2020-02-01

Error-State Estimation and Control for a Multirotor UAV Landing on a Moving Vehicle

Michael David Farrell
Brigham Young University

Error-State Estimation and Control for a Multirotor UAV Landing on a Moving Vehicle

Michael David Farrell

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

Master of Science

Timothy W. McLain, Chair
Randal W. Beard
Marc D. Killpack

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Error-State Estimation and Control for a Multirotor UAV
Landing on a Moving Vehicle

Michael David Farrell
Department of Mechanical Engineering, BYU
Master of Science

Though multirotor unmanned aerial vehicles (UAVs) have become widely used during the past decade, challenges in autonomy have prevented their widespread use when moving vehicles act as their base stations. Emerging use cases, including maritime surveillance, package delivery and convoy support, require UAVs to autonomously operate in this scenario. This thesis presents improved solutions to both the state estimation and control problems that must be solved to enable robust, autonomous landing of multirotor UAVs onto moving vehicles.

Current state-of-the-art UAV landing systems depend on the detection of visual fiducial markers placed on the landing target vehicle. However, in challenging conditions, such as poor lighting, occlusion, or extreme motion, these fiducial markers may be undected for significant periods of time. This thesis demonstrates a state estimation algorithm that tracks and estimates the locations of unknown visual features on the target vehicle. Experimental results show that this method significantly improves the estimation of the state of the target vehicle while the fiducial marker is not detected.

This thesis also describes an improved control scheme that enables a multirotor UAV to accurately track a time-dependent trajectory. Rooted in Lie theory, this controller computes the optimal control signal based on an error-state formulation of the UAV dynamics. Simulation and hardware experiments of this control scheme show its accuracy and computational efficiency, making it a viable solution for use in a robust landing system.

Keywords: state estimation, optimal control, unmanned vehicles, autonomous vehicles, error state, multirotor, micro air vehicle, linear quadratic regulator, LQR, UAV

ACKNOWLEDGMENTS

I would first like to thank Dr. McLain for introducing me to the world of robotics. I don't know if I ever would have found this passion of mine if not for him. I would also like to thank Dr. McLain for mentoring me throughout my college career. His life advice and example have also made permanent impacts to many aspects of my life outside of engineering.

I would like to acknowledge the National Science Foundation for funding the research that is described in this thesis. I would also like to acknowledge Gaemus Collins, Adam Jaffe, and Aubrey Clausse at Planck Aerosystems who provided valuable insights and feedback on my research.

I would like to thank the members of the MAGICC lab that I have had the privelege to associate with every day. Thank you for collaborating with me, helping me, and teaching me. I consider myself lucky to have had the opportunity to work with each of you.

I would especially like to acknowledge my parents for supporting and encouraging me in my education and for teaching me important values of hard work, perserverance and positivity. Most of all, I would like to thank my wife, Emilie, for her continous love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

As multirotor unmanned aerial vehicles (UAVs) have become popular platforms for commercial and consumer products over the past decade, a variety of new use cases have emerged that require autonomous operation from larger vehicles acting as moving, mobile base stations. Such applications include maritime surveillance, package delivery, and convoy support (see Fig. 1.1). While there are many facets to operating from moving vehicles, this thesis works toward creating a robust solution for autonomously landing multirotor UAVs onto moving vehicles in challenging conditions.

Figure 1.1: Multirotor UAVs used to increase situational awareness for a convoy of vehicles [1].

1.2 Background

Autonomous landing of multirotor UAVs onto moving vehicles has been previously demonstrated in a variety of scenarios [2]; however, many problems still remain to create a truly robust

solution. Here we detail several of these outstanding problems as they relate to the three principal problems that must be solved by all autonomous UAVs: state estimation, motion planning, and control.

1.2.1 State Estimation

Most autonomous landing solutions require that the landing target vehicle is equipped with a visual fiducial marker (e.g. see Fig. 1.2) that serves as the designated landing platform for the UAV. In these methods, a camera mounted to the UAV detects the fiducial landing marker, providing information about the relative pose of the target vehicle [3]. These measurements are often used in a filtering framework, such as a Kalman filter [4], to produce a high-rate estimate of the relative state between the UAV and the target vehicle. During landing, it is likely that the fiducial marker is not detected for periods of time due to occlusion, sun glare, or UAV motion. For this reason, it is important that the relative motion between the target vehicle and the UAV be modeled and used to predict the relative state when the fiducial marker is not detected. However, even with a good model, these methods quickly fail when the fiducial landing marker is not detected for several seconds [5].

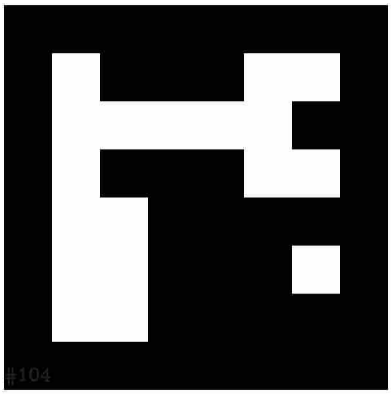

Figure 1.2: Visual fiducial markers such as this ArUco tag [6] are commonly used to assist in the landing of multirotor UAVs onto moving vehicles.

When landing in challenging conditions such as strong winds, bright sun, or dense fog, it is common that the fiducial marker is undetected by the UAV for long periods of time. Therefore, to create a truly robust landing solution, an accurate estimate of the state of the target vehicle must be maintained when the fiducial marker is not detected for significant durations. Chapter 3 describes an estimation algorithm, based on the error-state Kalman filter [7], that tracks and estimates the locations of visual features on the landing target vehicle to achieve this goal.

Visual feature tracking and estimation is a common technique in the field of visual odometry [8]; however, almost all implementations assume the tracked visual features are static with respect to an inertial reference frame. In this landing scenario, the target vehicle occupies progressively more of the field of view of the UAV's camera as the UAV descends. This makes it progressively more difficult for typical visual odometry algorithms to track features, resulting in a decreased reliability of the estimates as the UAV approaches the landing target. We also note that in most applications, more precise state estimates are required as the UAV approaches the landing target to avoid obstacles and to ensure a gentle landing. For these reasons, the described estimation algorithm only tracks and estimates visual features that are rigidly attached to the target vehicle. Simulation and hardware tests of the proposed algorithm demonstrated accurate and consistent estimates even when the fiducial marker was undetected for long periods of time.

1.2.2 Motion Planning

The motion planning problem associated with autonomous vehicles is especially important when navigating close to obstacles. In the case of landing on moving vehicles, it is often desired that the UAV descends vertically onto the landing pad to avoid potential obstacles such as buildings, terrain, or vehicle superstructure. When landing on certain vehicles, such as a boat at sea, the specific time of touchdown may also be important to minimize the risk of damage to the UAV.

Specific methods have been previously presented to generate dynamically feasible trajectories for UAVs landing on moving vehicles. One such method uses real-time model-predictive control techniques to generate trajectories for a UAV landing on a moving car [9]. While improvements to these methods are not presented in this thesis, directions for future work related to motion planning are described in Sec. 5.1.2.

1.2.3 Control

Many control algorithms have previously been presented that satisfy the requirements for robust landing on moving vehicles. However, a recent movement in the robotics community aims to appropriately deal with the evolution of a robot's state along a manifold using Lie theory [10]. Though Lie theory has been widely applied to the field of state estimation [11, 12], there is little work applying Lie theory to optimal control of UAVs.

Chapter 4 derives a linear-quadratic regulator (LQR) using Lie theory that computes control based on the error-state dynamics of the system. Not only is this a more principled approach than previous LQR methods, but it also achieves significant gains in computational efficiency. Simulation and hardware results show that the derived controller can accurately track a time-dependent trajectory, making it a good candidate for use in a robust landing system.

1.3 Summary of Contributions

The research described in this thesis makes two significant contributions:

- A method of state estimation is developed that allows a multirotor UAV to continue to operate reliably with respect to a landing target vehicle even when a fiducial landing marker is not detected for significant periods of time.
- An optimal control scheme for a multirotor UAV is derived using an error-state, LQR formulation that enables accurate tracking of any dynamically feasible, time-dependent trajectory.

These contributions are demonstrated in both simulation and hardware experiments found in Chapters 3 and 4. These results give us reason to believe that a complete and robust landing solution can be created by combining the proposed state estimation and control schemes with a competent motion planner.

1.4 Thesis Outline

Chapter 2 details the hardware and software systems used in the experiments described in this thesis. Chapter 3 describes an improved method of state estimation for a multirotor UAV landing on a moving vehicle. Chapter 4 derives and demonstrates the performance of an error-state

LQR controller for a multirotor UAV following a dynamically feasible, time-dependent trajectory. Chapter 5 provides concluding remarks including suggestions for future work that builds upon the work described in this thesis.

CHAPTER 2. EXPERIMENTAL APPARATUS

This chapter describes the software and hardware used in the experiments that are detailed in Chapters 3 and 4.

