.

The overconfidence caused by the drag term can be mitigated by using a partial update. In the (KF+DT+PU) filter, \(\gamma_b\) was set to 0.02, which reduced the effect of linearization error on the state
Figure 7.4: Average RMSE of the transform from the most recent keyframe (top) and average NEES (bottom) for each filter over the entire simulation time over 2016 runs.

and covariance. Figure 7.6, shows a single run of the drag term with and without the partial update. In this plot, the drag term without the partial update produces oscillations corresponding to changes in attitude. This is most certainly incorrect as we have no reason to believe that the constant drag term should be correlated with attitude. The partial update attenuates these oscillations and allows us to benefit from the improved dynamic model. A similar effect is observed in accelerometer and gyroscope bias estimates. We see in Figure 7.7, that without the drag term, accelerometer bias estimates become strongly correlated with attitude. Again, the partial update damps this oscillatory response and keeps the estimate more aligned with truth.

7.5 Conclusions

We have shown that augmenting visual-inertial extended Kalman filtering with keyframe resets, an improved dynamic model, and partial updates greatly improves accuracy and consistency in VI filtering. This is clearly demonstrated in Figures 7.4 and 7.5. The use of keyframe resets improves
Figure 7.5: The $\chi^2$ distribution with six degrees of freedom compared against each filter at the final simulation time of 5 minutes using 2016 samples.

Figure 7.6: Drag term estimates of a single MC iteration with and without the partial update compared against truth.
Figure 7.7: Accelerometer biases of a single MC iteration with and without the partial update compared against truth.
filter consistency and accuracy without any observed negative consequences. Augmenting the dynamic model with a linear drag term also improves accuracy but at the expense of degraded consistency. This inconsistency can be directly mitigated through the use of a partial update, thus, providing better accuracy from the improved dynamic model, while maintaining filter consistency. Finally, the combination of all three proposed improvements was shown to improve filter accuracy and consistency over the baseline filter.
Chapter 8

A Visual-Inertial Extended Kalman Filter Using Image Coordinates

8.1 Introduction

The development of low-cost inertial measurement units (IMU) and cameras in the last twenty years has provided a means of low-cost, accurate state estimation without GPS. There are also many circumstances when camera-IMU sensor combinations are a good choice, such as operating in unmapped indoor environments, under cost constraints, etc. These benefits and use cases have motivated the scientific community to enhance the capabilities of visual-inertial (VI) navigation.

A few primary methods of solving the VI navigation problem have emerged, namely analytical, filter, and nonlinear optimization methods. The authors in [82] derive the observability conditions and an analytical method to compute initial feature position, camera velocity, gravity direction, and accelerometer bias for some given window of data containing no gyroscope bias. The works of [19, 52, 70] develop Kalman filters to fuse IMU and camera information, resulting in filter-based visual-inertial odometry (VIO) algorithms. The filters developed in [52, 70] cite the benefit of performing a constrained update, which does not explicitly add error into the filter, while [19] claims to be robust to initial error. In [72, 75, 83], the authors develop VIO algorithms using nonlinear optimization techniques. These have proven to be some of the more accurate VIO algorithms because they perform inference over a window of data to find the maximum-likelihood estimate, however, they are sensitive to initial error. One of the most accurate VIO algorithms to date is given in [75], where a unique initialization scheme is used to significantly improve robustness in practical use.
A significant issue with VIO is the need for a precise knowledge of transformations between sensors and characterization of the potential time offset between the IMU and camera images. These parameters can be calculated online or through post-processing a batch of sensor data. In [33], the authors formulate an extended Kalman filter to estimate the time offset between camera and IMU online, while [34] uses nonlinear optimization over a window of measurements to achieve the same result. In [84], an offline, optimization-based calibration routine is developed using constraints to compute time offsets and spatial transforms. The authors in [32] perform a thorough calibration of a camera-IMU system, accounting for relative transformations and time offsets, as well as other parameters like camera exposure time. Precise knowledge of camera intrinsic parameters is also important to robust VIO performance, but these are easy to estimate offline and are not likely to change during operation. Hence, they are not often estimated online.

The two main contributions of this chapter are an EKF-based VIO by parameterizing the bearing to features directly as image coordinates in the state and an offline camera-IMU calibration that accounts for the body-level frame, needed for drag-enabled estimation of multirotors. The EKF developed in this chapter is most similar to [19], but instead of representing landmark directions with quaternions, we use pixel positions in the image. The motivations for this representation of landmark directions are to simplify filter mathematics and improve efficiency by removing the need for an iterated update [69] as each image is received. To improve the consistency and accuracy of the EKF, we also employ keyframe resets [53], partial updates [30], and a linear drag coefficient [31]. Additionally, our filter does not attempt to estimate the rotation from IMU to camera or the camera-IMU time offset online for a few reasons: we can assume that these parameters are static, the observability of these parameters depends on camera motion, and we wish to limit the number of states in the Kalman filter to improve robustness. Instead, we employ an offline, nonlinear optimization to obtain the transforms necessary for accurate VIO. This calibration routine is mostly inspired by [34] and [75] but additionally estimates both the time offset and coordinate transform between the IMU and a motion capture system, the rotation from the IMU coordinate...
frame to the multirotor body frame, where drag is modeled, and a linear drag coefficient. The rotation from IMU to body is especially important when using a drag-enabled filter. Error in this rotation may result in drag forces being applied in the wrong directions, thereby destabilizing the filter.

The remainder of the chapter is outlined as follows. Section 8.2 gives the mathematical preliminaries needed throughout the chapter. This is followed by a derivation of the filter in Section 8.3 and then a derivation of the offline calibration routine in Section 8.4. Lastly, Section 8.5 presents numerical results of the EKF and calibration routine, followed by some concluding remarks.

8.2 Mathematical Preliminaries

\( \hat{a} \) Estimate of true variable \( a \)

\( \bar{a} \) Measurement of \( a \)

\( \bar{\bar{a}} \) Error of \( a \)

\( \dot{a} \) Time derivative \( a \)

Superscript

\( I \) Inertial coordinate frame

\( b \) Aircraft body coordinate frame

\( u \) IMU coordinate frame

\( \top \) Matrix transpose

Subscript

\( a/b \) The state of frame \( a \) with respect to frame \( b \)

This chapter uses the skew-symmetric matrix operator defined by \([v]_\times w \triangleq v \times w\), where \( v \in \mathbb{R}^3 \) and \( w \in \mathbb{R}^3 \) are arbitrary vectors. We also frequently use the cross product identity for order
reversal, which is given by

\[ \mathbf{v} \times \mathbf{w} = -\mathbf{w} \times \mathbf{v}. \]  

(8.1)

8.2.1 Quaternions

We use Hamilton notation for unit quaternions, namely

\[ \mathbf{q} = q_0 + q_x \mathbf{e}_1 + q_y \mathbf{e}_2 + q_z \mathbf{e}_3 = \begin{bmatrix} q_0 \\ -\mathbf{q} \end{bmatrix} \in S^3, \]

where \( q_0 \) and \( \mathbf{q} \) are the scalar and vector part of \( \mathbf{q} \), respectively. Quaternion multiplication is defined by

\[ \mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0 q_0 - \mathbf{p}^\top \mathbf{q} \\ q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{p} \times \mathbf{q} \end{bmatrix}. \]

The associated rotation matrix is constructed by

\[ R(\mathbf{q}) = (2q_0^2 - 1) I - 2q_0 [\mathbf{q}]_\times + 2\mathbf{q} \mathbf{q}^\top \in SO(3). \]  

(8.2)

Because we are representing rotations with unit quaternions, we frequently interchange quaternions and rotation matrices, for example \( R^b_a = R(\mathbf{q}^b_a) \). Also, the rotations used throughout this chapter are considered passive rotations. For example, a vector defined in the coordinates of frame \( a \) may be expressed in frame \( b \) by \( \mathbf{r}^b = R^b_a \mathbf{r}^a \). Coordinate frames are rotated, not the vectors themselves.
This chapter occasionally uses 3-2-1 Euler angles. The composition of a unit quaternion from roll $\phi$, pitch $\theta$, and yaw $\psi$ angles is given by

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

(8.3)

and the decomposition is given by

$$
\phi = \tan^{-1} \frac{2q_0q_x + 2q_yq_z}{q_0^2 + q_x^2 - q_y^2 - q_z^2}
$$

(8.4)

$$
\theta = \sin^{-1}(2q_0q_y - 2q_xq_z)
$$

(8.5)

$$
\psi = \tan^{-1} \frac{2q_0q_z - 2q_xq_y}{q_0^2 + q_x^2 - q_y^2 - q_z^2}.
$$

(8.6)

The exponential map for a unit quaternion is defined as

$$
\exp : \mathbb{R}^3 \rightarrow S^3
$$

$$
\exp(\delta) = \begin{bmatrix}
\cos \left(\frac{\|\delta\|}{2}\right) \\
\sin \left(\frac{\|\delta\|}{2}\right) \frac{\delta}{\|\delta\|}
\end{bmatrix},
$$

with the corresponding logarithmic map defined as

$$
\log : S^3 \rightarrow \mathbb{R}^3
$$

$$
\log(q) = 2 \text{atan2} \left( \frac{\|q\|, q_0} {\|q\|} \right).$

141
To avoid numerical issues when $\|\delta\|$ and $\|\bar{q}\|$ are small, we also employ the approximations

\[
\exp(\delta) \approx \begin{bmatrix} 1 \\ \frac{\delta}{2} \end{bmatrix} \tag{8.7}
\]

\[
\log(q) \approx 2 \text{sign}(q_0) \bar{q}. \tag{8.8}
\]

The exponential map can be used to exactly integrate the kinematic equations of motion. For example, attitude kinematics are integrated as

\[
q^b_b(t + \Delta t) = q^b_b(t) \otimes \exp(\omega^b_b/\Delta t), \tag{8.9}
\]

assuming a constant angular velocity over the interval $\Delta t$.

Similarly, the logarithmic map is often used to cast the difference between two members of a Lie group to a vector space. For example, inverting (8.9) yields the difference between two subsequent quaternion states in terms of the average angular velocity experienced over the interval

\[
\omega^b_b/\Delta t = \log \left( \left( q^b_b(t) \right)^{-1} \otimes q^b_b(t + \Delta t) \right). \tag{8.10}
\]

### 8.2.2 Rigid Transforms

We define a rigid transform as a tuple of the translation vector and unit quaternion $T^b_a = (p^a_{b/a}, q^b_a)$, which allows for an efficient implementation. Let $T^b_a = (p^a_{b/a}, q^b_a), T^c_b = (p^b_{c/b}, q^c_b)$ then the multiplication of transforms is given by

\[
T^c_a = T^b_a \circ T^c_b = (p^a_{b/a} + R \left( \left( q^b_a \right)^{-1} \right) p^b_{c/b}, q^b_a \otimes q^c_b)
\]

142
with the inverse given by
\[
(T_a^b)^{-1} = \left(-R\left(q_a^b\right)p_{b/a}, \left(q_a^b\right)^{-1}\right),
\]
such that
\[
T_a^b \circ (T_a^b)^{-1} = (T_a^b)^{-1} \circ T_a^b = \begin{pmatrix} 0 & 1 \\ \mathbf{0} & \mathbf{1} \end{pmatrix}.
\]

Given linear and angular changes \((u \in \mathbb{R}^3, \theta \in \mathbb{R}^3) \sim \mathbb{R}^6\), the exponential map for a rigid transform is given by
\[
\exp : \mathbb{R}^6 \rightarrow SE(3)
\]
\[
\exp \left( \begin{bmatrix} u \\ \theta \end{bmatrix} \right) \triangleq (V(\theta)u, \exp \theta),
\]
where
\[
V(\theta) = I + \frac{1 - \cos \|\theta\|}{\|\theta\|^2} [\theta] \times + \frac{\|\theta\| - \sin \|\theta\|}{\|\theta\|^3} [\theta] \times [\theta] \times.
\] (8.11)

The corresponding logarithmic map is defined by
\[
\log : SE(3) \rightarrow \mathbb{R}^6
\]
\[
\log (T(p,q)) = (V^{-1}(\theta)p, \log(q)),
\]
where

\[
V^{-1}(\theta) = I - \frac{1}{2} [\theta]_\times + \frac{1}{\|\theta\|^2} \left(1 - \frac{\|\theta\| \sin \|\theta\|}{2 (1 - \cos \|\theta\|)}\right) [\theta] \times [\theta]_\times .
\]  

(8.12)

To prevent numerical instability when \(\|\theta\|\) is small, we employ Taylor series expansions in (8.11) and (8.12). The necessary expansions are given by

\[
\frac{1 - \cos \|\theta\|}{\|\theta\|^2} = \frac{1}{2} - \frac{\|\theta\|^2}{24} + \frac{\|\theta\|^4}{720} - \cdots
\]

\[
\frac{\|\theta\| - \sin \|\theta\|}{\|\theta\|^3} = \frac{1}{6} - \frac{\|\theta\|^2}{120} + \frac{\|\theta\|^4}{5040} - \cdots
\]

\[
\frac{1}{\|\theta\|^2} \left(1 - \frac{\|\theta\| \sin \|\theta\|}{2 (1 - \cos \|\theta\|)}\right) = \frac{1}{12} + \frac{\|\theta\|^2}{720} + \frac{\|\theta\|^4}{30240} + \cdots,
\]

and if we drop all nonlinear terms, we can approximate (8.11) and (8.12) as

\[
V(\theta) \approx I + \frac{1}{2} [\theta]_\times
\]  

(8.13)

\[
V^{-1}(\theta) \approx I - \frac{1}{2} [\theta]_\times.
\]  

(8.14)

8.2.3 ⊕ and ⊖ Operators

Hertzberg et al. [40] describe a new syntax that simplifies working with Lie groups in a filtering and optimization framework by introducing the ⊕ and ⊖ operators. This syntax allows us to work with elements of Lie groups in a notation similar to that of vectors and is used throughout this chapter. The ⊕ and ⊖ operators are defined differently for different groups. For \(\mathbb{R}^n\), they are simply defined as the typical addition and subtraction operations. For unit quaternions representing
attitude, these operators are defined by

\[
\boxdot: \mathbb{S}^3 \times \mathbb{R}^3 \to \mathbb{S}^3
\]

\[
\mathbf{q} \boxdot \theta \triangleq \mathbf{q} \otimes \exp(\theta)
\]  \hspace{1cm} (8.15)

\[
\boxminus: \mathbb{S}^3 \times \mathbb{S}^3 \to \mathbb{R}^3
\]

\[
\mathbf{q} \boxminus \mathbf{p} \triangleq \log(\mathbf{p}^{-1} \otimes \mathbf{q})
\]  \hspace{1cm} (8.16)

Using this notation, (8.9) and (8.10) can be written as

\[
\mathbf{q}^b_i (t + \Delta t) = \mathbf{q}^b_i (t) \boxdot \bm{\omega}^b_{b/I} \Delta t
\]

\[
\bm{\omega}^b_{b/I} \Delta t = \mathbf{q}^b_i (t + \Delta t) \boxminus \mathbf{q}^b_i (t).
\]

While this syntax is convenient, it is important to note that the dimensionality of \(\bm{\omega}^b_{b/I} \Delta t\) and \(\mathbf{q}^b_i\) are different in this case. The quaternion is not a vector and has four parameters, while \(\bm{\omega}^b_{b/I} \Delta t\) has only three parameters and exists in the vector space of \(\mathbb{R}^3\).