2.1 Software

2.1.1 Robot Operating System

The simulation and hardware experiments described in this thesis used the same C++ implementations of the proposed algorithms. This was made possible, in part, due to the use of the Robot Operating System[1] (ROS) as a middleware. ROS provides a way for separate programs, or nodes, to share information during runtime.

A network diagram of the software system can be seen in Fig. 2.1. The estimator and controller for the UAV were implemented as two separate nodes. The estimator node computed the state estimate of the UAV and the controller node computed the desired control action based on the estimated state. The ROSflight flight control stack [13] used this computed control action to actuate either a simulated UAV or the UAV hardware platform.

2.1.2 Gazebo

The Gazebo[2] simulation environment was used in conjunction with the ROSflight software-in-the-loop (SIL) simulation. This setup provided an easier transition from software experiments to hardware experiments as all of the necessary software was first proven in simulation.

[1] Robot Operating System: `www.ros.org`
[2] Gazebo: `www.gazebosim.org`

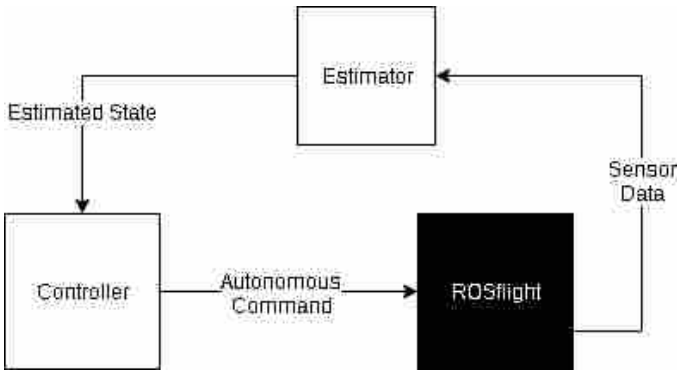

Figure 2.1: Network diagram of the software running during simulation and hardware experiments.

2.1.3 ROScopter

The simulation and hardware experiments described in this thesis also used the estimator and controller nodes of the ROScopter[3] project. During testing of the estimator proposed in Chapter 3, the ROScopter controller node was used to close the loop around the produced estimates. While experimenting with the controller proposed in Chapter 4, the ROScopter estimator node was used to provide state estimates of the UAV to the controller.

2.2 Hardware

The multirotor UAV used in hardware experimentation can be seen in Fig. 2.2. The UAV was built on a DJI Flamewheel 450 frame. Specific components contained on the UAV are detailed in the following subsections.

2.2.1 Flight Controller

The multirotor UAV was equipped with an OpenPilot CC3D Revolution 32-bit F4 flight controller as shown in Fig. 2.3. The flight controller ran the ROSflight firmware that provided an easy interface for the controller node on the onboard computer to control the UAV.

[3]ROScopter: `www.github.com/byu-magicc/roscopter`

Figure 2.2: Hardware platform used in experiments.

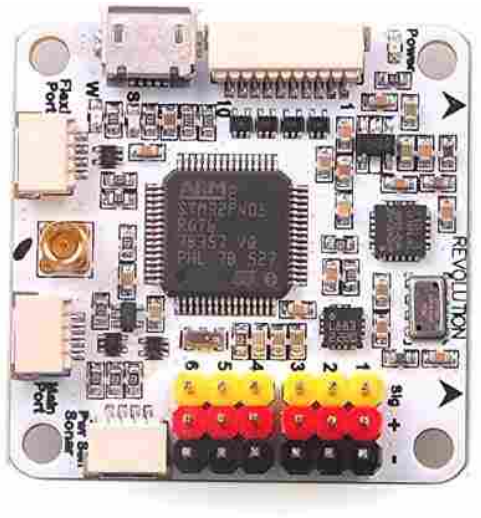

Figure 2.3: OpenPilot CC3D Revolution 32-bit F4 flight controller [14].

2.2.2 Onboard Computer

The computer onboard the UAV was an NVIDIA Jetson TX2 equipped with an Orbitty carrier board. This configuration is shown in Fig. 2.4. All computation was done on the onboard computer during the hardware experiments.

Figure 2.4: NVIDIA Jetson TX2 with an Orbitty carrier board attached [15].

2.2.3 Motion Capture

Hardware flight experiments were performed in the motion capture room in the MAGICC Lab at Brigham Young University. This room is equipped with an OptiTrack[4] motion tracking system that provided high-rate measurements of the position and attitude of the multirotor UAV during the experiments.

2.2.4 Camera

For the hardware experiments described in Chapter 3, the multirotor UAV was outfitted with an ELP USB camera with a 2.1 mm lens as shown in Fig. 2.5. The intrinsic parameters of the sensor were accurately calibrated using a software package provided by ROS[5]. However, the mounting position and attitude of the camera relative to the UAV were only roughly approximated.

Figure 2.5: ELP USB Camera with a 2.1 mm lens [16].

[4]OptiTrack: `www.optitrack.com`
[5]ROS camera_calibration: `wiki.ros.org/camera_calibration`

CHAPTER 3. IMPROVING STATE ESTIMATION OF A LANDING TARGET VEHICLE FROM A MULTIROTOR UAV USING VISUAL FEATURE TRACKING[1]

3.1 Introduction

Small multirotor unmanned air vehicles (UAVs) have rapidly become popular platforms for a variety of applications including inspection, reconnaissance, and search and rescue. For many of these use cases, UAVs are required to operate autonomously, as skilled pilots are not feasible since they are often unable to maintain direct line of sight to the UAV. Newly emerging use cases such as maritime surveillance and package delivery pose unique problems, requiring UAVs to operate autonomously from larger, mobile vehicles instead of from a stationary base station.

Nearly all current approaches to the operation of multirotor UAVs with respect to moving vehicles rely on the detection of a fiducial marker on the moving vehicle for relative pose measurements. One of the earliest of these works used a known configuration of infrared LEDs on the landing vehicle as a fiducial marker [17]. Since then, visual fiducial markers such as AprilTags [18] and ArUco markers [6] have become more widely used [3, 5, 19, 20]. While some landing methods control the UAV entirely based on the detections of the fiducial marker [2, 21], more robust methods compute control based on an estimate of the state of the target landing vehicle [5].

As the UAV descends toward the landing target, it is common that the fiducial marker remains undetected for periods of time due to poor lighting, occlusion, or extreme motion. For this reason, it is important that the dynamics of the target vehicle are modeled and used by the estimation algorithm to predict the state of the target vehicle when measurements are not available. The Kalman filter [4] has been frequently used for this task, producing accurate estimates when the fiducial marker is not detected for short periods of time [9]. Due to imperfect motion models, however, all of the mentioned approaches are likely to fail if the fiducial marker is not detected for significant periods of time. To improve these methods, we propose an estimation algorithm that

[1] This chapter is a modified version of the paper to be submitted to the IEEE/ASME Transactions on Mechatronics.

uses measurements of unknown visual features on the target vehicle in addition to measurements resulting from detections of a fiducial marker.

Many visual odometry methods such as [8, 22–24] use detected and tracked visual features to aid in camera motion estimation. In these methods, the tracked visual features are assumed to belong to the static world. Visual odometry techniques have also been previously applied to landing on moving platforms [25]. During landing, however, static features become more sparse as the dynamic target vehicle occupies progessively more of the field of view of the UAV's camera. This results in detoriating estimates as the UAV approaches the landing target. For this reason, the proposed estimator, instead, tracks and estimates the locations of visual features that are rigidly attached to the target vehicle. These tracked visual features provide information about the relative position of the target vehicle as long as the vehicle remains in the field of view of the camera. We show in simulation and hardware experiments that the estimation of these tracked features allows for accurate estimates of the state of the target vehicle even when the fiducial marker is not detected for significant periods of time.

The outline of this chapter is as follows. Sec. 3.2 explains the mathematical notation and conventions used throughout the chapter. Sec. 3.3 presents the proposed estimation algorithm including the state dynamics, state initialization and measurement models. Sec. 3.4 describes the simulation experiments conducted and Sec. 3.5 describes the hardware experiments conducted. Sec. 3.6 provides concluding remarks.

3.2 Mathematical Preliminaries

3.2.1 Notation

Throughout the paper, we represent vectors with a bold letter (e.g., $\mathbf{v}$) and matrices with a captial letter (e.g., A). Other common notation used throughout the paper is contained below.