We can similarly define the \(\boxdot/\boxminus\) operators for rigid transformations as

\[
\boxdot: \text{SE}(3) \times \mathbb{R}^6 \to \text{SE}(3)
\]

\[
\mathbf{T} \boxdot \delta \triangleq \mathbf{T} \circ \exp(\delta)
\]  \hspace{1cm} (8.17)

\[
\boxminus: \text{SE}(3) \times \text{SE}(3) \to \mathbb{R}^6
\]

\[
\mathbf{T}_2 \boxminus \mathbf{T}_1 \triangleq \log(T_1^{-1} \circ T_2).
\]

As an example, consider the integration of rigid body transforms. With body linear rate \(\mathbf{v}^b_{b/I}\), body angular rate \(\bm{\omega}^b_{b/I}\), and inertial to body transform \(\mathbf{T}^b_i\), we have

\[
\mathbf{T}^b_i (t + \Delta t) = \mathbf{T}^b_i (t) \boxdot \left(\mathbf{v}^b_{b/I} \Delta t, \bm{\omega}^b_{b/I} \Delta t\right)
\]
Aligning with the theory of manifold operators $\oplus/\ominus$, which we have just described, we also represent derivatives and error in the Lie Algebra, such that $\dot{q}, \ddot{q} \in \mathbb{R}^3$ and $\dot{T}, \ddot{T} \in \mathbb{R}^6$. Let’s clarify this notion by looking at the quaternion. It is clear from (8.16) that $\ddot{q} \in \mathbb{R}^3$, but the derivative is less obvious. Derivatives are fundamentally defined by limits, so let’s take a limit to verify that $\dot{q} \in \mathbb{R}^3$. Let $q_{t+\Delta t} = q_t \oplus \omega \Delta t$ for a constant angular rate $\omega \in \mathbb{R}^3$ and a time step $\Delta t$. Taking a limit with appropriate addition and subtraction operators given in (8.15) and (8.16), we have

$$\dot{q} = \lim_{\Delta t \to 0} \frac{1}{\Delta t} (q_{t+\Delta t} \ominus q_t)$$

$$= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left( \log (q_t^{-1} \otimes q_{t+\Delta t}) \right)$$

$$= \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left( \log (q_t^{-1} \otimes q_t \otimes \exp (\omega \Delta t)) \right)$$

$$= \omega.$$

### 8.3 Image-Based EKF

#### 8.3.1 Optical Flow

In contrast to similar work in VIO [19], we parameterize the bearing to each landmark in our state as the pixel position in the image frame. The kinematics of pixel position in this configuration are commonly known as optical flow. While these kinematics are well known, we offer a brief derivation of optical flow to enhance understanding of the filter, including forthcoming definitions.

Suppose we observe a landmark in the camera frame $p_{l/c}^c = \begin{bmatrix} x & y & z \end{bmatrix}^\top$, which has a pixel position $v = \begin{bmatrix} u & v \end{bmatrix}^\top$. Assuming a pin-hole camera model, the relationship between landmark 3D
position and pixel coordinates is given by

\[
\frac{u - u_0}{f_x} = \frac{x}{z}, \quad \frac{v - v_0}{f_y} = \frac{y}{z},
\]

where \((u_0, v_0)\) and \((f_x, f_y)\) are horizontal and vertical components of the image center and camera focal length in units of pixels, respectively. Rearranging terms yields

\[
u = f_x \frac{x}{z} + u_0 \quad v = f_y \frac{y}{z} + v_0,
\] (8.18)

and differentiating (8.18) with respect to time gives

\[
\dot{u} = f_x \left( \frac{\dot{x}}{z} - \frac{x}{z^2} \dot{z} \right), \quad \dot{v} = f_y \left( \frac{\dot{y}}{z} - \frac{y}{z^2} \dot{z} \right),
\] (8.19)

which expresses optical flow in terms of relative motion between the camera and the landmark.

The position of a static landmark relative to the camera in the camera frame \(p_{c/l} = R^c_I (p_{l/l} - p_{c/l})\) has the time derivative

\[
\dot{p}_{c/l} = \dot{R}_I^c \left( p_{l/l} - p_{c/l} \right) + R_I^c \left( \dot{p}_{l/l} - \dot{p}_{c/l} \right)
\]

\[
= - \left( \omega_{c/l}^c \right)^\times R_I^c p_{c/l} - R_I^c \dot{p}_{c/l}
\]

\[
= - \left( \omega_{c/l}^c \right)^\times p_{c/l} - v_{c/l},
\] (8.20)

where \(\omega_{c/l}^c = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\top\) is the camera’s angular rate expressed in the camera frame and \(v_{c/l} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^\top\) is the camera’s linear velocity expressed in the camera frame. In component form, (8.20) becomes

\[
\dot{x} = \omega_y z - \omega_z v_x
\] (8.21)

\[
\dot{y} = \omega_z x - \omega_x v_y
\] (8.22)
\[ \dot{z} = \omega_x x - \omega_y y - v_z. \] (8.23)

Substituting (8.21), (8.22), and (8.23) into (8.19) yields

\[ \dot{u} = f_x \left( \frac{\omega_x y - \omega_y z - v_x}{z} - \frac{x}{z^2} (\omega_x x - \omega_y y - v_z) \right) \] (8.24)
\[ \dot{v} = f_y \left( \frac{\omega_x z - \omega_z x - v_y}{z} - \frac{y}{z^2} (\omega_x x - \omega_y y - v_z) \right). \] (8.25)

With \( \varepsilon_x = u - u_0, \varepsilon_y = v - v_0, \) solving for \( x \) and \( y \) in (8.18) then substituting these into (8.24) and (8.25) gives

\[ \dot{u} = \frac{f_x}{f_y} \varepsilon_y \omega_x - \left( f_x + \frac{\varepsilon_x^2}{f_x} \right) \omega_y + \frac{\varepsilon_x \varepsilon_y}{f_y} \omega_x + \frac{1}{z} \varepsilon_x v_z - \frac{f_x}{z} v_x \]
\[ \dot{v} = -\frac{f_y}{f_x} \varepsilon_x \omega_z - \frac{\varepsilon_x \varepsilon_y}{f_x} \omega_y + \left( f_y + \frac{\varepsilon_y^2}{f_y} \right) \omega_x + \frac{1}{z} \varepsilon_y v_z - \frac{f_y}{z} v_y. \]

The optical flow equations may then be written as

\[ \dot{v} = \Omega (v) \omega_c^{c/i} + \frac{1}{z} W (v) v_c^{c/i}, \]

where

\[ \Omega (v) = \begin{bmatrix} \frac{\varepsilon_x \varepsilon_y}{f_y} & -\left( f_x + \frac{\varepsilon_x^2}{f_x} \right) & \frac{f_x}{f_y} \varepsilon_y \\ f_x + \frac{\varepsilon_x^2}{f_x} & -\frac{\varepsilon_x \varepsilon_y}{f_y} & -\frac{f_x}{f_y} \varepsilon_x \end{bmatrix}, \] (8.26)
\[ W (v) = \begin{bmatrix} -f_x & 0 & \varepsilon_x \\ 0 & -f_y & \varepsilon_y \end{bmatrix}. \] (8.27)
This gives us the kinematics of pixel positions of landmarks, but we also need the kinematics of inverse $z$-depth, which is the inverse of the $z$ component of $p^c_{1/c}$. With $\rho = \frac{1}{z}$, we have

$$\dot{\rho} = -\frac{\dot{z}}{z^2},$$

and using (8.23), we obtain

$$\dot{\rho} = -\frac{\omega_x x - \omega_y y - v_z}{z^2}.$$

Using (8.18) results in

$$\dot{\rho} = \rho \left( \frac{\varepsilon_y}{f_y} \omega_x - \frac{\varepsilon_x}{f_x} \omega_y \right) + \rho^2 v_z.$$

Writing this in vector form gives

$$\dot{\rho} = \rho M(\nu) \omega^c_{c/l} + \rho^2 \varepsilon^\top_3 v^c_{c/l},$$

where

$$M(\nu) = \begin{bmatrix} \frac{\varepsilon_y}{f_y} & -\frac{\varepsilon_x}{f_x} & 0 \end{bmatrix}. \tag{8.28}$$

Summarizing the kinematics of optical flow and inverse $z$-depth, we have

$$\dot{\nu} = \Omega(\nu) \omega^c_{c/l} + \frac{1}{z} V(\nu) v^c_{c/l}$$

$$\dot{\rho} = \rho M(\nu) \omega^c_{c/l} + \rho^2 \varepsilon^\top_3 v^c_{c/l}.$$

### 8.3.2 EKF Propagation

In this section, we define the composition of our EKF but leave many of the details in Appendix 8.7.2, due to the lengthiness of the derivation. Let the state $\mathbf{x} \in \mathbb{R}^6 \times \mathcal{S}^3 \times \mathbb{R}^7 \times \mathbb{R}^{3N}$ and
error state $\tilde{x} \in \mathbb{R}^6 \times \mathbb{R}^3 \times \mathbb{R}^7 \times \mathbb{R}^{3N}$ be defined by

$$
x = \begin{bmatrix}
p^l_{b/I} \\
v^l_{b/I} \\
q^b_l \\
b_a \\
b_\omega \\
\mu \\
v_1 \\
\rho_1 \\
\vdots \\
v_N \\
\rho_N
\end{bmatrix}, \quad \hat{x} = x \hat{=} \hat{x} = \begin{bmatrix}
p^l_{b/I} - \hat{p}^l_{b/I} \\
v^l_{b/I} - \hat{v}^l_{b/I} \\
q^b_l \hat{=} \hat{q}^b_l \\
b_a - \hat{b}_a \\
b_\omega - \hat{b}_\omega \\
\mu - \hat{\mu} \\
v_1 - \hat{v}_1 \\
\rho_1 - \hat{\rho}_1 \\
\vdots \\
v_N - \hat{v}_N \\
\rho_N - \hat{\rho}_N
\end{bmatrix},
$$

where $N$ is the number of landmarks, $p^l_{b/I}$ is multirotor position, $v^h_{b/I}$ is velocity, $q^b_I$ is attitude, and $\mu$ is a linear drag term [31]. We note that $x$ and $\hat{x}$ have one less element that $\tilde{x}$ because $q^b_I \hat{=} \hat{q}^b_I \in \mathbb{R}^3$.

The corresponding covariance matrix $P$ is then defined in terms of the error state as

$$
P = E \left[ \tilde{x} \tilde{x}^\top \right] \in \mathbb{R}^{(16 + 3N) \times (16 + 3N)}.
$$
The state estimate evolves in time according to

\[
\dot{\hat{x}} = f(\hat{x}, u) =
\begin{bmatrix}
\dot{\hat{l}} \\
\dot{\hat{p}}_{b/I} \\
\dot{\hat{q}}_{b/I} \\
\dot{\hat{b}}_a \\
\dot{\hat{b}}_\omega \\
\dot{\hat{\mu}} \\
\dot{\hat{\nu}}_1 \\
\dot{\hat{\rho}}_1 \\
\vdots \\
\dot{\hat{\nu}}_N \\
\dot{\hat{\rho}}_N
\end{bmatrix},
\quad u = \begin{bmatrix}
\bar{a}^u_{u/I} - \hat{b}_a \\
\bar{\omega}^u_{u/I} - \hat{b}_\omega
\end{bmatrix},
\]

where \( \bar{a}^u_{u/I} \) and \( \bar{\omega}^u_{u/I} \) are the measured acceleration and angular rate of the IMU, respectively, and the elements of \( f(\hat{x}, u) \) are given in (8.53)-(8.59) of Appendix 8.7.2.

As new IMU measurements are received, we propagate the state and covariance up to the most recent IMU measurement according to trapezoidal integration. Define \( k \) as the current time step, \( \Delta t \) as the time between IMU measurements, and we propagate the state by

\[
\hat{x}_k = \hat{x}_{k-1} \oplus \frac{1}{2} \left[ f(\hat{x}_{k-1}, u_{k-1}) + \right. \\
\left. f(\hat{x}_k, u_k) \right] \Delta t.
\]
Covariance propagation requires a knowledge of error-state kinematics, which is defined as
\[
\dot{x} = \frac{d}{dt} \tilde{f}(\tilde{x}, \hat{x}, \eta),
\]
where
\[
\tilde{f}(\tilde{x}, \hat{x}, \eta) =
\begin{bmatrix}
\dot{p}^I_{b/I} \\
\dot{v}^I_{b/I} \\
\dot{q}^b \\
\dot{b}_a \\
\dot{b}_\omega \\
\dot{\nu}_1 \\
\vdots \\
\dot{\nu}_N \\
\dot{\rho}_N
\end{bmatrix}, \quad \eta = \begin{bmatrix} \eta_a \\ \eta_\omega \end{bmatrix},
\]
where the elements of \(\tilde{f}(\tilde{x}, \hat{x}, \eta)\) are given in (8.82), (8.83), (8.84), (8.85), (8.86), (8.87), (8.90), and (8.92) of Appendix 8.7.2.