R_a^b	Rotation matrix from reference frame a to b
$\mathbf{v}_{a/b}^c$	Vector state $\mathbf{v}$ of frame a w.r.t. frame b, expressed in frame c
$\hat{a}$	Estimate of true variable a
$\bar{a}$	Measurement of a
$\dot{a}$	Time derivative of a
$\tilde{a}$	Error of variable a, i.e., $\tilde{a} \triangleq a - \hat{a}$

We also make use of the following coordinate frames:

I	The inertial coordinate frame in north-east-down
v	The aircraft's vehicle-1 (body-level) coordinate frame
b	The aircraft's body-fixed coordinate frame
c	The camera frame
g	The target landing vehicle's body-fixed coordinate frame located at the desired landing location (goal) of the aircraft

We use the standard basis vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$, where $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\mathsf{T}$, $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\mathsf{T}$, and $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\mathsf{T}$. We also use the skew-symmetric matrix operator

$$[\mathbf{v}]_\times \triangleq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{3.1}$$

which is related to the cross-product between two vectors as

$$\mathbf{v} \times \mathbf{w} = [\mathbf{v}]_\times \mathbf{w}, \tag{3.2}$$

and the skew-symmetric identity

$$[\mathbf{v}]_\times \mathbf{w} = -[\mathbf{w}]_\times \mathbf{v}. \tag{3.3}$$

We also use the two-dimensional skew-symmetric operator which acts on a scalar

$$[a]_\times \triangleq \begin{bmatrix} 0 & -a \\ a & 0 \end{bmatrix}. \tag{3.4}$$

We use the identity matrix, I, as well as submatricies of I denoted with subscripts such as

$$I_{2\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.5}$$

3.2.2 Quaternion Representation

Unit quaternions $\in S^3$ throughout the paper follow the Hamiltonian notation

$$\mathbf{q} = q_0 + q_x i + q_y j + q_z k. \tag{3.6}$$

where i, j, and k are the fundamental quaternion units. We write quaternions as the tuple

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{\mathbf{q}} \end{pmatrix} \tag{3.7}$$

where q_0 represents the real portion of the quaternion and

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\mathsf{T} \tag{3.8}$$

respresents the complex portion of the quaternion.

Given this representation of the quaternion, the quaternion group operator $\otimes$ can be written as the matrix-like products

$$\mathbf{q}^a \otimes \mathbf{q}^b = \begin{pmatrix} q_0^a & (-\bar{\mathbf{q}}^a)^\mathsf{T} \\ \bar{\mathbf{q}}^a & q_0^a I + [\bar{\mathbf{q}}^a]_\times \end{pmatrix} \begin{pmatrix} q_0^b \\ \bar{\mathbf{q}}^b \end{pmatrix} \tag{3.9}$$

$$= \begin{pmatrix} q_0^b & (-\bar{\mathbf{q}}^b)^\mathsf{T} \\ \bar{\mathbf{q}}^b & q_0^b I - [\bar{\mathbf{q}}^b]_\times \end{pmatrix} \begin{pmatrix} q_0^a \\ \bar{\mathbf{q}}^a \end{pmatrix}. \tag{3.10}$$

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

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0 [\bar{\mathbf{q}}]_\times + 2\bar{\mathbf{q}}\bar{\mathbf{q}}^\top \in SO(3). \tag{3.11}$$

We also note that rotations are written equivalently as $\mathbf{q}_a^b = R\left(\mathbf{q}_a^b\right) = R_a^b$ throughout the paper.

To operate in a vector space, we frequently convert between the Lie group, S^3, and the vector space, $\mathbb{R}^3$, that is isomorphic to the Lie algebra. This is done with the exponential and logarithmic mappings. The exponential mapping for a unit quaternion is defined as

$$\begin{aligned} \exp_{\mathbf{q}} &: \mathbb{R}^3 \to S^3 \\ \exp_{\mathbf{q}}(\delta) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\delta\|}{2}\right) \\ \sin\left(\frac{\|\delta\|}{2}\right)\frac{\delta}{\|\delta\|} \end{bmatrix}, \end{aligned} \tag{3.12}$$

with the corresponding logarithmic map defined as

$$\begin{aligned} \log_{\mathbf{q}} &: S^3 \to \mathbb{R}^3 \\ \log_{\mathbf{q}}(\mathbf{q}) &\triangleq 2\,\mathrm{atan2}\left(\|\bar{\mathbf{q}}\|, q_0\right)\frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}. \end{aligned} \tag{3.13}$$

With this formulation, δ also represents the axis-angle representation of a rotation where the unit vector $\frac{\delta}{\|\delta\|}$ represents the axis of rotation, and $\|\delta\|$ represents the angular magnitude of rotation. When $\|\delta\| \approx 0$, we employ the small-angle approximations of the quaternion exponential and logarithm given by

$$\exp_{\mathbf{q}}(\delta) \approx \begin{bmatrix} 1 \\ \frac{\delta}{2} \end{bmatrix} \tag{3.14}$$

$$\log_{\mathbf{q}}(\mathbf{q}) \approx 2\,\mathrm{sign}(q_0)\,\bar{\mathbf{q}}. \tag{3.15}$$

3.2.3 Planar Rotations

We parameterize planar rotations as an angle, ψ, which respresents the angle of rotation about a given axis. We treat $\psi \in \mathbb{R}^1$ so that common addition and subtraction operators can be used such as

$$\psi_a^c = \psi_a^b + \psi_b^c \tag{3.16}$$

$$\psi_a^b = \psi_a^c - \psi_b^c. \tag{3.17}$$

We note, however, that with this formulation, all addition and subtraction operations must be wrapped such that the resultant angle $\psi \in [-\pi, \pi)$. The passive 2D rotation matrix can be created from any ψ as

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}. \tag{3.18}$$

If ψ represents the angle of rotation about the z axis of a reference frame, then the corresponding passive 3D rotation matrix is given by

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.19}$$

3.3 Estimation

This work focuses on the estimation of the state of the landing target vehicle. However, as several of the measurement models employed depend on the state of the UAV, we also estimate the state of the UAV in the same filter to properly account for the uncertainty in its state. We, therefore, estimate the position, attitude, and velocity of the UAV given by $\hat{\mathbf{p}}^I_{b/I}$, $\hat{\mathbf{q}}^b_I$, and $\hat{\mathbf{v}}^b_{b/I}$. In addition, we estimate bias states for the acclerometer and gyroscope sensors that are used as inputs to the filter. These estimated states are given by $\hat{\beta}_a$ and $\hat{\beta}_\omega$.

The estimated state of the target vehicle is defined as the position, velocity, attitude, and angular rate of the target vehicle. We note that the estimated position of the target vehicle, $\hat{\mathbf{p}}^v_{g/b}$, is relative to the position of the UAV. We formulate this state relatively, as this relative state is observable even with poor estimates of the UAV's global position, $\hat{\mathbf{p}}^I_{b/I}$, due to the relative information provided by the measurements as described in Sec. 3.3.3. For this work, we assume that the target vehicle's motion is constrained to a two-dimensional plane. This means that the estimated target vehicle velocity, $\hat{\mathbf{v}}^g_{g/I}$, is only of two dimensions and that the estimated attitude, $\hat{\psi}^g_I$, represents a planar rotation as described in Sec. 3.2.3, implying that the estimated angular rate, $\hat{\omega}^g_{g/I}$ is of one dimension.

As previously mentioned, we also estimate the locations of unknown visual features on the target vehicle. The vectors $\hat{\mathbf{r}}^g_{1/g}, \ldots \hat{\mathbf{r}}^g_{n/g}$ represent the estimated locations of visual features $1, \ldots n$

with respect to the goal frame and expressed in the goal frame. As we assume these features are rigidly attached to the target vehicle, these vectors remain constant as the vehicle moves and rotates. We show in the experiments described in Sec. 3.4 and Sec. 3.5 that the addition of only ten visual features to the estimated state significantly improves the estimates of the state of the target vehicle while the fiducial landing marker is not detected.

We express the full state of the estimated system as the tuple

$$\hat{\mathbf{x}} = \left(\hat{\mathbf{x}}_{\text{UAV}}, \hat{\mathbf{x}}_{\text{Goal}}, \hat{\mathbf{x}}_{\text{Features}}\right) \tag{3.20}$$

with the components defined as

$$\hat{\mathbf{x}}_{\text{UAV}} = \left(\hat{\mathbf{p}}^{I}_{b/I}, \hat{\mathbf{q}}^{b}_{I}, \hat{\mathbf{v}}^{b}_{b/I}, \hat{\beta}_{a}, \hat{\beta}_{\omega}\right) \in \mathbb{R}^3 \times S^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \tag{3.21}$$

$$\hat{\mathbf{x}}_{\text{Goal}} = \left(\hat{\mathbf{p}}^{v}_{g/b}, \hat{\mathbf{v}}^{g}_{g/I}, \hat{\psi}^{g}_{I}, \hat{\omega}^{g}_{g/I}\right) \in \mathbb{R}^3 \times \mathbb{R}^2 \times \mathbb{R}^1 \times \mathbb{R}^1 \tag{3.22}$$

$$\hat{\mathbf{x}}_{\text{Features}} = \left(\hat{\mathbf{r}}^{g}_{1/g}, \ldots \hat{\mathbf{r}}^{g}_{n/g}\right) \in \mathbb{R}^3 \times \ldots \mathbb{R}^3. \tag{3.23}$$

The inputs to the estimated system are given by

$$\mathbf{u} = \left(\bar{\mathbf{a}}^{b}_{b/I}, \bar{\omega}^{b}_{b/I}\right) \in \mathbb{R}^3 \times \mathbb{R}^3, \tag{3.24}$$

which are directly measured from an inertial measurement unit mounted on the UAV.