The covariance evolves continuously according to
\[
\dot{P} = FP + PF^\top + GQ_u G^\top + Q_x, 
\]
where the Jacobians \(F\) and \(G\) are given in Appendix 8.7.3, \(Q_u\) is the IMU covariance, and \(Q_x\) is the process noise covariance. Propagating \(P\) according to (8.29) does not guarantee that \(P\) remains positive definite. Therefore, we propagate covariance in the discrete form according to
\[
P_{k+1} = F_k P_k F_k^\top + G_k Q_u G_k^\top + Q_x \Delta t^2, 
\]
\[ F_k \approx I + F \Delta t + \frac{1}{2} FF^T \Delta t^2 \]
\[ G_k \approx \left( I + F \frac{\Delta t}{2} \right) G \Delta t. \]

We have defined how to propagate the state and covariance of our EKF using IMU measurements. In the next section, we define measurement models and observation matrices for camera pixel and depth measurements needed to apply discrete updates to the filter.

### 8.3.3 Measurement Models

#### 8.3.3.1 Landmark Direction Update

In our filter, the measurement model and observation matrix for the landmark direction associated with a static landmark is straightforward because the pixel positions in our state are measured directly by the camera. When only pixel measurements are available, the true and estimated measurement models are given by

\[
\begin{align*}
    h_{\text{cam}}(x) &= \begin{bmatrix} \nu_1 + \eta_{\text{pix}} \\ \vdots \\ \nu_N + \eta_{\text{pix}} \end{bmatrix}, \\
    \hat{h}_{\text{cam}}(\hat{x}) &= \begin{bmatrix} \hat{\nu}_1 \\ \vdots \\ \hat{\nu}_N \end{bmatrix},
\end{align*}
\]

where \( \eta_{\text{pix}} \) is Gaussian noise. The error model is then given by

\[
\tilde{h}_{\text{cam}}(\tilde{x}) = h_{\text{cam}}(x) - \hat{h}_{\text{cam}}(\hat{x}) = \begin{bmatrix} \tilde{\nu}_1 + \eta_{\text{pix}} \\ \vdots \\ \tilde{\nu}_N + \eta_{\text{pix}} \end{bmatrix}
\]
and the resulting observation matrix is given as

$$H_{cam} = \frac{\partial \hat{h}_{cam}}{\partial \hat{\bar{x}}} = \begin{bmatrix}
0_{2 \times 16} & I & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0
\end{bmatrix}
= \begin{bmatrix}
0_{2 \times 16} & I & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0
0_{2 \times 16} & 0 & 0 & I & 0 & \cdots & 0 & 0 & 0 & 0
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots 
0_{2 \times 16} & 0 & 0 & 0 & 0 & \cdots & I & 0 & 0 & 0 \\
0_{2 \times 16} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & I & 0
\end{bmatrix}.$$
where

\[ h_d(x) = \left\| \frac{1}{\rho_s} K^{-1} \begin{bmatrix} v_s + \eta_{pix} \\ 1 \end{bmatrix} \right\| + \eta_d \]

\[ \hat{h}_d(\hat{x}) = \left\| \frac{1}{\hat{\rho}_s} K^{-1} \begin{bmatrix} \hat{v}_s \\ 1 \end{bmatrix} \right\|. \]

The error model is then given by

\[ \tilde{h}_{cam}(\tilde{x}, \hat{x}) = h_{cam}(x) - \hat{h}_{cam}(\hat{x}) \]

\[ = \begin{bmatrix} \hat{v}_1 + \eta_{pix} \\ \hat{h}_d(\tilde{x}, \hat{x}) \\ \vdots \\ \hat{v}_N + \eta_{pix} \\ \tilde{h}_d(\tilde{x}, \hat{x}) \end{bmatrix}, \]

where

\[ \tilde{h}_d(\tilde{x}, \hat{x}) = \frac{1}{\hat{\rho}_s} \left( \frac{\hat{p}_{l/c}^c}{\| \hat{p}_{l/c}^c \|} \right)^\top K^{-1} \begin{bmatrix} \hat{v}_s + \eta_{pix} \\ 1 \end{bmatrix} + \eta_d. \]

The resulting observation matrix is given as

\[ H_{cam} = \frac{\partial \tilde{h}_{cam}}{\partial \tilde{x}} \]
where

\[
H_{\rho}^\ast = \begin{bmatrix}
0_{3 \times 16} & I & 0 & 0 & 0 & \cdots & 0 & 0 \\
0_{3 \times 16} & H_{\rho_1}^\ast H_{\rho_1}^\ast & 0 & 0 & \cdots & 0 & 0 \\
0_{3 \times 16} & 0 & 0 & I & 0 & \cdots & 0 & 0 \\
0_{3 \times 16} & 0 & 0 & H_{\rho_2}^\ast H_{\rho_2}^\ast & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0_{3 \times 16} & 0 & 0 & 0 & 0 & \cdots & 0 & I \\
0_{3 \times 16} & 0 & 0 & 0 & 0 & \cdots & H_{\rho_N}^\ast H_{\rho_N}^\ast \\
\end{bmatrix},
\]

Now that we have the necessary measurement models and their Jacobians, we can apply discrete updates to the state as camera images are received. This chapter employs a non-traditional update, called the partial update, which is described in the following section.

### 8.3.4 EKF Update and Keyframe Reset

#### 8.3.4.1 Partial Update

A common difficulty in Kalman filtering is the estimation of nuisance states that may only be partially observable depending on the vehicle’s motion. In our filter, these nuisance states include the IMU biases and the linear drag coefficient. As noted in [30], estimating these terms in the traditional manner can result in poor performance and even divergence, but ignoring them or considering them as known constants may produce an overconfident estimate. Using a partial update
allows us to limit the effect of measurement updates to specific states with a scalar gain $\gamma_i$, while remaining consistent by correctly accounting for the reduced update in covariance estimation. While this method loses optimality guarantees in a linear Kalman-filtering framework, it has been shown to speed up convergence of these nuisance states by limiting the effect of linearization errors when applied to the nonlinear IMU-camera extrinsics estimation problem [30].

The partial update is derived as follows. First define $\lambda_i = 1 - \gamma_i$ and

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_N \end{bmatrix},$$

which contains the tuning parameters for $N$ states. The values in this vector range from zero to one with ones indicating a full update to those particular states and are empirically selected, typically remaining constant during runtime. When a non-IMU measurement $z_s$ is received, we compute the Kalman gain by

$$K_s = PH_s^\top (R_s + H_s PH_s^\top)^{-1},$$

where $R_s$ is the measurement covariance. The state and covariance updates are then given by

$$\hat{x} = \hat{x} \oplus (\lambda \odot K_s (z_s \oplus h_s (\hat{x})))$$

$$P = P + \Lambda \odot \left( (I - K_s H_s) P (I - K_s H_s)^\top + K_s R_s K_s^\top - P \right),$$

where $\odot$ is the Hadamard product, and

$$1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top$$

$$\Lambda = 1\lambda^\top + \lambda 1^\top - \lambda\lambda^\top.$$
8.3.4.2 Keyframe Reset

This filter also employs the recently introduced keyframe reset [29, 53]. Attempting to estimate global position and heading in a Kalman filter with only IMU and pixel measurements results in filter inconsistency because these states are not observable. This may be countered with keyframe resets. A keyframe reset simply resets the global position and heading states to zero, then updates the covariance matrix appropriately. This process results in the establishment of many nodes, which can then be incorporated into a pose graph structure to perform loop closures as part of a global optimization routine. We must point out that our setup is slightly different than [53] and [29] because there is no altimeter measurement available, so altitude is also unobservable and must be reset.

In this chapter, the state and covariance resets are given by

\[
\hat{x}^+ = \begin{bmatrix} 0 \\ \hat{v}_{b/I} \\ (\hat{q}_I^b)^+ \\ \hat{b}_a \\ \hat{b}_o \\ \hat{\mu} \\ \hat{v}_1 \\ \hat{\rho}_1 \\ \vdots \\ \hat{v}_N \\ \hat{\rho}_N \end{bmatrix}, \quad P^+ = NPN^\top,
\]

where using (8.3), (8.4), (8.5), and (8.6), we also have

\[
(\hat{q}_I^b)^+ = q(\hat{\phi}, \hat{\theta}, 0)
\]
\[
N_q = \begin{bmatrix}
1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\
0 & \cos \hat{\phi}^2 & -\cos \hat{\phi} \sin \hat{\phi} \\
0 & -\cos \hat{\phi} \sin \hat{\phi} & \sin \hat{\phi}^2
\end{bmatrix}
\]

\[
N = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & I & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & N_q & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & I & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1
\end{bmatrix}
\]

Our new VI-EKF that we have just described uses several transformations, like \(q^b_u\) and \(T^c_u\), which must be known prior to implementing the filter. The following section described the offline, nonlinear optimization we use to obtain these transformations.

### 8.4 Calibration

Good performance of a VI-EKF requires an accurate estimate of several extrinsic parameters defining the spatial and temporal relationship between the camera and IMU. These parameters include the transform from the camera frame to the IMU frame and the time offset between the camera and the IMU clock. Because we are also employing the linear drag model described in [31] to improve accuracy, we must also have an accurate initial estimate of the drag coefficient and the IMU to body rotation because drag is applied in the body frame. We estimate these parameters...
by post-processing recorded sensor information collected during a multirotor flight in a motion
capture room, where the multirotor is equipped with an RGB camera and an IMU. To properly
compute these parameters, we must also estimate all of the parameters listed in Table 8.1, where
\( x_k = \begin{bmatrix} T^u_k & v^u_k \end{bmatrix} \) is the pose and velocity of the IMU at the \( k^{th} \) node. Note that we assume constant
IMU bias throughout the calibration window, which is a reasonable assumption once the running
IMU temperature has reached equilibrium.

Table 8.1: List of states and calibration parameters.

\[
\begin{array}{l}
\begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix} : \text{IMU state history for } N \text{ time steps} \\
\begin{bmatrix} \rho_1 & \cdots & \rho_M \end{bmatrix} : \text{initial inverse depth of } M \text{ features} \\
T^m_u : \text{IMU to motion capture transform} \\
q^b_u : \text{IMU to body rotation} \\
T^c_u : \text{IMU to camera transform} \\
b_\omega : \text{rate gyro bias} \\
b_a : \text{accelerometer bias} \\
t_m : \text{motion capture-IMU time offset} \\
t_c : \text{camera-IMU time offset} \\
\mu : \text{drag coefficient}
\end{array}
\]

The calibration tool is setup as a graph optimization problem and uses the Ceres-Solver [85]
to perform a nonlinear, local optimization. This means that we need good initial guesses for the
parameters being optimized, which are given in Table 8.1. Fortunately, using a motion capture sys
ystem yields good initial values for parameters like pose, velocity, and landmarks, and the remaining
parameters must be initialized through physical measurements or estimation from manufacturer
data.

To improve robustness to initial error, Our calibrator performs three stages of optimization.
The first stage computes \( \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix} \), \( b_a \), \( b_\omega \), \( T^m_u \), and \( t_m \). The second stage computes \( T^c_u \),
\( \begin{bmatrix} \rho_1 & \cdots & \rho_M \end{bmatrix} \), and \( t_c \). The third stage computes \( q^b_u \) and \( \mu \). Performing the optimization in separate
stages reduces the number of degrees of freedom in the optimizer, improving its robustness to
error, while still achieving an accurate solution. In the following sections, we detail each stage of
optimization, including derivations of the residual errors or factors that the optimizer minimizes.
8.4.1 Stage 1: Pose and Motion Capture Calibration

The initial phase of calibration involves the IMU and motion capture measurements, where we add nodes for each of the motion capture measurements to our graph and integrate IMU measurements in between nodes. We adopt an IMU preintegration strategy similar to [75], which is thoroughly outlined using our notation with some subtle changes in Appendix 8.7.4.

8.4.1.1 IMU and Motion Capture Factors

The IMU factor is defined by moving all terms to the left side in (8.94)-(8.95) and replacing the IMU preintegration terms with their best estimates from (8.96)-(8.97). This yields the factor

\[
\mathbf{r}_{uk} = \left[ R^I I (\mathbf{p}^I_{uk+1/I} - \mathbf{p}^I_{uk/I} - \mathbf{v}^I_{uk/I} \Delta t_k - \frac{1}{2} \mathbf{ge}_3 \Gamma^I_k) - \mathbf{\hat{\alpha}}^u_{uk+1/uk} \right] \\
R^I I (\mathbf{v}^I_{uk+1/I} - \mathbf{v}^I_{uk/I} - \mathbf{ge}_3 \Delta t_k) - \mathbf{\hat{\beta}}^u_{uk+1/uk} \\
\left( (q^I_{u})^{-1} \otimes \mathbf{q}^I_{uk+1} \right) \mathbf{\hat{\gamma}}^u_{uk+1}.
\]

The motion capture residual requires the time derivative of pose \( \dot{T}^m_I = f_T(t, T^m_I) \), where because the motion capture sample rate is mostly constant, we approximate \( \dot{T}^m_I \) with a central differencing scheme on the manifold by

\[
f_T(t, \mathbf{T}^m_I) \approx \frac{\mathbf{T}^m_{I+1} - \mathbf{T}^m_{I-1}}{u_{I+1/1} - u_{I-1}}.
\]

The motion capture residual is then defined by

\[
\mathbf{r}_{mk} = T^m_I (\delta t_m) \mathbf{\hat{T}}^m_I (\mathbf{x}), \quad (8.30)
\]
where $\delta t_m$ is the change in IMU to motion capture time offset and

$$
T_I^{mk}(\delta t_m) = T_I^{mk} \boxplus f_T(t, \tilde{T}_I^m) \delta t_m
$$

$$
\hat{T}_I^{mk}(\hat{x}) = \hat{T}_u^x \circ \hat{T}_u^m.
$$

The time derivative of pose is used to adjust the potential time offset between IMU and motion capture in the same way [34] adjusts the camera-IMU time offset. Because $\delta t_m$ is based on a linearization of the motion capture measurements, it may accrue significant error if the actual time offset is large. Therefore, we run the optimization for a given time offset $t_m$, apply the resulting estimate of $\delta t_m$ to $t_m$, adjust the original motion capture measurements according to the updated $t_m$, and recompute all the IMU integration intervals. We then repeat this process until $\delta t_m$ is sufficiently small.

### 8.4.1.2 Rigid Transform Parameterization

Taking derivatives of (8.30) without accounting for the fact that $T_I^m \in SE(3)$ is inefficient because the rigid transform is represented by a translation and a quaternion but only has six degrees of freedom (DOF). Fortunately, the Ceres Solver has the ability to define local parameterizations, so that we can tell the optimizer how to take these derivatives with respect to the underlying DOF. First, stack the change in translation $\delta_t \in \mathbb{R}^3$ and change in orientation $\delta_q \in \mathbb{R}^3$ into the vector $
abla = \begin{bmatrix} \delta_t^\top & \delta_q^\top \end{bmatrix}^\top \in \mathbb{R}^6$. Then for an arbitrary vector $x$, we make use of the chain rule

$$
\frac{\partial x}{\partial \delta} = \frac{\partial x}{\partial T} \frac{\partial T}{\partial \delta} \boxplus \frac{\partial \delta}{\partial \delta},
$$

where automatic differentiation provides us with $\frac{\partial x}{\partial T}$ and $\frac{\partial T}{\partial \delta} \boxplus \frac{\partial \delta}{\partial \delta}$ is identity. Using (8.7), (8.13), and (8.14), we obtain

$$
\frac{\partial T \boxplus \delta}{\partial \delta} = \frac{\partial T \circ \exp(\delta)}{\partial \delta}.
$$
\[
\frac{\partial}{\partial \delta} \left[ \begin{array}{c}
p + R(q)^T \left( I + \frac{1}{2} [\delta_q \times] \right) \delta_t \\
q \otimes \exp(\delta_q) \end{array} \right]
\approx \frac{\partial}{\partial \delta} \left[ \begin{array}{c}
p + R(q)^T \left( I + \frac{1}{2} [\delta_q \times] \right) \delta_t \\
q \otimes \left[ \begin{array}{c}1 \\ \frac{\delta_y}{2} \end{array} \right] \end{array} \right]
\approx \left[ \begin{array}{c}
R(q)^T \left( I + \frac{1}{2} [\delta_q \times] \right) \\
0_{4 \times 3}
\end{array} \right] - \frac{1}{2} R(q)^T [\delta_q]_x \left[ \begin{array}{c}
-q_x \\
-q_y \\
-q_z \\
q_w \\
-q_x \\
q_y \\
q_z \\
-q_w
\end{array} \right].
\] (8.31)

Implementation using the Ceres Solver can now be accomplished by defining (8.31) in code, which yields the best performance, or by allowing Ceres to automatically differentiate (8.17) with respect to \( \delta \in \mathbb{R}^6 \).

8.4.2 Stage 2 - Camera Extrinsic Calibration

The second stage of calibration estimates both the transform from IMU to camera and the initial inverse depth of each feature tracked throughout the sequence. Having already computed the IMU pose at each node, we hold them constant during this and the third stage of optimization.

8.4.2.1 Camera Factor

Similar to the motion capture residual, the camera residual requires pixel velocities to adjust \( t_c \). Because the camera does not directly measure pixel velocities \( \dot{\nu}_v = f_v(t, \nu_v) \), we compute this numerically. The sample rate of the camera may not be constant, mostly due to auto-exposure, and so we do not use a traditional differencing scheme. Numerical differentiation of data with unequal
steps can be accomplished by differentiating the Lagrange Interpolating Polynomial

\[ f_n(x) = \sum_{i=0}^{n} \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j} f(x_i), \]

to obtain

\[ \frac{\partial f_n}{\partial x} = \sum_{i=0}^{n} \left[ \prod_{j=0, j \neq i}^{n} \frac{1}{x_i - x_j} \right] \left[ \sum_{j=0, j \neq k}^{n} \prod_{k=0, k \neq i, j}^{n} x - x_k \right] f(x_i). \] (8.32)

Using (8.32) to obtain pixel velocity, \( x \) is replaced with an image time of interest, \( x_{i,j,k} \) are replaced with sampled image times, and \( f(x_i) \) is replaced with pixel position at the \( i^{th} \) sample time. Because this fits polynomials to data points, there isn’t any lag involved in the calculation, but error arises from the approximation of the pixel motion as a polynomial. We also note that we achieved good results with second order polynomials, where \( n = 2 \) in (8.32).

Now, we derive the relationship between each image. Let reference frame \( c_0 \) represent the camera pose when a landmark is first detected, and also let reference frame \( c_k \) represent a subsequent detection of the same landmark. We can relate these two camera poses by

\[ p_{c_k}^{c_0} = R_{c_k}^{c_0} \left( p_{l/c_0} + R_I^{c_0} (p_{c_0}^{I} - p_{c_k}^{I}) \right), \]

and writing this in terms of the calibration parameters, we obtain

\[ p_{l/c_k}^{c_0} = R_k^{I} \left( R_d^{I} \left( R_{d_0}^{I} \left( \frac{1}{\rho_0} e_{l/c_0} + p_{c/u}^{u} \right) + p_{u_0}^{I} - p_{u_k}^{I} \right) - p_{c/u}^{u} \right). \]
We have \( R_u^c \) and \( p_{c/u}^u \) as the rotation and translation from IMU to camera contained in \( T_u^c \), while the initial landmark direction is computed by

\[
\mathbf{e}_{l/c_0}^{c_0} = K^{-1} \begin{bmatrix} v_0(\delta t_c) \\ 1 \end{bmatrix},
\]

where

\[
v_\ast(\delta t_c) = \bar{v}_\ast + f_v(t, v_\ast) \delta t_c.
\]

The residual error is now written as

\[
r_c = v_k(\delta t_c) - \pi \left( \frac{p_{l/c_k}}{||p_{l/c_k}||} \right),
\]

where \( \pi(\mathbf{e}_\ast) \) projects the unit vector \( \mathbf{e}_\ast \) into image coordinates. Using time stamps and pixel measurements in (8.32) gives us an approximate pixel velocity, which is acceptable for camera-IMU time offset calculation by following the same optimize-adjust-repeat process described in Section 8.4.1. We also employ a Huber loss function with this residual, to reduce the effect of outliers in the image feature tracking and depth estimation.

### 8.4.2.2 Inverse-Depth Initialization

Since we began Stage 2 with accurate estimates of body poses, we use that information to initialize the inverse depths of all landmarks in the sequence. Suppose we have the inertial positions of a landmark \( p_{l/I}^l \) and a camera \( p_{c/l}^l \). We also have the camera’s intrinsic matrix \( K \) and an undistorted pixel measurement of the landmark \( v \). The relationship between the pixel coordinates
and 3D landmark is given by
\[
\begin{bmatrix}
v \\
1
\end{bmatrix} = \frac{1}{z}KR_j^c \left( p_{l/I}^f - p_{c/I}^f \right),
\]
where \( R_j^c \) is the rotation from the inertial frame to the camera frame, \( K \) is the camera intrinsic matrix, and \( z \) is the landmark distance along the camera’s optical axis. This can be rewritten in terms of a projection matrix by
\[
\begin{bmatrix}
v \\
1
\end{bmatrix} = \frac{1}{z}P \mathbf{x},
\]
(8.33)

where
\[
P = KR_j^c \begin{bmatrix} I & -p_{c/I}^f \end{bmatrix} \quad \quad \mathbf{x} = \begin{bmatrix} p_{l/I}^f \\ 1 \end{bmatrix}.
\]

Writing (8.33) in terms of the rows of \( P \) yields
\[
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} p_0^\top \\ p_1^\top \\ p_2^\top \end{bmatrix} \mathbf{x},
\]
and can eliminate a dependence on \( z \) with the relation
\[
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \frac{1}{z} \begin{bmatrix} p_0^\top \\ p_1^\top \\ p_2^\top \end{bmatrix} \mathbf{x} = 0,
\]
which yields the equations
\[
\left(v \mathbf{p}_2^\top - \mathbf{p}_1^\top\right) \mathbf{x} = 0
\]
\[
\begin{align*}
(p_0^\top - up_2^\top)x &= 0 \\
(up_1^\top - vp_0^\top)x &= 0.
\end{align*}
\]

Only the first two equations are needed for further calculations. If we can stack equations for \(N\) observations of the same landmark then we can form the homogeneous equation

\[
Ax = 0, \quad (8.34)
\]

where

\[
A = \begin{bmatrix}
v_1 (p_2^\top)_1 - (p_1^\top)_1 \\
(p_0^\top)_1 - u_1 (p_2^\top)_1 \\
\vdots \\
v_N (p_2^\top)_N - (p_1^\top)_N \\
(p_0^\top)_N - u_N (p_2^\top)_N
\end{bmatrix}.
\]

Using the singular value decomposition (SVD), we obtain \(A = USV^\top\). The solution to (8.34) is the normalized right singular vector associated with the smallest singular value. In other words, we take the right most column of \(V\) and normalize by the fourth element. This gives the triangulated landmark position in homogeneous coordinates.

**8.4.3 Stage 3 - Drag Constant and Body Frame Calibration**

The third stage of optimization calculates the drag coefficient and IMU to body rotation. Initially, we employed two factors: one based on the \(x/y\) accelerometer measurements [31] while the other was based on drag-enabled kinematics. We found that the \(x/y\) accelerometer measurements were noisy enough to degrade the final estimate of the drag coefficient, even using a Huber loss function to reduce outlier effects, while relying just on the drag-enabled kinematics resulted in
much more consistent results. This is likely because using a kinematic drag factor is less dependent on specific trajectories to make drag observable.

8.4.3.1 Kinematic Drag Factor

The kinematics of inertial position, velocity, and attitude are defined in terms of drag by

\[
\begin{align*}
\dot{p}_{u_k/I}^I &= f_p \left( q_{ut}^b, \mu \right) \\
\dot{v}_{u_k/I}^I &= f_v \left( q_{ut}^b, \mu \right) \\
\dot{q}_I^{u_k} &= f_q \left( q_{ut}^b, \mu \right),
\end{align*}
\]

where

\[
\begin{align*}
f_p \left( q_{ut}^b, \mu \right) &= v_{u_k/I}^I \\
f_v \left( q_{ut}^b, \mu \right) &= (R_{I}^{u_k})^\top \left( R_{I}^{b} \right)^\top \left( e_3 e_3^T R_{I}^{b} \left( a_{u_k/I}^b - b_d \right) - \mu \left( I - e_3 e_3^T \right) R_{I}^{b} R_{I}^{u_k/I} v^I_{u_k/I} \right) + g e_3 \\
f_q \left( q_{ut}^b, \mu \right) &= \omega_{u_k/I}^b - b_\omega.
\end{align*}
\]

Using these newly defined drag kinematics, the kinematic drag factor is then defined by

\[
\begin{align*}
\mathbf{r}_\mu &= \left[ \begin{array}{c}
p'_{u_k+1/I} - \left( p_{u_k/I}^I + \int_{t_k}^{t_{k+1}} f_p \left( q_{ut}^b, \mu \right) dt \right) \\
v'_{u_k+1/I} - \left( v_{u_k/I}^I + \int_{t_k}^{t_{k+1}} f_v \left( q_{ut}^b, \mu \right) dt \right) \\
q_{I}^{u_k+1} \oplus \left( q_{I}^{u_k} \oplus \int_{t_k}^{t_{k+1}} f_q \left( q_{ut}^b, \mu \right) dt \right)
\end{array} \right].
\end{align*}
\] (8.35)

This factor is similar to the IMU factor given in 8.4.1, however, we now have a fixed IMU position, velocity, and attitude at each node, and therefore, we only need to estimate the the drag coefficient and IMU to body rotation.
8.4.3.2 Body to IMU Rotation Parameterization

We employ a special parameterization for $q^b_u$ to improve the optimization of (8.35). Since we use a single drag coefficient to account for all drag forces, the yaw component of $q^b_u$ is not observable and thus, $q^b_u$ only contains two DOF. Attempting to estimate the yaw component is incorrect and could result in unexpected behavior when rotating IMU measurements into the body frame. Therefore, we employ the $\boxplus/\boxminus$ operators defined in Appendix 8.7.1. This prevents optimizer from modifying the yaw component of $q^b_u$ and reduces optimization time in this stage.

As in Section 8.4.1, we must also define the local parameterization but this time for a 2-DOF quaternion. This means that we represent a unit vector $e \in S^2$ with a rotation $q \in S^3$ but define the $\boxplus/\boxminus$ operators such that only two DOF are used. For $q \in S^3$, $\delta \in \mathbb{R}^3$, and using the $\boxplus$ operator defined in Appendix 8.7.1, the mapping to this local parameterization is given by

$$
\frac{\partial q \boxplus \delta}{\partial \delta} = \frac{1}{2} \begin{bmatrix}
-q_y & -q_z \\
q_z & q_y \\
-q_x & q_z \\
q_y & q_x 
\end{bmatrix}.
$$

(8.36)

As in the rigid transform parameterization, this is implemented using the Ceres Solver by defining (8.36) in code for best performance, or by allowing Ceres to automatically differentiate (8.39) with respect to $\delta \in \mathbb{R}^2$.

8.5 Results

To investigate the performance and robustness of the proposed calibrator and estimator, we built a multirotor simulator in C++ that contains aircraft dynamics, sensor models including time delays, and an environment with randomly placed, static landmarks on the walls and floor of a simulated room. These landmarks are projected onto the camera image with some noise, which
are then used in the calibrator and estimator. It would be more realistic to render and build camera images from a graphical scene, but we chose not to do this because that makes that filter depend on the feature tracker. Our results show the capabilities of the calibrator and estimator under ideal feature tracking circumstances. We also tune the noise parameters to mimic what we have seen in other hardware experiments. Motion capture only has a small amount of noise, while IMU has a large amount of noise, as well as biases with a slowly varying random walk. Feature measurements from the camera have a small amount of noise. For the EKF experiments, we only track up to ten features in the filter.