We also note that the estimated state is of dynamic size. As visual features are detected they are added to the estimated state until a maximum size of the state is reached. As visual features leave the field of view of the camera, or are otherwise no longer tracked, they are removed from the estimated state to make room for new visual features to be added.

3.3.1 Error-State Definition

As the estimated state is not a vector, but rather a tuple of Lie groups, we employ the error-state Kalman filter (ESKF) as described in [12]. We define the error-state of the estimated system

as

$$\tilde{\mathbf{x}} = \left[\tilde{\mathbf{p}}^I_{b/I}, \tilde{\theta}^b_I, \tilde{\mathbf{v}}^b_{b/I}, \tilde{\beta}_a, \tilde{\beta}_\omega, \tilde{\mathbf{p}}^v_{g/b}, \tilde{\mathbf{v}}^g_{g/I}, \tilde{\psi}^g_I, \tilde{\omega}^g_{g/I}, \tilde{\mathbf{r}}^g_{1/g}, \ldots \tilde{\mathbf{r}}^g_{n/g}\right] \in \mathbb{R}^{22+3n} \tag{3.25}$$

with the error-state components related to the vector states, $\mathbf{x_v}$, defined with the vector subtraction operator as

$$\tilde{\mathbf{x}}_\mathbf{v} \triangleq \mathbf{x}_\mathbf{v} - \hat{\mathbf{x}}_\mathbf{v} \tag{3.26}$$

such that

$$\tilde{\mathbf{p}}^I_{b/I} = \mathbf{p}^I_{b/I} - \hat{\mathbf{p}}^I_{b/I}. \tag{3.27}$$

We reiterate that we treat $\psi^g_I \in \mathbb{R}^1$ such that

$$\tilde{\psi}^g_I = \psi^g_I - \hat{\psi}^g_I. \tag{3.28}$$

We follow [12], defining the error-state of the quaternion, $\mathbf{q}^b_I$, as the minimal representation

$$\tilde{\theta}^b_I \triangleq \log_\mathbf{q}\left(\left(\hat{\mathbf{q}}^b_I\right)^{-1} \otimes \mathbf{q}^b_I\right) \tag{3.29}$$

which implies

$$\mathbf{q}^b_I = \hat{\mathbf{q}}^b_I \otimes \exp_\mathbf{q}\left(\tilde{\theta}^b_I\right). \tag{3.30}$$

As rotation matrices concatenate in the order opposite to quaternions, (3.30) can also be expressed as

$$R^b_I = R\left(\exp_\mathbf{q}\left(\tilde{\theta}^b_I\right)\right)\hat{R}^b_I. \tag{3.31}$$

To derive the error-state dynamics and the measurement residual Jacobians in the following sections, we use an approximation for (3.31) developed by first expanding the quaternion exponential using (3.14) as $\tilde{\theta}^b_I$ is assumed to be small

$$R^b_I \approx R\left(\begin{bmatrix} 1 \\ \frac{1}{2}\tilde{\theta}^b_I \end{bmatrix}\right)\hat{R}^b_I. \tag{3.32}$$

We then employ (3.11), neglecting higher-order terms, to yield the approximation

$$R_I^b \approx \left(I - \left[\tilde{\theta}_I^b\right]_\times\right)\hat{R}_I^b. \tag{3.33}$$

It can similarly be shown that

$$\left(R_I^b\right)^\mathsf{T} \approx \left(\hat{R}_I^b\right)^\mathsf{T}\left(I + \left[\tilde{\theta}_I^b\right]_\times\right). \tag{3.34}$$

3.3.2 Propagation Model

To model the motion of the UAV, we use common rigid-body kinematics given by

$$\begin{aligned}
\dot{\mathbf{p}}_{b/I}^I &= \left(R_I^b\right)^\mathsf{T}\mathbf{v}_{b/I}^b \\
\dot{\mathbf{q}}_I^b &= \mathbf{q}_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right) \end{pmatrix} \\
\dot{\mathbf{v}}_{b/I}^b &= R_I^b\mathbf{g}^I + \left[\mathbf{v}_{b/I}^b\right]_\times\left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right) + \left(\bar{\mathbf{a}}_{b/I}^b - \beta_a - \upsilon_a\right) \\
\dot{\beta}_a &= \eta_{\beta_a} \\
\dot{\beta}_\omega &= \eta_{\beta_\omega},
\end{aligned} \tag{3.35}$$

where $\mathbf{g}^I$ represents the gravity vector expressed in the inertial frame, η_{β_a} and η_{β_ω} are zero-mean Gaussian noise processes corresponding to the state dynamics, and υ_ω and υ_a are zero-mean Gaussian noise processes corresponding to the noise in the inputs to the system.

We model the motion of the landing target vehicle with a constant-velocity and constant-angular-velocity motion model such that

$$\begin{aligned}
\dot{\mathbf{p}}_{g/b}^v &= I_{3\times 2}\left(R_I^g\right)^\mathsf{T}\mathbf{v}_{g/I}^g - \left(R_I^b\right)^\mathsf{T}\mathbf{v}_{b/I}^b \\
\dot{\mathbf{v}}_{g/I}^g &= \eta_{gv} \\
\dot{\psi}_I^g &= \omega_{g/I}^g \\
\dot{\omega}_{g/I}^g &= \eta_{g\omega},
\end{aligned} \tag{3.36}$$

where η_{gv} and $\eta_{g\omega}$ are zero-mean Gaussian noise processes. Though this is a simplified motion model for the target vehicle, we show in Sec. 3.4 and Sec. 3.5 that it is satisfactory for our experiments. We intend the motion model of the target vehicle to be easily modified for vehicles with more complex motion such as a boat at sea.

As mentioned previously, we assume the tracked visual features are rigidly attached to the landing vehicle such that

$$\dot{\mathbf{r}}^g_{i/g} = \mathbf{0}. \tag{3.37}$$

In the ESKF, the estimated state is propagated independently of the filter using the expected value of the modeled dynamics. We use the expected values of (3.35), (3.36), and (3.37) given by

$$\begin{aligned}
\dot{\hat{\mathbf{p}}}^I_{b/I} &= \left(\hat{R}^b_I\right)^\mathsf{T} \hat{\mathbf{v}}^b_{b/I} \\
\dot{\hat{\mathbf{q}}}^b_I &= \hat{\mathbf{q}}^b_I \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\left(\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right) \end{pmatrix} \\
\dot{\hat{\mathbf{v}}}^b_{b/I} &= \hat{R}^b_I \mathbf{g}^I + \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \left(\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right) + \left(\bar{\mathbf{a}}^b_{b/I} - \hat{\beta}_a\right) \\
\dot{\hat{\beta}}_a &= \mathbf{0} \\
\dot{\hat{\beta}}_\omega &= \mathbf{0} \\
\dot{\hat{\mathbf{p}}}^v_{g/b} &= I_{3\times 2}\left(\hat{R}^g_I\right)^\mathsf{T} \hat{\mathbf{v}}^g_{g/I} - \left(\hat{R}^b_I\right)^\mathsf{T} \hat{\mathbf{v}}^b_{b/I} \\
\dot{\hat{\mathbf{v}}}^g_{g/I} &= \mathbf{0} \\
\dot{\hat{\psi}}^g_I &= \hat{\omega}^g_{g/I} \\
\dot{\hat{\omega}}^g_{g/I} &= 0 \\
\dot{\hat{\mathbf{r}}}^g_{i/g} &= \mathbf{0}.
\end{aligned} \tag{3.38}$$

The error-state dynamics used to propagate the filter are found by relating the modeled true-state dynamics from (3.35), (3.36), and (3.37) with (3.38) using the error-state definitions

from (3.26) and (3.29). The first-order approximation of the error-state dynamics is given by

$$
\begin{aligned}
\dot{\tilde{\mathbf{p}}}^I_{b/I} &\approx \left(\hat{R}^b_I\right)^{\mathsf{T}} \tilde{\mathbf{v}}^b_{b/I} - \left(\hat{R}^b_I\right)^{\mathsf{T}} \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tilde{\theta}^b_I \\
\dot{\tilde{\theta}}^b_I &\approx -\left[\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right]_\times \tilde{\theta}^b_I - \tilde{\beta}_\omega - \upsilon_\omega \\
\dot{\tilde{\mathbf{v}}}^b_{b/I} &\approx \left[\hat{R}^b_I \mathbf{g}^I\right]_\times \tilde{\theta}^b_I - \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tilde{\beta}_\omega - \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \upsilon_\omega - \left[\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right]_\times \tilde{\mathbf{v}}^b_{b/I} - \tilde{\beta}_a - \upsilon_a \\
\dot{\tilde{\beta}}_a &= \eta_{\beta_a} \\
\dot{\tilde{\beta}}_\omega &= \eta_{\beta_\omega} \\
\dot{\tilde{\mathbf{p}}}^v_{g/b} &\approx I_{3\times 2}\left(\hat{R}^g_I\right)^{\mathsf{T}} \tilde{\mathbf{v}}^g_{g/I} + I_{3\times 2}\left(\hat{R}^g_I\right)^{\mathsf{T}} \left[\tilde{\psi}^g_I\right]_\times \hat{\mathbf{v}}^g_{g/I} + \left(\hat{R}^b_I\right)^{\mathsf{T}} \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tilde{\theta}^b_I - \left(\hat{R}^b_I\right)^{\mathsf{T}} \tilde{\mathbf{v}}^b_{b/I} \\
\dot{\tilde{\mathbf{v}}}^g_{g/I} &= \eta_{gv} \\
\dot{\tilde{\psi}}^g_I &= \tilde{\omega}^g_{g/I} \\
\dot{\tilde{\omega}}^g_{g/I} &= \eta_{g\omega} \\
\dot{\tilde{\mathbf{r}}}^g_{i/g} &= \mathbf{0},
\end{aligned}
\tag{3.39}
$$

or succinctly,

$$
\dot{\tilde{\mathbf{x}}} = f\left(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}\right). \tag{3.40}
$$