The multirotor aircraft is set to fly a set number of waypoints in all experiments. These waypoints consist of a position and heading only several meters apart to mimic flying in a moderately sized motion capture room. We have chosen waypoints to fly back and forth in a plus configuration with some variance in heading to provide sufficient excitation and make landmark depths and IMU biases more observable.

8.5.1 Calibration Performance

Figures 8.1 and 8.2 show the results of 100 Monte Carlo simulations, where the calibrator’s parameters were initialized with significant errors designed to mimic a poor guess by the user. Rotation errors were on the order of 30°, accelerometer bias up to 1.0 m/s², gyro bias up to 0.5 rad/s, the drag coefficient up to 0.2, motion capture time offset up to 100 milliseconds, and camera time offset up to 50 milliseconds. In spite of these errors, we see from Figures 8.1 and 8.2 that the calibrator was able to find the correct solution in every simulation. The error does not converge to exactly zero because of the noise in all of the sensor measurements and random walk in the IMU biases. We also notice a tight grouping in the final errors because the sensor data was collected once, while the calibrator was initialized to random values and ran many times on the same data. This shows that the user’s best guess of the transforms and time offsets prior to calibration are
most likely going to be good enough, assuming the sensor data excited all of the dynamic modes necessary for observability.

Figure 8.1: Motion capture and camera parameter calibration error results of 100 Monte Carlo simulations with significant initial errors.

8.5.2 EKF Performance

The visual-inertial problem solved using only an RGB camera and IMU is inherently not robust to initial error, primarily error in velocity, because the camera can only observe depth up to a scale factor and the IMU only provides integrated data. Therefore, a poor initial velocity estimate drives the landmark depth estimates to the wrong values and this is difficult, if not impossible to recover from in the filter alone. However, when depth information is available, the velocity state becomes much more observable, so most engineers will opt for something like a stereo camera pair or a depth camera. If neither of these fit the solution, then a good initialization algorithm can be used.
Figure 8.2: Other parameter calibration error results of 100 Monte Carlo simulations with significant initial errors.
Figures 8.3 and 8.4 demonstrate robustness of the filter to significant initial error when feature depth measurements are available. We set the depth measurements to have zero-mean, Gaussian noise with a standard deviation of one meter. The initial velocity errors were up to 5 m/s, roll and pitch up to 45°, and drag coefficient up to 0.3. Feature estimates were initialized at the first measurement of pixel position and depth. Error in velocity, roll/pitch, IMU biases, and the drag coefficient converge to acceptable values in Figures 8.3 and 8.4, while position and heading quickly plateau. Again, these do not reduce to small values because position and heading are unobservable. Most of the error in position and heading accrues as error in the other states is reduced because position and heading are computed from integration of other states. Only the accelerometer $z$ bias is plotted here because using the drag term removes accelerometer $x/y$ biases from propagation.

Figure 8.5 shows filter convergence for 100 simulations, where the initial pose and velocity error was small and depth measurements were unavailable. There is no need to show pose and velocity errors because they start small. We see that the IMU biases and feature inverse $z$-depths converge close to their true values in all simulations. This is expected because the velocity and attitude begin close to truth and have small uncertainties, which allows the filter to push better corrections to the unknown states.

To compare the linearity of our pixel-based VI-EKF with the quaternion-based VI-EKF [19], we derived the error state kinematics of the quaternion representation with inverse depth in Appendices 8.7.1 and 8.7.2 and then designed another simulation to compare these two approaches. We initialized both the pixel-based and quaternion-based error states at the true initial state error, and propagated both forward in time, feeding them the same sensor data. This process was repeated 1000 times for varying initial state errors, saving the difference between the true state error and the propagated error for each state formulation. Figure 8.6 is the result of these simulations, showing the average difference over time for both state formulations. We see that both the pixel-based and quaternion-based formulations are comparable in linearity as they begin at truth and deviate over
Figure 8.3: EKF error in position, velocity, and attitude over 100 Monte Carlo simulations with significant initial error, using depth measurements.
Figure 8.4: EKF error in IMU biases and drag coefficient over 100 Monte Carlo simulations with significant initial error, using depth measurements.
Figure 8.5: EKF error in IMU biases and inverse z-depth estimates over 100 Monte Carlo simulations with small initial error in pose and velocity, without depth measurements.
time on very similar trajectories. At the final time, we also note that the pixel-based version has less error. While this difference is not significant, we must remember that using image coordinates removes the need for an iterated EKF update, providing significant computational savings with slightly less linearization error than the quaternion-based formulation.

![Graph showing average error over time](image)

**Figure 8.6:** Average error in error-state propagation over 1000 Monte Carlo runs for pixel-based and quaternion-based landmark direction representations.

### 8.6 Conclusions

This chapter proposed a novel VI-EKF based directly on image coordinates measured by the camera and a new calibration scheme to estimate transformations needed for an accurate estimator using a linear drag coefficient. We have shown robustness in the calibrator to poor initial guesses by numerical simulation. The VI-EKF was also shown to be robust to initial error when using either depth information or a good initialization of pose and velocity. Finally, we compared the linearity of our new image-based VI-EKF to the state-of-the-art quaternion-based VI-EKF and found that our parameterization removes the need for an expensive iterated update with slightly less linearization error.
8.7 Appendix

8.7.1 2-DOF Quaternions

As in [19], landmark directions may be parameterized in the camera frame as rotations $q_l^c \in S^3$, which describe the rotation from the camera optical axis to the unit vector directed at the feature. The unit vector directed at feature $l$ with respect to the camera frame $c$ is then defined by

$$e^c_{l/c} = (R^l_c)^\top e_3 \in S^2 \subset \mathbb{R}^3,$$  \hspace{1cm} (8.37)

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

The difference between two landmark unit vectors $e_l \sqotimes e_j$ can be described using axis-angle representation, where the direction of the axis of rotation is orthogonal to both of the unit vectors, and its length is scaled by the magnitude of rotation. There are actually only two degrees of freedom in this parameterization because rotation about either feature vector does not change unit vector direction. To remove the redundant degree of freedom, we ensure that the axis of shortest rotation is always in the plane normal to $e_l$ and define a projection matrix

$$P_l = (R^l_c)^\top \begin{bmatrix} e_1 & e_2 \end{bmatrix} \in \mathbb{R}^{3 \times 2},$$  \hspace{1cm} (8.38)

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

We must then define the $\boxplus/\boxminus$ operators associated with feature direction vectors as

$$\boxplus : S^3 \times \mathbb{R}^2 \rightarrow S^3$$

$$q^l_c \boxplus \delta \triangleq \exp(P_l \delta) \otimes q^l_c$$  \hspace{1cm} (8.39)
\[ \mathbb{S}^3 \times \mathbb{S}^3 \rightarrow \mathbb{R}^2 \]
\[ \mathbf{q}_L^l \mathbf{q}_L^b \triangleq \theta P_I^\top \mathbf{s}, \]

where the axis \( \mathbf{s} \) and angle \( \theta \) between the two feature direction vectors are given by

\[
\theta = \cos^{-1} \left( \mathbf{e}_i^\top \mathbf{e}_j \right) \\
\mathbf{s} = \frac{\mathbf{e}_i \times \mathbf{e}_j}{\|\mathbf{e}_i \times \mathbf{e}_j\|}.
\]

With only two degrees of freedom, and with all feature vectors referencing the camera optical axis, there are an infinite number of unit quaternions which can be used to represent the same unit vector. The difference between these rotations is some angle of rotation about the bearing vector itself. This is removed by the projection operation and can therefore be neglected.

### 8.7.2 Error State Kinematics

In this appendix, we derive the error state kinematics associated with both image-based and quaternion-based visual-inertial filters. Both filters have some common states, so we begin with those along with the kinematics of each landmark representation. We have the true kinematics of position, velocity, attitude, IMU biases, drag coefficient, pixel position, inverse \( z \)-depth, landmark quaternion, and inverse landmark distance given by

\[
\dot{\mathbf{p}}_{b/I}^I = \left( R_b^b \right)^\top \mathbf{v}_{b/I}^b 
\]
\[
\ddot{\mathbf{v}}_{b/I}^b = \mathbf{e}_3 \mathbf{e}_3^\top R_b^b \mathbf{u}_{u/I}^u + g R_b^b \mathbf{e}_3 - \\
\mu \left( I - \mathbf{e}_3 \mathbf{e}_3^\top \right) \mathbf{v}_{b/I}^b - \left[ R_b^b \omega_{u/I}^u \right] \times \mathbf{v}_{b/I}^b 
\]
\[
\dot{\mathbf{q}}_l^b = R_b^b \omega_{u/I}^u 
\]
\[
\dot{\mathbf{b}}_{a} = 0 
\]
\[
\dot{\mathbf{b}}_{\omega} = 0
\]
\[ \dot{\mu} = 0 \quad (8.44) \]
\[ \dot{\nu} = \Omega(\nu) \omega^c_{c/I} + \rho W(\nu) \dot{v}^c_{c/I} \quad (8.45) \]
\[ \dot{\rho} = \rho M(\nu) \omega^c_{c/I} + \rho^2 \epsilon^3 \mathbf{v}^c_{c/I} \quad (8.46) \]
\[ \dot{q}^c_c = -R^T \left( \omega^c_{c/I} + \rho \left[ \epsilon^I_{l/c} \right] \times \mathbf{v}^c_{c/I} \right) \quad (8.47) \]
\[ \dot{\dot{\rho}} = \rho^2 \left( \epsilon^I_{l/c} \right)^T \mathbf{v}^c_{c/I} \quad (8.48) \]

where the unbiased/denoised IMU measurements and camera velocities are

\[ \mathbf{a}^u_{u/I} = \mathbf{a}^u_{u/I} - \mathbf{b}_a - \eta_a \quad (8.49) \]
\[ \mathbf{\omega}^u_{u/I} = \mathbf{\omega}^u_{u/I} - \mathbf{b}_\omega - \eta_\omega \quad (8.50) \]
\[ \mathbf{v}^c_{c/I} = R^c_u \left( \left( R^b_u \right)^T \mathbf{v}^b_{b/I} + \left[ \mathbf{\omega}^u_{u/I} \times \mathbf{p}^u_{u/I} \right] \right) \quad (8.51) \]
\[ \mathbf{\omega}^c_{c/I} = R^c_u \mathbf{\omega}^u_{u/I} \quad (8.52) \]

Estimated kinematics are then given by

\[ \ddot{p}^I_{b/I} = \left( \hat{R}^b_l \right)^T \dot{v}^b_{b/I} \quad (8.53) \]
\[ \dot{\mathbf{v}}^b_{b/I} = \mathbf{e}_3 \mathbf{e}_3^T R^b_u \hat{a}^u_{u/I} + g \hat{R}^b_l \mathbf{e}_3 - \hat{\mu} \left( I - \mathbf{e}_3 \mathbf{e}_3^T \right) \dot{\mathbf{v}}^b_{b/I} - \left[ R^b_u \hat{\omega}^u_{u/I} \right] \times \dot{\mathbf{v}}^b_{b/I} \quad (8.54) \]
\[ \dot{q}^b_l = R^b_u \hat{a}^u_{u/I} \quad (8.55) \]
\[ \dot{\mathbf{b}}_a = 0 \quad (8.56) \]
\[ \dot{\mathbf{b}}_\omega = 0 \quad (8.57) \]
\[ \hat{\mu} = 0 \quad (8.58) \]
\[ \dot{\nu} = \Omega(\dot{\nu}) \hat{\omega}^c_{c/I} + \hat{\rho} W(\dot{\nu}) \hat{\dot{v}}^c_{c/I} \quad (8.59) \]
\[ \dot{\rho} = \hat{\rho} M(\dot{\nu}) \hat{\omega}^c_{c/I} + \hat{\rho}^2 \epsilon^3 \mathbf{v}^c_{c/I} \]
\[
\dot{\mathbf{q}}_c = -\mathbf{P}_l^T \left( \hat{\omega}_{c/I} + \hat{\varrho} \left[ \hat{\mathbf{e}}_{l/c} \times \hat{\mathbf{v}}_{c/I} \right] \right),
\]

(8.60)

\[
\dot{\mathbf{v}} = \hat{\mathbf{v}}_c^c \mathbf{v}_{c/I},
\]

(8.61)

where

\[
\hat{\mathbf{a}}_{u/I} = \mathbf{a}_{u/I} - \hat{\mathbf{a}}_u,
\]

(8.62)

\[
\hat{\omega}_{u/I} = \omega_{u/I} - \hat{\omega},
\]

(8.63)

\[
\hat{\mathbf{v}}_{c/I} = R_u^c \left( \left( R_u^b \right)^T \hat{\mathbf{v}}_b^b + \left[ \hat{\mathbf{a}}_{u/I} \right] \times \mathbf{p}_{u/b} \right),
\]

\[
\hat{\omega}_{c/I} = R_u^c \hat{\omega}_{u/I}.
\]

Now, we define the error states

\[
\hat{\mathbf{p}}_{b/I} = \mathbf{p}_{b/I} - \hat{\mathbf{p}}_{b/I},
\]

(8.64)

\[
\hat{\mathbf{v}}_{b/I} = \mathbf{v}_{b/I} - \hat{\mathbf{v}}_{b/I},
\]

(8.65)

\[
\hat{\mathbf{q}}_l = q_l^l \mathbf{q}_{l/I} \in \mathbb{R}^3
\]

(8.66)

\[
\hat{b}_u = \mathbf{b}_u - \hat{\mathbf{b}}_u,
\]

(8.67)

\[
\hat{\mathbf{b}}_\omega = \mathbf{b}_\omega - \hat{\mathbf{b}}_\omega
\]

\[
\hat{\mu} = \mu - \hat{\mu}
\]

(8.68)

\[
\hat{\nu} = \mathbf{v} - \hat{\mathbf{v}}
\]

(8.69)

\[
\hat{\rho} = \rho - \hat{\rho}
\]

(8.70)

\[
\hat{\mathbf{q}}_c^l = q_c^l \mathbf{q}_{c/I} \in \mathbb{R}^2
\]

(8.71)

\[
\hat{\mathbf{b}} = \mathbf{b} - \hat{\mathbf{b}}.
\]

(8.72)

We will differentiate these error states with respect to time to obtain the error state kinematics. We will also need an approximation to rotation error. Using (8.15) to add attitude error to an estimate
yields true attitude as

\[ \mathbf{q}_I^b = \hat{\mathbf{q}}_I^b \otimes \exp \left( \hat{\mathbf{q}}_I^b \right). \]

Noting that rotation matrices concatenate in an order opposite to Hamilton quaternions, using (8.2), and (8.8), we have

\[
R_I^b = R \left( \exp \left( \hat{\mathbf{q}}_I^b \right) \right) \hat{R}_I^b \\
\approx R \left( \begin{bmatrix} 1 \\ \frac{1}{2} \hat{\mathbf{q}}_I^b \end{bmatrix} \right) \hat{R}_I^b \\
= \left( I - \left[ \hat{\mathbf{q}}_I^b \right]_\times + \frac{1}{2} \hat{\mathbf{q}}_I^b \left( \hat{\mathbf{q}}_I^b \right)^\top \right) \hat{R}_I^b.
\]

Neglecting nonlinear terms, we have the true attitude is given in terms of error and estimate by

\[
R_I^b \approx \left( I - \left[ \hat{\mathbf{q}}_I^b \right]_\times \right) \hat{R}_I^b. \tag{8.73}
\]

We must compute a similar approximation for landmark quaternions. Using (8.39) to add landmark direction error to an estimate yields the true landmark direction quaternion as

\[ \mathbf{q}_c^l = \exp \left( \hat{\mathbf{p}}_c^l \hat{\mathbf{q}}_c^l \right) \otimes \hat{\mathbf{q}}_c^l. \]

Again, noting that rotation matrices concatenate in an order opposite to Hamilton quaternions, using (8.2), and (8.8), we have

\[
R_c^l = \hat{R}_c^l R \left( \exp \left( \hat{\mathbf{p}}_c^l \hat{\mathbf{q}}_c^l \right) \right) \\
\approx \hat{R}_c^l R \left( \begin{bmatrix} 1 \\ \frac{1}{2} \hat{\mathbf{p}}_c^l \hat{\mathbf{q}}_c^l \end{bmatrix} \right) \\
= \hat{R}_c^l \left( I - \left[ \hat{\mathbf{p}}_c^l \hat{\mathbf{q}}_c^l \right]_\times + \frac{1}{2} \hat{\mathbf{p}}_c^l \left( \hat{\mathbf{p}}_c^l \hat{\mathbf{q}}_c^l \right)^\top \right).
\]
Neglecting nonlinear terms, we have the true landmark direction is given in terms of error and estimate by

\[ R^l_c \approx \hat{R}_c^l \left( I - [\hat{\mathbf{p}}_c^{l^\top}]_\times \right). \]  

(8.74)

All but the quaternion states in (8.64)-(8.72) are written in terms of error and estimate as

\[ p^{i^l/l}_b = \hat{p}^{i^l/l}_b + \tilde{p}^{i^l/l}_b \]  

(8.75)

\[ v^{b/l}_b = \hat{v}^{b/l}_b + \tilde{v}^{b/l}_b \]  

(8.76)

\[ b_a = \hat{b}_a + \tilde{b}_a \]  

(8.77)

\[ b_\omega = \hat{b}_\omega + \tilde{b}_\omega \]  

(8.78)

\[ \mu = \hat{\mu} + \tilde{\mu} \]  

(8.79)

\[ \nu = \hat{\nu} + \tilde{\nu} \]  

(8.80)

\[ \rho = \hat{\rho} + \tilde{\rho} \]  

(8.81)

### 8.7.2.1 Position

Differentiating (8.64) with respect to time and substituting (8.40) and (8.53) yields

\[ \dot{\hat{p}}^{i^l/l}_b = \hat{p}^{i^l/l}_b - \hat{\dot{p}}^{i^l/l}_b \]

\[ = \left( R_f^b \right)^\top v^{b/l}_b - \left( \hat{R}_f^b \right)^\top \hat{\dot{v}}^{b/l}_b. \]

Now substituting (8.73), (8.75), and dropping nonlinear error terms while simplifying gives

\[ \dot{\hat{p}}^{i^l/l}_b = \left( \hat{R}_f^b \right)^\top \left( [\hat{q}_f^b]_\times \hat{v}^{b/l}_b + \tilde{v}^{b/l}_b \right). \]
Lastly, using (8.1) results in
\[
\dot{\tilde{p}}_{b/1} = \left( \hat{R}_f^b \right)^\top \left( \hat{v}_{b/1}^b - \left[ \hat{v}_{b/1}^b \times \dot{q}_f^b \right] \times \hat{d}_f^b \right). \tag{8.82}
\]

### 8.7.2.2 Velocity

Differentiating (8.65) with respect to time and substituting (8.41) and (8.54) yields
\[
\dot{\tilde{v}}_{b/1}^b = \dot{v}_{b/1}^b - \dot{\hat{v}}_{b/1}^b
\]
\[
= e_3^e_3^\top R_u^b \tilde{a}_u/ - I_3 - \mu \left( I - e_3^e_3^\top \right) v_{b/1}^b -
\]
\[
\left[ R_u^b \omega_u/ \right] \times v_{b/1}^b - \left( e_3^e_3^\top R_u^b \tilde{a}_u/ - I_3 \right) \dot{v}_{b/1}^b -
\]
\[
\hat{\mu} \left( I - e_3^e_3^\top \right) \dot{\tilde{v}}_{b/1}^b - \left[ R_u^b \dot{\tilde{a}}_u/ \right] \times \dot{v}_{b/1}^b).
\]

Inserting (8.49), (8.50), (8.62), and (8.63) gives
\[
\dot{\tilde{v}}_{b/1}^b = e_3^e_3^\top R_u^b \left( \tilde{a}_u/ - b_a - \eta_a \right) + g R_f^b e_3 -
\]
\[
\mu \left( I - e_3^e_3^\top \right) v_{b/1}^b - \left[ R_u^b \left( \tilde{\omega}_u/ - b_\omega - \eta_\omega \right) \right] \times v_{b/1}^b -
\]
\[
\left( e_3^e_3^\top R_u^b \left( \tilde{a}_u/ - \tilde{b}_a \right) + g R_f^b e_3 - \hat{\mu} \left( I - e_3^e_3^\top \right) \dot{v}_{b/1}^b -
\]
\[
\left[ R_u^b \left( \tilde{\omega}_u/ - \tilde{b}_\omega \right) \right] \times \dot{v}_{b/1}^b).
\]

Now substituting (8.75), (8.73), (8.76), (8.77), (8.78), dropping nonlinear error terms while simplifying, and using (8.1) to pull out error terms results in
\[
\dot{\tilde{v}}_{b/1}^b = - \left( \hat{\mu} \left( I - e_3^e_3^\top \right) + \left[ R_u^b \left( \tilde{\omega}_u/ - \tilde{b}_\omega \right) \right] \times \right) \tilde{v}_{b/1}^b +
\]
\[
g \left[ R_f^b e_3 \right] \times \hat{q}_f^b - e_3^e_3^\top R_u^b \left( \tilde{b}_a + \eta_a \right) -
\]
\[
\hat{\mu} \left( I - e_3^e_3^\top \right) \dot{\tilde{v}}_{b/1}^b - \left[ \tilde{v}_{b/1}^b \right] \times R_u^b \left( \tilde{b}_\omega + \eta_\omega \right). \tag{8.83}
\]
### 8.7.2.3 Attitude

Rather than trying to differentiate through the $\square$ operation in (8.66), we note that the derivative of attitude lies in its tangent space. Also, noting that the true derivative and estimated derivative lie on different tangent spaces, we simply move one into the other and perform regular subtraction because the tangent space is a vector space. Therefore, the derivative of (8.66) may be written as

$$\dot{q}_i^b = q_i^b - R_i^b (\dot{R}_i^b)^\top \dot{q}_i^b,$$

and substituting (8.42) and (8.55) yields

$$\dot{q}_i^b = R_i^b \omega_{u/I} - R_i^b (\dot{R}_i^b)^\top R_i^b \dot{\omega}_{u/I}.$$

Inserting (8.50) and (8.63) gives

$$\dot{q}_i^b = R_i^b \left( \tilde{\omega}^u_{u/I} - \mathbf{b}_\omega - \eta_\omega \right) - R_i^b (\dot{R}_i^b)^\top R_i^b \left( \tilde{\omega}^u_{u/I} - \mathbf{b}_\omega \right),$$

and substituting (8.73), (8.77), dropping nonlinear error terms while simplifying, and using (8.1) yields results in

$$\dot{q}_i^b = - \left[ R_i^b \left( \tilde{\omega}^u_{u/I} - \mathbf{b}_\omega \right) \right] \times \dot{q}_i^b - R_i^b \left( \mathbf{b}_\omega + \eta_\omega \right). \quad (8.84)$$

### 8.7.2.4 IMU Biases and Drag Term

Differentiating (8.67)-(8.68) and substituting in (8.43)-(8.44) and (8.56)-(8.57) gives

$$\dot{b}_a = 0 \quad (8.85)$$

$$\dot{b}_\omega = 0 \quad (8.86)$$

$$\dot{\mu} = 0. \quad (8.87)$$
8.7.2.5 Pixel Position

Error kinematics of pixel position require two more identities. Inserting (8.79) into (8.26) and (8.27), we obtain

\[ \Omega (\hat{\nu} + \tilde{\nu}) = \Omega (\hat{\nu}) + \hat{\Omega} (\hat{\nu}, \tilde{\nu}) \]  
(8.88)

\[ V (\hat{\nu} + \tilde{\nu}) = W (\hat{\nu}) + \tilde{\nu} e_3^\top \]  
(8.89)

\[ \hat{\Omega} (\hat{\nu}, \tilde{\nu}) = \begin{bmatrix} \frac{\hat{\epsilon}_y \hat{u}_x + \hat{\epsilon}_x \hat{u}_y}{f_y} & -\frac{2 \hat{\epsilon}_y \hat{u}_x}{f_x} & \frac{f_x \tilde{\nu}_y}{f_y} \\ \frac{2 \hat{\epsilon}_y \tilde{\nu}_x}{f_y} & -\frac{\hat{\epsilon}_y \hat{u}_x + \hat{\epsilon}_x \tilde{\nu}_y}{f_x} & -\frac{f_x \tilde{\nu}_y}{f_x} \end{bmatrix}. \]

Now, differentiating (8.69) with respect to time and substituting (8.45) and (8.58) yields

\[ \dot{\tilde{\nu}} = \dot{\nu} - \hat{\dot{\nu}} = \Omega (\nu) \omega_{c/I} + \rho W (\nu) \nu_{c/I} - \\
\phantom{= \Omega (\nu) \omega_{c/I} + \rho W (\nu) \nu_{c/I}} \left( \Omega (\tilde{\nu}) \omega_{c/I} + \rho W (\tilde{\nu}) \nu_{c/I} \right). \]

Inserting (8.51) and (8.52), followed by (8.50) gives

\[ \dot{\nu} = \Omega (\nu) R_u^c \left( \omega_{u/I} - b_\omega - \eta_\omega \right) + \\
\rho W (\nu) R_u^c \left( R_u^b \nu_{b/I} + [\tilde{\omega}_{u/I}^{\nu} - b_\omega - \eta_\omega] \times p_{c/u}^{u} \right) - \\
\left( \Omega (\tilde{\nu}) \omega_{c/I} + \rho W (\tilde{\nu}) \nu_{c/I} \right). \]

Now substituting (8.75), (8.77), (8.79), (8.80), followed by using (8.88), (8.89), dropping nonlinear error terms while simplifying, and using (8.1) results in

\[ \dot{\nu} = \rho W (\tilde{\nu}) R_u^c \left( R_u^b \nu_{b/I} + \rho W (\tilde{\nu}) p_{c/u}^{u} \right) - \\
\Omega (\tilde{\nu}) R_u^c \left( b_\omega + \eta_\omega \right) + \tilde{\Omega} (\tilde{\nu}, \tilde{\nu}) \omega_{c/I}+. \]
\[ \dot{\rho} e_3^c \hat{v}^c_{c/l} \hat{\nu} + W(\hat{v}) \hat{v}^c_{c/l} \dot{\rho}. \]

It is convenient to pull \( \hat{\nu} \) out of \( \hat{\Omega}(\hat{v}, \hat{\nu}) \hat{\omega}^c_{c/l} \) when computing Jacobians. Thus, we finally have

\[ \dot{\rho} e_3^c \hat{v}^c_{c/l} \hat{\nu} + W(\hat{v}) \hat{v}^c_{c/l} \dot{\rho}. \] (8.90)

where

\[ \Omega_2(\hat{v}, \hat{\omega}^c_{c/l}) = \begin{bmatrix} \frac{\hat{\epsilon}_y}{f_y} \omega_x - 2 \frac{\hat{\epsilon}_x}{f_x} \omega_y & \frac{1}{f_y} (\hat{\epsilon}_x \omega_y + f_x \omega_z) \\ -\frac{1}{f_x} (\hat{\epsilon}_z \omega_x + f_y \omega_z) & 2 \frac{\hat{\epsilon}_x}{f_x} \omega_x - \frac{\hat{\epsilon}_z}{f_z} \omega_y \end{bmatrix}. \]

**8.7.2.6 Inverse z-depth**

Error kinematics of inverse z-depth require another identity. Inserting (8.79) into (8.28), we obtain

\[ M(\hat{v} + \tilde{\nu}) = M(\hat{v}) + \tilde{\nu}^T M_2, \] (8.91)

where

\[ M_2 = \begin{bmatrix} 0 & -\frac{1}{f_x} & 0 \\ \frac{1}{f_y} & 0 & 0 \end{bmatrix}. \]