The derivation of these error-state dynamics can be found in Appendix A. In practice, the expected value of the error state remains zero over the propagation window, and only the error covariance, P, is propagated. The continous-time derivative of the error covariance is given by

$$
\dot{P} = FP + PF^{\mathsf{T}} + GQ_{\mathbf{u}}G^{\mathsf{T}} + Q_{\mathbf{x}} \tag{3.41}
$$

where $Q_{\mathbf{u}}$ is the input noise covariance, $Q_{\mathbf{x}}$ is the process noise covariance,

$$F = \frac{\partial \dot{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{x}}} \tag{3.42}$$

$$= \begin{bmatrix}
\mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{p}}}^I_{b/I}}{\partial \tilde{\theta}^b_I} & \frac{\partial \dot{\tilde{\mathbf{p}}}^I_{b/I}}{\partial \tilde{\mathbf{v}}^b_{b/I}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \frac{\partial \dot{\tilde{\theta}}^b_I}{\partial \tilde{\theta}^b_I} & \mathbf{0} & \mathbf{0} & \frac{\partial \dot{\tilde{\theta}}^b_I}{\partial \tilde{\beta}_\omega} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{v}}}^b_{b/I}}{\partial \tilde{\theta}^b_I} & \frac{\partial \dot{\tilde{\mathbf{v}}}^b_{b/I}}{\partial \tilde{\mathbf{v}}^b_{b/I}} & \frac{\partial \dot{\tilde{\mathbf{v}}}^b_{b/I}}{\partial \tilde{\beta}_a} & \frac{\partial \dot{\tilde{\mathbf{v}}}^b_{b/I}}{\partial \tilde{\beta}_\omega} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{p}}}^v_{g/b}}{\partial \tilde{\theta}^b_I} & \frac{\partial \dot{\tilde{\mathbf{p}}}^v_{g/b}}{\partial \tilde{\mathbf{v}}^b_{b/I}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{p}}}^v_{g/b}}{\partial \tilde{\mathbf{v}}^g_{g/I}} & \frac{\partial \dot{\tilde{\mathbf{p}}}^v_{g/b}}{\partial \tilde{\psi}^g_I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & \frac{\partial \dot{\tilde{\psi}}^g_I}{\partial \tilde{\omega}^g_{g/I}} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0}
\end{bmatrix}, \tag{3.43}$$

with

$$\frac{\partial \dot{\tilde{\mathbf{p}}}^I_{b/I}}{\partial \tilde{\theta}^b_I} = -\left(\hat{R}^b_I\right)^\mathsf{T} \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tag{3.44}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}^I_{b/I}}{\partial \tilde{\mathbf{v}}^b_{b/I}} = \left(\hat{R}^b_I\right)^\mathsf{T} \tag{3.45}$$

$$\frac{\partial \dot{\tilde{\theta}}^b_I}{\partial \tilde{\theta}^b_I} = -\left[\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right]_\times \tag{3.46}$$

$$\frac{\partial \dot{\tilde{\theta}}_I^b}{\partial \tilde{\beta}_\omega} = -I \tag{3.47}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}_{b/I}^b}{\partial \tilde{\theta}_I^b} = \left[\hat{R}_I^b \mathbf{g}\right]_\times \tag{3.48}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}_{b/I}^b}{\partial \tilde{\mathbf{v}}_{b/I}^b} = -\left[\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right]_\times \tag{3.49}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}_{b/I}^b}{\partial \tilde{\beta}_a} = -I \tag{3.50}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}_{b/I}^b}{\partial \tilde{\beta}_\omega} = -\left[\hat{\mathbf{v}}_{b/I}^b\right]_\times \tag{3.51}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}_{g/b}^v}{\partial \tilde{\theta}_I^b} = \left(\hat{R}_I^b\right)^\mathsf{T} \left[\hat{\mathbf{v}}_{b/I}^b\right]_\times \tag{3.52}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}_{g/b}^v}{\partial \tilde{\mathbf{v}}_{b/I}^b} = -\left(\hat{R}_I^b\right)^\mathsf{T} \tag{3.53}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}_{g/b}^v}{\partial \tilde{\mathbf{v}}_{g/I}^g} = I_{3\times 2}\left(\hat{R}_I^g\right)^\mathsf{T} \tag{3.54}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}_{g/b}^v}{\partial \tilde{\psi}_I^g} = I_{3\times 2}\left(\hat{R}_I^g\right)^\mathsf{T} [1]_\times \hat{\mathbf{v}}_{g/I}^g \tag{3.55}$$

$$\frac{\partial \dot{\tilde{\psi}}_I^g}{\partial \tilde{\omega}_{g/I}^g} = 1, \tag{3.56}$$

and

$$G = \frac{\partial \dot{\tilde{\mathbf{x}}}}{\partial \upsilon} \tag{3.57}$$

$$= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -I \\ -I & -\left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.58}$$

However, to ensure numerical stability, we propagate the covariance using a first-order discrete approximation defined by

$$P_{k+1} = F_k P_k F_k^\mathsf{T} + G_k Q_u G_k^\mathsf{T} + Q_x \Delta t^2 \tag{3.59}$$

where

$$F_k \approx I + F\Delta t \tag{3.60}$$

$$G_k \approx G\Delta t. \tag{3.61}$$

3.3.3 Measurement Models

When updating the filter with measurements, we make use of the partial Kalman update which has been shown to improve estimates of bias states and constant values [26]. The partial Kalman update provides a means to limit the effect of measurement updates to certain estimated

states by providing a tuning vector

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_N \end{bmatrix} \tag{3.62}$$

where $\lambda_i \in [0,1]$ determines the proportion of the measurement update applied to state i. In practice, we use $\lambda < 1$ only for the bias states, β_a and β_ω, and the constant-value states $\mathbf{r}^g_{1/g}, \dots \mathbf{r}^g_{n/g}$.

With this formulation, when a measurement is received, we compute the Kalman gain

$$K = PH^\mathsf{T}\left(HPH^\mathsf{T} + R\right)^{-1} \tag{3.63}$$

where H is the residual Jacobian for the measurement and R is the measurement covariance. We then follow [26], using K to update the filter such that

$$\hat{\tilde{\mathbf{x}}}^+ = \lambda \odot K\mathbf{r} \tag{3.64}$$

$$P^+ = P + \Lambda \odot \left((I-KH)P(I-KH)^\mathsf{T} + KRK^\mathsf{T} - P\right) \tag{3.65}$$

where $\odot$ is the Hadamard product, $\mathbf{r}$ is the residual of the measurement and

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^\mathsf{T} \tag{3.66}$$

$$\Lambda = \mathbf{1}\lambda^\mathsf{T} + \lambda\mathbf{1} - \lambda\lambda^\mathsf{T}. \tag{3.67}$$

As the estimate of the error state of the system, $\hat{\tilde{\mathbf{x}}}$, becomes non-zero after this update, we use this estimate to correct the estimated state, $\hat{\mathbf{x}}$. As the estimated state is not a vector, this correction is done piecewise. The vector components of the estimated state are updated as

$$\hat{\mathbf{x}}^+_\mathbf{v} = \hat{\mathbf{x}}_\mathbf{v} + \tilde{\mathbf{x}}_\mathbf{v} \tag{3.68}$$

and the quaternion state is updated as

$$\left(\hat{\mathbf{q}}^b_I\right)^+ = \hat{\mathbf{q}}^b_I \otimes \exp_\mathbf{q}\left(\tilde{\theta}^b_I\right). \tag{3.69}$$

After this correction, the estimate of the error state of the system resets to zero.

The measurement model, residual model and residual Jacobian are defined below for each type of measurement used in the filter.