Now, differentiating (8.70) with respect to time and substituting (8.46) and (8.59) yields

\[ \dot{\rho} = \dot{\rho} - \dot{\rho} \]

\[ = \rho M(\nu) \omega^c_{c/l} + \rho^2 e_3^T \nu^c_{c/l} - \left( \dot{\rho} M(\hat{v}) \hat{\omega}^c_{c/l} + \dot{\rho}^2 e_3^T \hat{\nu}^c_{c/l} \right), \]
Inserting (8.51) and (8.52) followed by (8.50) gives

\[
\dot{\hat{\rho}} = \rho M (\hat{\nu}) R_u^e \left( \hat{\omega}_{u/l}^u - b_\omega - \eta_\omega \right) + \\
\rho^2 e_3^\top R_u^c \left( R_u^b \right)^\top \hat{\nu}_{b/l}^b + \left[ \hat{\omega}_{u/l}^u - b_\omega - \eta_\omega \right] \times p_{c/u}^u - \\
\left( \dot{\hat{\rho}} M (\hat{\nu}) \hat{\omega}_{c/l}^c + \rho^2 e_3^\top \hat{\nu}_{c/l}^c \right).
\]

Now substituting (8.75), (8.77), (8.79), (8.80), followed by using (8.91), dropping nonlinear error terms while simplifying, and using (8.1) results in

\[
\dot{\hat{\rho}} = \rho^2 e_3^\top R_u^c \left( R_u^b \right)^\top \hat{\nu}_{b/l}^b + \\
\dot{\hat{\rho}} \left( \rho e_3^\top R_u^c \left[ p_{c/u}^u \right] \times M (\hat{\nu}) R_u^c \right) (b_\omega + \eta_\omega) + \\
\dot{\hat{\rho}} \left( M_2 \hat{\omega}_{c/l}^c \right)^\top \hat{\nu} + \left( M (\hat{\nu}) \hat{\omega}_{c/l}^c + 2 \rho e_3^\top \hat{\nu}_{c/l}^c \right) \hat{\rho}.
\]  

(8.92)

### 8.7.2.7 2-DOF Quaternion

Rather than trying to differentiate through the \(\Box\) operation in (8.71), we note that this derivative lies in its tangent space. Also, noting that the true derivative and estimated derivative lie on different tangent spaces, we simply move one into the other and perform regular subtraction because the tangent space is a vector space. Therefore, the derivative of (8.71) may be written as

\[
\dot{q}^l_* = \dot{q}^l_* - P_l^\top \dot{\hat{\rho}} q^l_*
\]

(8.93)

and expanding \(P_l^\top \dot{\hat{\rho}}\) with (8.38), followed by (8.74) gives

\[
P_l^\top \dot{\hat{\rho}} = \left( \left( R_c^l \right)^\top \left[ e_1 \ e_2 \right] \right)^\top \hat{\rho} l \\
= \left[ e_1 \ e_2 \right]^\top R_c^l \hat{\rho} l
\]

188
\[
= \hat{P}_l^T \left( I - \left[ \hat{P}_l \hat{q}_c^l \right] \times \right) \hat{P}_l \\
= I - \hat{P}_l^T \left[ \hat{P}_l \hat{q}_c^l \right] \times \hat{P}_l.
\]

It can be shown that \( \hat{P}_l^T \left[ \hat{P}_l \right] \times \hat{P}_l = 0_{2 \times 2} \) for an arbitrary vector \( v \in \mathbb{R}^2 \) by expanding terms and using the identity \( R[v] \times R^T = [Rv]_x \). Thus, \( P_l^T \hat{P}_l = I \) and (8.93) becomes

\[
\dot{q}_c^l = q_c^l - \hat{q}_c^l.
\]

Substituting (8.47) and (8.60) yields

\[
\dot{q}_c^l = -P_l^T \left( \omega_{c/l}^c + \hat{\rho} \left[ \hat{e}_{c/l}^c \times \hat{v}_{c/l}^c \right] \right) + \\
\hat{P}_l^T \left( \hat{\omega}_{c/l}^c + \hat{\rho} \left[ \hat{e}_{c/l}^c \times \hat{v}_{c/l}^c \right] \right),
\]

then inserting (8.51), (8.52), (8.37), and (8.38), followed by (8.50) gives

\[
\dot{q}_c^l = - \left( R_c^l \right)^T \left[ \begin{array}{c} e_1 \\ e_2 \end{array} \right] \left( R_u^e \left( \bar{\omega}_{u/l}^u - b_{\omega} - \eta_{\omega} \right) \\ \hat{\rho} \left[ \left( R_c^l \right)^T e_3 \times R_u^c \left( R_u^b \right)^T v_{b/l}^b \right] \right) + \\
\hat{P}_l^T \left( \hat{\omega}_{c/l}^c + \hat{\rho} \left[ \hat{e}_{c/l}^c \times \hat{v}_{c/l}^c \right] \right) - \left( R_u^c \right)^T \left[ \bar{\omega}_{u/l}^u - b_{\omega} - \eta_{\omega} \right] \times p_{c/u}^u \\
\hat{P}_l^T \left( \hat{\omega}_{c/l}^c + \hat{\rho} \left[ \hat{e}_{c/l}^c \times \hat{v}_{c/l}^c \right] \right) - \left( R_u^c \right)^T \left[ \bar{\omega}_{u/l}^u - b_{\omega} - \eta_{\omega} \right] \times p_{c/u}^u \\
\hat{P}_l^T \left( \hat{\omega}_{c/l}^c + \hat{\rho} \left[ \hat{e}_{c/l}^c \times \hat{v}_{c/l}^c \right] \right) - \left( R_u^c \right)^T \left[ \bar{\omega}_{u/l}^u - b_{\omega} - \eta_{\omega} \right] \times p_{c/u}^u.
\]

Now, substituting (8.74), (8.75), (8.77), (8.81), dropping nonlinear error terms while simplifying, and using (8.1) results in

\[
\dot{q}_c^l = -\hat{P}_l^T \left( \hat{\rho} \left[ \hat{e}_{c/l}^c \times R_u^c \left( R_u^b \right)^T v_{b/l}^b \right] \\
\left( \hat{\rho} \left[ \hat{e}_{c/l}^c \times R_u^c \left[ p_{c/u}^u \right] \times - R_u^c \right) \left( b_{\omega} + \eta_{\omega} \right) + \right.
\]

189
\[
\begin{align*}
\left( \hat{\dot{\omega}} \left[ \hat{\dot{\mathbf{v}}}^e_{c/I} \times \left[ \hat{\mathbf{e}}^e_{c/I} \right] \right] + 
\hat{\mathbf{\omega}}^e_{c/I} + \hat{\hat{\mathbf{e}}}^e_{c/I} \times \hat{\mathbf{v}}^c_{c/I} \right) \hat{\mathbf{P}}^l \hat{\mathbf{q}}^l + \\
\left[ \hat{\hat{\mathbf{e}}}^e_{c/I} \times \hat{\mathbf{v}}^c_{c/I} \right] \hat{\mathbf{\omega}}^e_{c/I} \hat{\mathbf{\omega}}^e_{c/I} \left[ \hat{\mathbf{v}}^c_{c/I} \times \hat{\mathbf{\omega}}^e_{c/I} \right].
\end{align*}
\]

**8.7.2.8 Inverse Depth**

Differentiating (8.72) with respect to time and substituting (8.48) and (8.61) yields

\[
\hat{\hat{\nu}} = \hat{\nu} - \hat{\hat{\nu}} = \hat{\nu}^2 \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \times \\
\hat{\mathbf{v}}^e_{c/I}
\end{array} \right) \mathbf{v}^c_{c/I} - \hat{\hat{\nu}}^2 \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \times \\
\hat{\mathbf{v}}^c_{c/I}
\end{array} \right) \mathbf{v}^e_{c/I},
\]

and inserting (8.51) and (8.37), followed by (8.50) gives

\[
\hat{\hat{\nu}} = \hat{\nu}^2 \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \\
\hat{\mathbf{v}}^c_{c/I}
\end{array} \right) \mathbf{v}^c_{c/I} - \hat{\hat{\nu}}^2 \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \\
\hat{\mathbf{v}}^c_{c/I}
\end{array} \right) \mathbf{v}^e_{c/I}.
\]

Now substituting (8.75), (8.74), (8.77), (8.81), dropping nonlinear error terms while simplifying, and using (8.1) to pull out error terms results in

\[
\hat{\hat{\nu}} = \hat{\nu} \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \\
\hat{\mathbf{v}}^c_{c/I}
\end{array} \right) \mathbf{v}^c_{c/I} - \hat{\hat{\nu}}^2 \left( \begin{array}{c}
\hat{\mathbf{e}}^e_{c/I} \\
\hat{\mathbf{v}}^c_{c/I}
\end{array} \right) \mathbf{v}^e_{c/I}.
\]
8.7.3 EKF Propagation Jacobians

The Jacobian of error state kinematics with respect to error state is given by

\[
F = \frac{\partial \tilde{f}}{\partial \tilde{x}} = \begin{bmatrix}
0 & F^p_v & F^q_v & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & F^v_q & F^v_q & F^v_{b\omega} & F^v_{b\mu} & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & F^q_q & 0 & F^q_{b\omega} & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\end{bmatrix},
\]

where

\[
F^p_v = \left( \hat{R}^b_I \right)^\top \\
F^p_q = -\left( \hat{R}^b_I \right)^\top \left[ \hat{v}^b_{b/I} \right] \times \\
F^v_v = -\left( \hat{\mu} \left( I - e_3 e_3^\top \right) + R_u^b \left( \bar{\omega}_{u/I} - \bar{b}_\omega \right) \right) \times \\
F^v_q = g \left[ \hat{R}^b_I e_3 \right] \times \\
F^v_{b\omega} = -e_3 e_3^\top R_u^b \\
F^v_{b\mu} = -\left[ \hat{v}^b_{b/I} \right] R_u^b
\]

191
\[ F^v_{\mu} = - \left( I - e_3 e_3^\top \right) \hat{v}^b_{b/I} \]
\[ F^q = - \left[ R^b_u \left( \hat{\omega}^\mu_{u/I} - \hat{b}_{\omega} \right) \right] \times \]
\[ F^q_{b\omega} = -R^b_u \]
\[ F^v_{\nu} = \hat{\rho}_s W \left( \hat{\nu}_s \right) R^c_u \left( R^b_u \right)^\top \]
\[ F^v_{b\omega} = \hat{\rho}_s W \left( \hat{\nu}_s \right) R^c_u \left[ p^u_{c/u} \right] \times -\Omega \left( \hat{\nu}_s \right) R^c_u \]
\[ F^v_{\nu} = \Omega_2 \left( \hat{\nu}_s, \hat{\omega}^c_{c/I} \right) + \hat{\rho} e_3^\top \hat{v}^c_{c/I} I_{2 \times 2} \]
\[ F^v_{\rho} = W \left( \hat{\nu}_s \right) \hat{v}^c_{c/I} \]
\[ F^p = \hat{\rho}_s e_3^\top R^c_u \left( R^b_u \right)^\top \]
\[ F^p_{\nu} = \hat{\rho}_s \left( \hat{\rho}_s e_3^\top \right) \left[ p^u_{c/u} \right] \times -M \left( \hat{\nu}_s \right) R^c_u \]
\[ F^p_{\nu} = \hat{\rho}_s \left( M_2 \hat{\omega}^c_{c/I} \right)^\top \]
\[ F^p_{\rho} = M \left( \hat{\nu}_s \right) \hat{\omega}^c_{c/I} + 2 \hat{\rho}_s e_3^\top \hat{v}^c_{c/I} \]
The Jacobian of error state kinematics with respect to IMU input noise is given by

\[
G = \frac{\partial \tilde{f}}{\partial \eta} = \begin{bmatrix}
0 & 0 \\
-\mathbf{e}_3 \mathbf{e}_3^\top R_u^b & - \left[ \hat{\mathbf{v}}_b^c / I \right] \times R_u^b \\
0 & -R_u^b \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
\vdots & \vdots \\
0 & G_{\eta_\omega}^{\nu_N} \\
0 & G_{\eta_\omega}^{\rho_N}
\end{bmatrix},
\]

where

\[
G_{\eta_\omega}^{\nu} = \hat{\rho} W (\hat{\nu}) R_u^c \left[ \begin{array}{c} p_u^c \\
p_c / u \end{array} \right] - \Omega (\hat{\nu}) R_u^c \\
G_{\eta_\omega}^{\rho} = \hat{\rho} \left( \hat{\rho} \mathbf{e}_3^\top R_u^c \left[ \begin{array}{c} p_u^c \\
p_c / u \end{array} \right] - M (\hat{\nu}) R_u^c \right).
\]

### 8.7.4 IMU Preintegration

A typical MEMS IMU contains an accelerometer and a rate gyro. The accelerometer measures acceleration due to the specific forces acting on its housing, while the rate gyro measures absolute
angular rate. The IMU output is modeled by

\[
\ddot{a}_{u/I} = \frac{1}{m} R_{I}^{u} \sum_{j} f_{j}^{I} + b_{a} + \eta_{a} \\
\ddot{\omega}_{u/I} = \omega_{u/I} + b_{\omega} + \eta_{\omega}.
\]

Position, velocity, and attitude kinematics of the IMU are given by

\[
\dot{p}_{u/I}^{J} = v_{u/I}^{J} \\
\dot{v}_{u/I}^{J} = \frac{1}{m} \left( \sum_{j} f_{j}^{I} + f_{g}^{J} \right) \\
\dot{q}_{t}^{u} = \omega_{u/I}^{u}.
\]

where the change in velocity is a function of specific forces, plus the force due to gravitational acceleration. This may also be written in terms of the measurements by substituting \(\sum_{j} f_{j}^{I} = m (R_{I}^{u})^{\top} (\ddot{a}_{u/I}^{u} - b_{a} - \eta_{a})\) and \(f_{g} = m g e_{3}\) to get

\[
\dot{p}_{u/I}^{J} = v_{u/I}^{J} \\
\dot{v}_{u/I}^{J} = (R_{I}^{u})^{\top} \left( \ddot{a}_{u/I}^{u} - b_{a} - \eta_{a} \right) + g e_{3} \\
\dot{q}_{t}^{u} = \omega_{u/I}^{u} - b_{\omega} - \eta_{\omega}.
\]

Integrating from the \(k^{th}\) node to the next yields

\[
\begin{align*}
\dot{p}_{u_{k+1}/I}^{J} & = \dot{p}_{u_{k}/I}^{J} + v_{u_{k}/I}^{J} \Delta t_{k} + \frac{1}{2} g e_{3} \Delta t_{k}^{2} + \int_{t_{k}}^{t_{k+1}} (R_{I}^{u})^{\top} \left( \ddot{a}_{u_{t}/I}^{u} - b_{a} - \eta_{a} \right) dt^{2} \\
\dot{v}_{u_{k+1}/I}^{J} & = \dot{v}_{u_{k}/I}^{J} + g e_{3} \Delta t_{k} + \int_{t_{k}}^{t_{k+1}} \omega_{u_{t}/I}^{u} dt^{2}.
\end{align*}
\]
\[
\int_{t_k}^{t_{k+1}} \left( \left( R_{t_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right) \right) dt
\]
\[
q_{i_k}^{u_{k+1}} = q_i^{u_k} \otimes \int_{t_k}^{t_{k+1}} \left( \bar{\omega}_{u_t/I} - b_\omega - \eta_\omega \right) dt,
\]
where \( \Delta t_k = t_{k+1} - t_k \). Rotating position and velocity to the IMU frame at time \( k \) gives

\[
R_{I}^{u_k} p_{u_k+1/I} = R_{I}^{u_k} \left( p_{u_k/I} + v_{u_k/I} \Delta t_k + \frac{1}{2} \bar{e}_3 \Delta t_k \right) + \int_{t_k}^{t_{k+1}} \left( \left( R_{u_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right) \right) dt^2 \tag{8.94}
\]
\[
R_{I}^{u_k} v_{u_k+1/I} = R_{I}^{u_k} \left( \left( v_{u_k/I} \right)^\top \bar{e}_3 \Delta t_k \right) + \int_{t_k}^{t_{k+1}} \left( \left( R_{u_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right) \right) dt
\]
\[
q_{i_k}^{u_{k+1}} = q_i^{u_k} \otimes \int_{t_k}^{t_{k+1}} \left( \bar{\omega}_{u_t/I} - b_\omega - \eta_\omega \right) dt. \tag{8.95}
\]

We now define the IMU preintegration terms as

\[
\alpha_{u_k+1/u_k} = \int_{t_k}^{t_{k+1}} \left. \left( \left( R_{u_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right) \right) \right| dt^2
\]
\[
\beta_{u_k+1/u_k} = \int_{t_k}^{t_{k+1}} \left( \left( R_{u_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right) \right) dt
\]
\[
\gamma_{u_k}^{u_{k+1}} = \int_{t_k}^{t_{k+1}} \left( \bar{\omega}_{u_t/I} - b_\omega - \eta_\omega \right) dt,
\]
where \( \gamma_{u_k}^{u_{k+1}} \) is the quaternion representing the rotation from node \( k \) to node \( k+1 \) and \( R_{u_k}^{u_t} = R \left( \gamma_{u_k}^{u_{k+1}} \right) \) is the rotation obtained by the integration of unbiased gyro measurements at time \( t \). From this, we can form a new set of differential equations

\[
\dot{\alpha}_{u_k+1/u_k} = \beta_{u_k+1/u_k}
\]
\[
\dot{\beta}_{u_k+1/u_k} = \left( R_{u_k}^{u_t} \right)^\top \left( \bar{a}_{u_t/I} - b_a - \eta_a \right)
\]
\[
\dot{\gamma}_{u_k}^{u_{k+1}} = \bar{\omega}_{u_t/I} - b_\omega - \eta_\omega,
\]

195
with their estimated counterparts given by

\[ \dot{\hat{\alpha}}_{u_{k+1}/u_k} = \hat{\beta}_{u_{k+1}/u_k} \]  
\[ \dot{\hat{\beta}}_{u_{k+1}/u_k} = (\hat{K}_{u_k})^\top \left( \hat{a}_{u_k/I} - \hat{b}_a \right) \]  
\[ \dot{\gamma}_{u_k}^{\mu_k+1} = \hat{\omega}_{u_k/I} - \hat{b}_\omega. \]  

(8.96)

(8.97)

Define the error states

\[ \tilde{\alpha}_{u_{k+1}/u_k} = \alpha_{u_{k+1}/u_k} - \hat{\alpha}_{u_{k+1}/u_k} \]  
\[ \tilde{\beta}_{u_{k+1}/u_k} = \beta_{u_{k+1}/u_k} - \hat{\beta}_{u_{k+1}/u_k} \]  
\[ \tilde{\gamma}_{u_k}^{\mu_k+1} = \gamma_{u_k}^{\mu_k+1} - \hat{\gamma}_{u_k}^{\mu_k+1} \]  
\[ \tilde{b}_a = b_a - \hat{b}_a \]  
\[ \tilde{b}_\omega = b_\omega - \hat{b}_\omega, \]

and we derive the error state kinematics

\[ \dot{\tilde{\alpha}}_{u_{k+1}/u_k} = \tilde{\beta}_{u_{k+1}/u_k} \]  
\[ \dot{\tilde{\beta}}_{u_{k+1}/u_k} = -\left( \tilde{\gamma}_{u_k}^{\mu_k+1} \right)^\top \left[ \left( \hat{a}_{u_k/I} - \hat{b}_a \right) \times \tilde{\gamma}_{u_k}^{\mu_k+1} + \tilde{b}_a + \eta_a \right] \]  
\[ \dot{\tilde{\gamma}}_{u_k}^{\mu_k+1} = -\left[ \left( \hat{\omega}_{u_k/I} - \hat{b}_\omega \right) \right] \times \tilde{\gamma}_{u_k}^{\mu_k+1} - \tilde{b}_\omega - \eta_\omega \]  
\[ \dot{\tilde{b}}_a = 0 \]  
\[ \dot{\tilde{b}}_\omega = 0. \]
Stacking the error terms into an error state and IMU noise terms into an input noise vector, we have

\[
\begin{bmatrix}
\tilde{\alpha}_{u_k+1} \\
\tilde{\beta}_{u_k+1} \\
\tilde{\gamma}_{u_k+1} \\
\tilde{b}_x \\
\tilde{b}_\omega
\end{bmatrix}
\]
\[
\eta = \begin{bmatrix} \eta_a \\ \eta_\omega \end{bmatrix},
\]

which is used to form a linear system \( \dot{\tilde{x}} = F\tilde{x} + Gu \), where

\[
F = \begin{bmatrix}
0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & - (\hat{R}_{u_k})^\top & \left[ \left( \tilde{a}_{u_t} / I - \tilde{b}_a \right) \times - (\hat{R}_{u_k})^\top \right] & 0 \\
0 & 0 & - \left[ \left( \tilde{a}_{u_t} / I - \tilde{b}_\omega \right) \times \right] & 0 & - I \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
G = \begin{bmatrix}
0 & 0 \\
- (\hat{R}_{u_k})^\top & 0 \\
0 & - I \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
\end{bmatrix}
\]

With IMU measurement noise covariance \( Q \in \mathbb{R}^{6 \times 6} \) and a time step \( \Delta t \), we propagate the state covariance by

\[
P_{t+\Delta t} = F_t P_t F_t^\top + G_t Q G_t^\top,
\]

where the covariance is initialized at zero and

\[
F_t = \exp(F \Delta t)
\]

197
\[ \approx I + F \Delta t \]

\[ G_t = \left( \int_0^{\Delta t} \exp (F \tau) d\tau \right) G \]

\[ \approx G \Delta t. \]

The Jacobian \( J \in \mathbb{R}^{15 \times 15} \) used to update the preintegration state is initialized at identity and updated by

\[ J_{t+\Delta t} = F_t J_t, \]

and the state is updated using sub-blocks of \( J_k \) by

\[
\begin{align*}
\alpha_{u_{k+1}/u_k}^{u_k} & \approx \hat{\alpha}_{u_{k+1}/u_k}^{u_k} + J_{\alpha/b_a} \hat{b}_a + J_{\alpha/b_\omega} \hat{b}_\omega \\
\beta_{u_{k+1}/u_k}^{u_k} & \approx \hat{\beta}_{u_{k+1}/u_k}^{u_k} + J_{\beta/b_a} \hat{b}_a + J_{\beta/b_\omega} \hat{b}_\omega \\
\gamma_{u_k}^{u_{k+1}} & \approx \hat{\gamma}_{u_k}^{u_{k+1}} \oplus J_{\gamma/b_\omega} \hat{b}_\omega.
\end{align*}
\]

The sub-block \( J_{\alpha/b_a} \) corresponds the rows and columns associated with \( \alpha_{u_{k+1}/u_k}^{u_k} \) and \( b_a \), respectively, and the other sub-blocks follow the same logic.
Chapter 9

Conclusions and Future Work

The goal of this dissertation was to develop a complete GN&C solution for fully autonomous target tracking and circumnavigation with unmanned aircraft. In doing so, this dissertation contributed a ground target tracking framework with guarantees and bounds on target localization and circumnavigation when tracking moving ground targets from an aerial vehicle dependent on a minimal set of relative sensors, such as an IMU and camera. Novel control algorithms were presented for fixed-wing and multirotor aircraft that operate independent of absolute position and heading with robustness to aircraft parameter error and external disturbances. Additionally, this dissertation combined two control and estimation paradigms: the well-known error-state formulation and modern on-manifold formulation. The combination of these two paradigms was shown to simplify derivation, implementation, and guarantee on-manifold operation of group elements. These contributions all form a complete GN&C solution for fully autonomous target tracking and circumnavigation with unmanned aircraft but could be improved in future work. The following sections provide several avenues to improve upon the work completed in this dissertation.

9.1 Image-Frame and Fixed-Frame Tracking Analysis and Comparison

It is unclear whether target tracking yields better results when performed in image coordinates or in ground coordinates. The work of Chapters 3 and 4 provide methods for obtaining ground target measurements in a fixed, 3D reference frame. The purpose of this line of inquiry would be to determine what conditions yield better track continuity for each formulation. This has not been
done until now primarily because the tools for localizing ground targets have not yet been fully developed.

Table 9.1: Advantages of fixed-frame and imaged-based tracking.

<table>
<thead>
<tr>
<th>Advantages of each tracking formulation</th>
<th>Fixed-frame</th>
<th>Image-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>• straightforward to use target specific motion propagation models</td>
<td></td>
<td>• noise introduced into the measurement only comes from one sensor (the camera)</td>
</tr>
<tr>
<td>• tracks are readily available for multi-vehicle problems</td>
<td></td>
<td>• image noise is typically minimal for modern cameras</td>
</tr>
<tr>
<td>• the tracker can account for geography</td>
<td></td>
<td>• the tracker does not depend on camera pose estimation</td>
</tr>
<tr>
<td>• poor homography estimates do not directly affect the tracker</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The remarks in Table 9.1 affect how well each formulation tracks targets. Several different metrics could be used to define this quality of tracking, such as track continuity and target state accuracy. These metrics allow us to put hard numbers on the tracking quality, but for many applications, the relative locations of the targets are needed regardless of the tracking method. Therefore, the dilemma of localize-then-track or track-(in the image)-then-localize could be explored, verifying items in Table 9.1 and augmenting that list.

9.2 Globally Stable SLAM in Real-Time

Perhaps the first requirement of autonomous flight is perceiving the environment and estimating the aircraft state. With this information, intelligent guidance and control can be applied to achieve some desired goal. Perception has been accomplished in this dissertation primarily with a camera and an IMU, while the estimation was performed with extended Kalman filters. An EKF lacks any convergence guarantees and therefore, provides an undesirable characteristic. Chapter 4 detailed an attempt at adapting nonlinear observer SLAM from [38] for real-time application, where the
key issues overcome were a moving origin and slow observer convergence. This was not a perfect solution nor was it demonstrated on real hardware.

The SLAM problem can also be formulated as a linear time-varying (LTV) Kalman filter [43], which retains the convergence guarantees of linear Kalman filters. The work in [43] assumes a known gyro bias that is estimated by their attitude heading reference system (AHRS) prior to the LTV Kalman filter. This is an undesirable characteristic and further work should explore the possibilities of both nonlinear observers and LTV Kalman filters for SLAM that include the gyro bias in the filter.

9.3 Guidance for Clustered MTT and Maximized Observability

With a state estimator and its observability conditions defined, the goal of this line of inquiry would be to develop guidance strategies that maximize estimator observability while accomplishing the desired task. Chapter 5 focused on a circumnavigation guidance strategy, similar to [21]. This could be improved in two ways: circumnavigate multiple targets as in [22] and use a more sophisticated guidance strategy.

A better guidance strategy may be to maximize observability of targets or landmarks relative to targets, depending on the target localization scheme being used. When directly estimating target states, the aircraft would have to adjust its velocity based on the target’s velocity, in order to achieve enhanced observability. When estimating target position relative to static landmarks as in [35], the aircraft would adjust its flight path to achieve observability of as many landmarks in view as possible.

9.4 System Identification of Fixed-Wing Aircraft Using Local Optimization

Identifying aircraft parameters that affect its dynamics is typically a difficult and tedious process. Chapter 6 showed that its controller is robust to aircraft parameter error, but having accurately
identified these parameters greatly improves control accuracy. Inspired by the offline calibration routine described in Chapter 8, a similar parameter estimation scheme for aircraft appears to be a possibility, given a good enough initial guess of the parameters. Using an existing tools such as [66], one could get an initial estimate of the aircraft’s parameters. With these initial estimates and some flight data containing IMU and RTK GPS sensor data, it may be possible to refine the initial parameter estimates in a graph-based, local optimization routine.

9.5 Robust Initialization of Visual-Inertial Systems

The filters developed in Chapters 7 and 8 lack convergence guarantees, but the EKF in Chapter 8 was shown to be robust given either landmark depth information or a good initialization of camera pose and velocity. With the emergence of moving horizon estimators (MHE), work on the initialization of visual-inertial filters has progressed [34, 75] because they also suffer from initialization error. An initialization routine for these visual-inertial systems with accuracy guarantees would be valuable to the robotics community. Further work into this subject should be a priority for future work related to visual-inertial estimation and would be an excellent addition to the works of Chapters 7 and 8.
Bibliography