Global UAV Position Measurement

We assume to receive a measurement of the position of the UAV with respect to the inertial frame. In our experiments, this measurement results from a motion capture system; however, in other applications, a sensor such as a real-time kinematic GPS unit could provide this measurement. The measurement and its model are written as

$$\mathbf{z}_{\text{pos}} = h_{\text{pos}}(\mathbf{x}) + \eta_{\text{pos}} \tag{3.70}$$

$$h_{\text{pos}}(\mathbf{x}) = \mathbf{p}^I_{b/I}, \tag{3.71}$$

where η_{pos} is a zero-mean Gaussian process describing the sensor noise. For a given measurement of position, $\bar{\mathbf{z}}_{\text{pos}}$, the residual is given by

$$\mathbf{r}_{\text{pos}} = \bar{\mathbf{z}}_{\text{pos}} - h_{\text{pos}}(\hat{\mathbf{x}}). \tag{3.72}$$

For the error-state Kalman filter, the residual is modeled as

$$\mathbf{r}_{\text{pos}} = \mathbf{z}_{\text{pos}} - h_{\text{pos}}(\hat{\mathbf{x}}) \tag{3.73}$$

$$= \mathbf{p}^I_{b/I} + \eta_{\text{pos}} - \hat{\mathbf{p}}^I_{b/I} \tag{3.74}$$

$$= \tilde{\mathbf{p}}^I_{b/I} + \eta_{\text{pos}}. \tag{3.75}$$

This results in the residual Jacobian

$$H_{\text{pos}} = \frac{\partial \mathbf{r}_{\text{pos}}}{\partial \tilde{\mathbf{x}}} \tag{3.76}$$

$$= \begin{bmatrix} \dfrac{\partial \mathbf{r}_{\text{pos}}}{\partial \tilde{\mathbf{p}}^I_{b/I}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{3.77}$$

$$= \begin{bmatrix} I_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}. \tag{3.78}$$

Global UAV Attitude Measurement

Similar to the position measurement above, we assume to receive a measurement of the attitude of the body frame of the UAV with respect to the inertial frame. In our experiments, this measurement results from a motion capture system; however, in other applications, a sensor such as an attitude and heading reference system could provide this measurement. The measurement and its model are written as

$$\mathbf{z}_{\text{att}} = h_{\text{att}}(\mathbf{x}) \otimes \exp_{\mathbf{q}}(\eta_{\text{att}}) \tag{3.79}$$

$$h_{\text{att}}(\mathbf{x}) = \mathbf{q}_I^b, \tag{3.80}$$

where η_{att} is a zero-mean Gaussian process describing the sensor noise. For a given measurement of attitude, $\bar{\mathbf{z}}_{\text{att}}$, the residual is given by

$$\mathbf{r}_{\text{att}} = \log_{\mathbf{q}}\left(h_{\text{att}}(\hat{\mathbf{x}})^{-1} \otimes \bar{\mathbf{z}}_{\text{att}}\right), \tag{3.81}$$

which is modeled as

$$\mathbf{r}_{\text{att}} = \log_{\mathbf{q}}\left(h_{\text{att}}(\hat{\mathbf{x}})^{-1} \otimes \mathbf{z}_{\text{att}}\right) \tag{3.82}$$

$$= \log_{\mathbf{q}}\left(\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \mathbf{q}_I^b \otimes \exp_{\mathbf{q}}(\eta_{\text{att}})\right). \tag{3.83}$$

This is expanded using (3.30) and simplified to yield

$$\mathbf{r}_{\text{att}} = \log_{\mathbf{q}}\left(\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \hat{\mathbf{q}}_I^b \otimes \exp_{\mathbf{q}}\left(\tilde{\theta}_I^b\right)\right) + \eta_{\text{att}} \tag{3.84}$$

$$= \tilde{\theta}_I^b + \eta_{\text{att}}. \tag{3.85}$$

This results in the residual Jacobian

$$H_{\text{att}} = \frac{\partial \mathbf{r}_{\text{att}}}{\partial \tilde{\mathbf{x}}} \tag{3.86}$$

$$= \begin{bmatrix} \mathbf{0} & \dfrac{\partial \mathbf{r}_{\text{att}}}{\partial \tilde{\theta}_I^b} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{3.87}$$

$$= \begin{bmatrix} \mathbf{0} & I_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}. \tag{3.88}$$

Fiducial Translation Measurement

We assume that a known fiducial marker serves as the desired landing position for the multirotor UAV on the target vehicle. The goal frame is, therefore, located at the center of the fiducial marker. In consequence, every detection of the fiducial marker yields a measurement of the relative translation and rotation from the camera frame to the goal frame. The measurement and its model for this relative translation measurement are written as

$$\mathbf{z}_{\text{ft}} = h_{\text{ft}}(\mathbf{x}) + \eta_{\text{ft}} \tag{3.89}$$

$$h_{\text{ft}}(\mathbf{x}) = \mathbf{p}_{g/c}^{c} \tag{3.90}$$

$$= R_b^c \left(R_I^b \mathbf{p}_{g/b}^{v} - \mathbf{p}_{c/b}^{b} \right), \tag{3.91}$$

where R_b^c and $\mathbf{p}_{c/b}^{b}$ are assumed to be known constants, and η_{ft} is a zero-mean Gaussian process describing the measurement noise. For a given measurement of the relative translation to the fiducial marker, $\bar{\mathbf{z}}_{\text{ft}}$, the residual is given by

$$\mathbf{r}_{\text{ft}} = \bar{\mathbf{z}}_{\text{ft}} - h_{\text{ft}}(\hat{\mathbf{x}}) \tag{3.92}$$

and modeled as

$$\mathbf{r}_{\text{ft}} = \mathbf{z}_{\text{ft}} - h_{\text{ft}}(\hat{\mathbf{x}}) \tag{3.93}$$

$$= R_b^c \left(R_I^b \mathbf{p}_{g/b}^{v} - \mathbf{p}_{c/b}^{b} \right) + \eta_{\text{ft}} - R_b^c \left(\hat{R}_I^b \hat{\mathbf{p}}_{g/b}^{v} - \mathbf{p}_{c/b}^{b} \right) \tag{3.94}$$

$$= R_b^c R_I^b \mathbf{p}_{g/b}^{v} - R_b^c \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^{v} + \eta_{\text{ft}}. \tag{3.95}$$

Using (3.33), we expand (3.95) and ignore second-order terms to yield

$$\mathbf{r}_{\text{ft}} \approx R_b^c \left(I - \left[\tilde{\theta}_I^b \right]_\times \right) \hat{R}_I^b \left(\hat{\mathbf{p}}_{g/b}^v + \tilde{\mathbf{p}}_{g/b}^v \right) - R_b^c \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v + \eta_{\text{ft}} \tag{3.96}$$

$$\approx R_b^c \hat{R}_I^b \tilde{\mathbf{p}}_{g/b}^v - R_b^c \left[\tilde{\theta}_I^b \right]_\times \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v + \eta_{\text{ft}} \tag{3.97}$$

$$\approx R_b^c \hat{R}_I^b \tilde{\mathbf{p}}_{g/b}^v + R_b^c \left[\hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v \right]_\times \tilde{\theta}_I^b + \eta_{\text{ft}}. \tag{3.98}$$

This results in the residual Jacobian

$$H_{\text{ft}} = \frac{\partial \mathbf{r}_{\text{ft}}}{\partial \tilde{\mathbf{x}}} \tag{3.99}$$

$$= \begin{bmatrix} \mathbf{0} & \dfrac{\partial \mathbf{r}_{\text{ft}}}{\partial \tilde{\theta}_I^b} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dfrac{\partial \mathbf{r}_{\text{ft}}}{\partial \tilde{\mathbf{p}}_{g/b}^v} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{3.100}$$

$$= \begin{bmatrix} \mathbf{0} & R_b^c \left[\hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v \right]_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & R_b^c \hat{R}_I^b & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}. \tag{3.101}$$

Fiducial Rotation Measurement

We use the relative rotation measurement that results from a detection of the fiducial marker to create a pseudo measurement of the orientation of the goal frame. In theory, a measurement model could be developed to use the entire measurement, $\bar{R}_c^g$; however, in practice, this may cause complications if the fiducial landing marker is not perfectly aligned with the plane of the target vehicle's motion (i.e., the fiducial marker is slighly rolled or pitched with respect to the goal frame). The pseudo measurement is created with

$$\bar{\psi}_I^g = \text{yaw} \left(\bar{R}_c^g R_b^c \hat{R}_I^b \right) \tag{3.102}$$

where yaw () is a function that extracts the yaw angle from a rotation matrix given by

$$\text{yaw} \left(\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \right) = \text{atan2} \left(r_{12}, r_{11} \right). \tag{3.103}$$

The measurement and its model are written as

$$\mathbf{z}_{\text{fr}} = h_{\text{fr}}(\mathbf{x}) + \eta_{\text{fr}} \tag{3.104}$$

$$h_{\text{fr}}(\mathbf{x}) = \psi_I^g, \tag{3.105}$$

where η_{fr} is a zero-mean Gaussian process describing the measurement noise. For a given (pseudo) measurement of the rotation of the fiducial marker, $\bar{\psi}_I^g$, the residual is given by

$$r_{\text{fr}} = \bar{\psi}_I^g - h_{\text{fr}}(\hat{\mathbf{x}}), \tag{3.106}$$

which is modeled as

$$r_{\text{fr}} = \mathbf{z}_{\text{fr}} - h_{\text{fr}}(\hat{\mathbf{x}}) \tag{3.107}$$

$$= \psi_I^g + \eta_{\text{fr}} - \hat{\psi}_I^g \tag{3.108}$$

$$= \tilde{\psi}_I^g + \eta_{\text{fr}}. \tag{3.109}$$

This results in the residual Jacobian

$$H_{\text{fr}} = \frac{\partial \mathbf{r}_{\text{fr}}}{\partial \tilde{\mathbf{x}}} \tag{3.110}$$

$$= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dfrac{\partial \mathbf{r}_{\text{fr}}}{\partial \tilde{\psi}_I^g} & 0 & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{3.111}$$

$$= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 0 & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}. \tag{3.112}$$

Visual Feature Pixel Measurement

The estimator receives measurements of the location of each tracked visual feature in the camera image. We assume that the pixel locations received have already been corrected for lens

distortion. We also assume to know the camera intrinsic matrix,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.113}$$

where f_x and f_y are the focal lengths of the camera and c_x and c_y are the coordinates of the principal point in the camera image. Using the pinhole camera model, we express the pixel location of visual feature i as

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \frac{1}{\mathbf{e}_3^\mathsf{T} \mathbf{p}_{i/c}^c} K \mathbf{p}_{i/c}^c \tag{3.114}$$

where

$$\mathbf{p}_{i/c}^c = R_b^c \left(R_I^b \left(R_I^g \right)^\mathsf{T} \mathbf{r}_{i/g}^g + R_I^b \mathbf{p}_{g/b}^v - \mathbf{p}_{c/b}^b \right). \tag{3.115}$$

The measurement and its model are, therefore, written as

$$\mathbf{z}_{\text{pix}} = h_{\text{pix}}(\mathbf{x}) + \eta_{\text{pix}} \tag{3.116}$$

$$h_{\text{pix}}(\mathbf{x}) = \begin{bmatrix} p_x & p_y \end{bmatrix}^\mathsf{T} \tag{3.117}$$

$$= \frac{1}{\mathbf{e}_3^\mathsf{T} \mathbf{p}_{i/c}^c} I_{2\times 3} K \mathbf{p}_{i/c}^c, \tag{3.118}$$

where η_{pix} is a zero-mean Gaussian process describing the measurement noise. For a given measurement of the pixel location of a feature, $\bar{\mathbf{z}}_{\text{pix}}$, the residual is given by

$$\mathbf{r}_{\text{pix}} = \bar{\mathbf{z}}_{\text{pix}} - h_{\text{pix}}(\hat{\mathbf{x}}), \tag{3.119}$$

which is modeled as

$$\mathbf{r}_{\text{pix}} = \mathbf{z}_{\text{pix}} - h_{\text{pix}}\left(\hat{\mathbf{x}}\right) \tag{3.120}$$

$$= \frac{1}{\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c} I_{2\times 3} K \mathbf{p}_{i/c}^c + \eta_{\text{pix}} - \frac{1}{\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c} I_{2\times 3} K \hat{\mathbf{p}}_{i/c}^c. \tag{3.121}$$

This results in the residual Jacobian

$$H_{\text{pix}} = \frac{\partial \mathbf{r}_{\text{pix}}}{\partial \tilde{\mathbf{x}}} \tag{3.122}$$

$$\approx \frac{1}{\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c} I_{2\times 3} K \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}_{i/c}^c - \frac{\mathbf{e}_3^\mathsf{T} \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}_{i/c}^c}{\left(\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c\right)^2} I_{2\times 3} K \hat{\mathbf{p}}_{i/c}^c \tag{3.123}$$

where the non-zero components of $\frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}_{i/c}^c$ are given by

$$\frac{\partial}{\partial \tilde{\theta}_I^b} \mathbf{p}_{i/c}^c = R_b^c \left[\hat{R}_I^b \left(\left(\hat{R}_I^g\right)^\mathsf{T} \hat{\mathbf{r}}_{i/g}^g + \hat{\mathbf{p}}_{g/b}^v\right)\right]_\times \tag{3.124}$$

$$\frac{\partial}{\partial \tilde{\mathbf{p}}_{g/b}^v} \mathbf{p}_{i/c}^c = R_b^c \hat{R}_I^b \tag{3.125}$$

$$\frac{\partial}{\partial \tilde{\psi}_I^g} \mathbf{p}_{i/c}^c = R_b^c \hat{R}_I^b \left(\hat{R}_I^g\right)^\mathsf{T} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times \hat{\mathbf{r}}_{i/g}^g \tag{3.126}$$

$$\frac{\partial}{\partial \tilde{\mathbf{r}}_{i/g}^g} \mathbf{p}_{i/c}^c = R_b^c \hat{R}_I^b \left(\hat{R}_I^g\right)^\mathsf{T}. \tag{3.127}$$

The derivation of this residual Jacobian is found in Appendix B.

3.3.4 State Initialization

While the states associated with the UAV and sensor biases, $\hat{\mathbf{x}}_{\text{UAV}}$, are initialized at startup, the states associated with the target vehicle, $\hat{\mathbf{x}}_{\text{Goal}}$ are initialized upon receiving the first measurement from the detection of the fiducial marker given by

$$\bar{\mathbf{z}} = \begin{bmatrix} \bar{\mathbf{p}}_{g/c}^c & \bar{R}_c^g \end{bmatrix}. \tag{3.128}$$

We use this measurement to initialize the estimates of the states with

$$\hat{\mathbf{p}}_{g/b}^{v} = \left(\hat{R}_{I}^{b}\right)^{\mathsf{T}} \left((R_{b}^{c})^{\mathsf{T}} \bar{\mathbf{p}}_{g/c}^{c} + \mathbf{p}_{c/b}^{b}\right) \tag{3.129}$$

$$\hat{\mathbf{v}}_{g/I}^{I} = \mathbf{0} \tag{3.130}$$

$$\hat{\psi}_{I}^{g} = \text{yaw}\left(\bar{R}_{c}^{g} R_{b}^{c} \hat{R}_{I}^{b}\right) \tag{3.131}$$

$$\hat{\omega}_{g/I}^{g} = 0. \tag{3.132}$$

Similar to the manner in which the target vehicle states are intialized, the visual feature states are initialized based on the first corresponding measurement received. As we only receive a measurement of the pixel location of the visual feature in the camera image,

$$\bar{\mathbf{z}} = \begin{bmatrix} \bar{p}_x & \bar{p}_y \end{bmatrix}, \tag{3.133}$$

the estimated state, $\hat{\mathbf{r}}_{i/g}^{g}$, is not entirely observered. Therefore, to initialize the feature state, we assume the feature lies in the xy plane of the goal frame, initializing the z component of $\hat{\mathbf{r}}_{i/g}^{g}$ to zero. To compute the x and y components of $\hat{\mathbf{r}}_{i/g}^{g}$, we start with (3.114) which can be rotated and solved to yield

$$\mathbf{p}_{i/c}^{v} = \left(\mathbf{e}_{3}^{\mathsf{T}} \mathbf{p}_{i/c}^{c}\right) \left(R_{I}^{b}\right)^{\mathsf{T}} (R_{b}^{c})^{\mathsf{T}} K^{-1} \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ 1 \end{bmatrix}. \tag{3.134}$$

As $\mathbf{e}_{3}^{\mathsf{T}} \mathbf{p}_{i/c}^{c}$ is unknown, we define

$$\mathring{\mathbf{p}}_{i/c}^{v} \triangleq \left(R_{I}^{b}\right)^{\mathsf{T}} (R_{b}^{c})^{\mathsf{T}} K^{-1} \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ 1 \end{bmatrix}, \tag{3.135}$$

which is equivalent to $\mathbf{p}_{i/c}^{v}$ up to a scale factor. As previously mentioned, we estimate this scale factor by assuming $\mathbf{e}_{3}^{\mathsf{T}} \mathbf{r}_{i/g}^{g} = 0$ such that

$$\mathbf{e}_{3}^{\mathsf{T}} \hat{\mathbf{p}}_{i/c}^{v} = \mathbf{e}_{3}^{\mathsf{T}} \hat{\mathbf{p}}_{g/b}^{v} - \mathbf{e}_{3}^{\mathsf{T}} \left(\hat{R}_{I}^{b}\right)^{\mathsf{T}} \mathbf{p}_{c/b}^{b}. \tag{3.136}$$

We therefore initialize

$$\hat{\mathbf{r}}^g_{i/g} = \hat{R}^g_I \left(\hat{\mathbf{p}}^v_{i/b} - \hat{\mathbf{p}}^v_{g/b} \right) \tag{3.137}$$

where

$$\hat{\mathbf{p}}^v_{i/b} = \frac{\mathbf{e}_3^\mathsf{T} \hat{\mathbf{p}}^v_{i/c}}{\mathbf{e}_3^\mathsf{T} \mathring{\mathbf{p}}^v_{i/c}} \mathring{\mathbf{p}}^v_{i/c} + \left(\hat{R}^b_I \right)^\mathsf{T} \mathbf{p}^b_{c/b}. \tag{3.138}$$

3.4 Simulation

To demonstrate the effectiveness of the proposed estimation algorithm, we first present results from simulation. A multirotor UAV and a landing target vehicle were simulated using the dynamics presented in (3.35) and (3.36) with the zero-mean Gaussian noise processes as described in Table 3.1. Positions of visual features on the target vehicle were also simulated by randomly sampling from the uniform distribution

$$\mathbf{r}^g_{i/g} = \mathscr{U} \begin{pmatrix} [-2,2] \\ [-2,2] \\ [-1,1] \end{pmatrix}. \tag{3.139}$$

To emulate a visual feature tracker losing track of features, simulated features randomly disappeared at each time step of the simulation. When a feature disappeared, a new feature was generated by sampling from (3.139) such that several hundred, different features were used during a 30 second simulation.

Table 3.1: Simulated Motion Model Parameters.

Parameter	**Std. Deviation**
η_{β_a}	0.05 m/s^2
η_{β_ω}	0.01 rad/s
η_{gv}	5 m/s
$\eta_{g\omega}$	5 rad/s

The initial state of the landing target vehicle was given by

$$\begin{bmatrix} \mathbf{p}^{v}_{g/b} \\ \mathbf{v}^{g}_{g/I} \\ \psi^{g}_{I} \\ \omega^{g}_{g/I} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 5 \end{bmatrix}^{\mathsf{T}} \text{m} \\ \begin{bmatrix} 0.5 & 0.0 \end{bmatrix}^{\mathsf{T}} \text{m/s} \\ 1.5 \text{ rad} \\ 0.5 \text{ rad/s} \end{bmatrix}. \tag{3.140}$$

The simulated multirotor UAV was controlled to orbit around the true position of the target vehicle such that the target vehicle remained in the field of view of the simulated camera for the duration of the simulation.

The sensor measurements described in Sec. 3.3.3 were simulated using the true state of the simulation. The rate at which each simulated sensor was sampled and the standard deviation of the zero-mean Guassian noise added to each measurement are found in Table 3.2. The simulated 640×480 pixel camera, described by its intrinsic matrix

$$K = \begin{bmatrix} 410 & 0 & 320 \\ 0 & 420 & 240 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.141}$$

was oriented at a yaw angle of $\pi/2$ rad with respect to the body frame, such that

$$R^{c}_{b} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.142}$$

and was positioned such that $\mathbf{p}^{b}_{c/b} = \begin{bmatrix} 0.25, & -0.20, & 0.40 \end{bmatrix}^{\mathsf{T}}$ m.

We present the results of two simulation experiments: Fig. 3.1 shows the results of a simulation experiment in which the ESKF does not estimate the locations of any visual features ($n = 0$); Fig. 3.2 shows the results of a simulation experiment in which the ESKF estimates the location of ten visual features ($n = 10$). To demonstrate the performance of the proposed estimation algorithm when the fiducial landing marker is not detected, the measurements from the fiducial landing

Table 3.2: Simulated Sensor Characteristics.

Measurement Type	**Std. Deviation**	**Rate**
Accelerometer	0.2 m/s^2	250 Hz
Gyroscope	0.1 rad/s	250 Hz
UAV global position	0.1 m	10 Hz
UAV global attitude	0.1 rad	10 Hz
Fiducial marker translation	0.1 m	30 Hz
Fiducial marker rotation	0.1 rad	30 Hz
Visual feature image point	2.0 pixels	30 Hz

marker were not used in these experiments after $t = 5$ seconds. Note that we do not include plots for the estimated UAV states, $\hat{\mathbf{x}}_{\text{UAV}}$, as it is well known that these states can be accurately estimated with the given measurements.

Fig. 3.1 clearly shows that the covariance of the estimated states began to grow unbounded after $t = 5$ s when $n = 0$. This matches our intuition, as during that time, the filter received no measurements to contrain these states. We can also see significant error in the estimated state beginning at $t = 5$ s. This error resulted from an imperfect model of the target vehicle's motion. Fig. 3.2, however, shows that when $n = 10$, the estimates remained accurate and the covariance of the estimates remained small for the duration of the simulation. Note the scale differences between Fig. 3.1 and Fig. 3.2 required to properly depict these results.

To further demonstrate performance, the two previous experiments were each repeated 100 times. The error in the estimated position of the target vehicle in the xy plane of the inertial frame is plotted with respect to time for each of these experiments in Fig. 3.3. While the error when $n = 10$ remained under one meter for all 100 simulations, the error when $n = 0$ grew quickly, reaching an error of over ten meters in many cases.

3.5 Hardware

3.5.1 Platform

The proposed estimation algorithm was implemented and flown in hardware to validate the performance observed in the simulation results. The multirotor used for the experiments was built

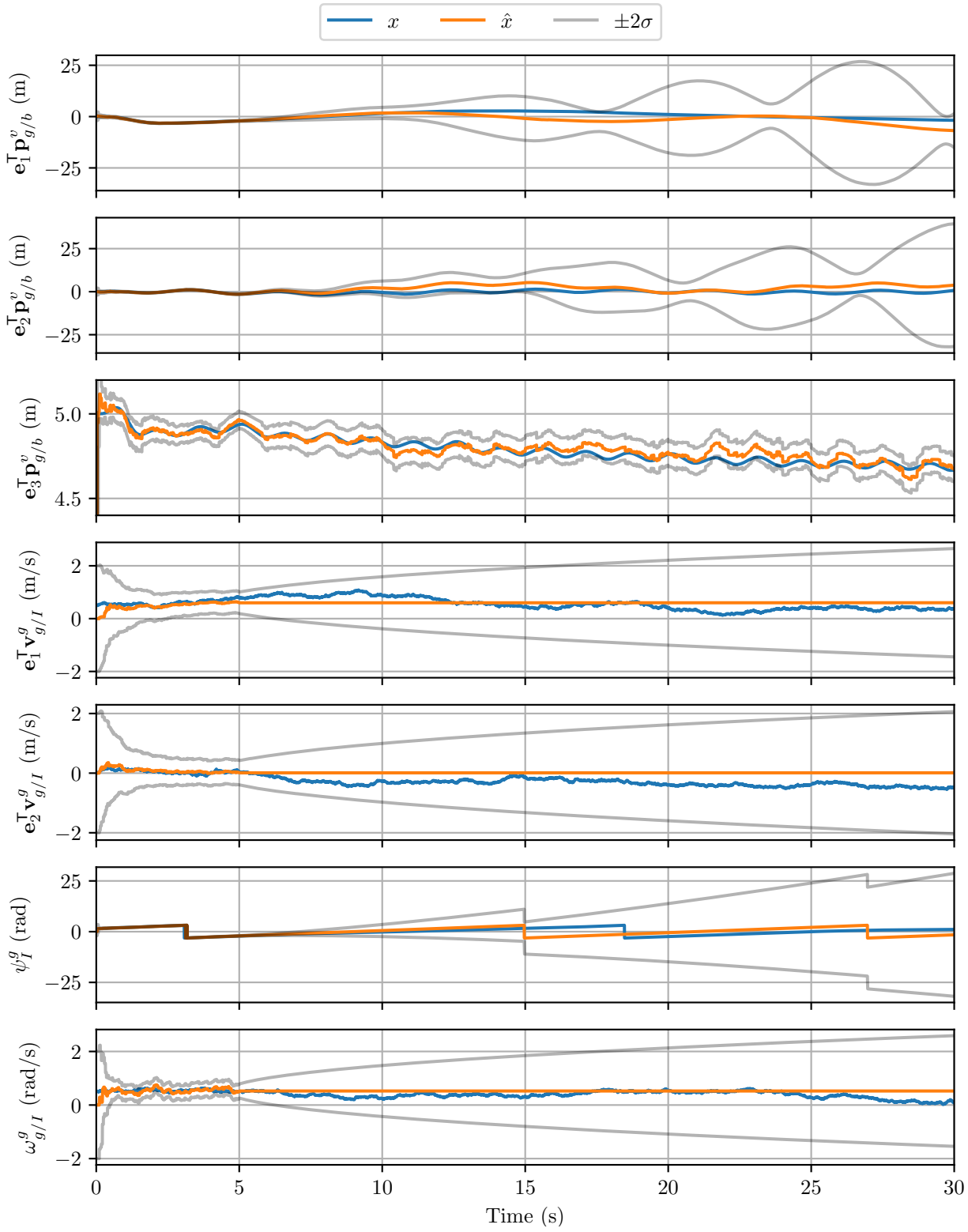

Figure 3.1: Simulation results when the ESKF estimated the positions of *no* visual features. The blue line represents the true state while the orange line represents the estimated state. The two grey lines show $\pm 2\sigma$ bounds for the estimate based on the estimated covariance. Measurements from the fiducial marker were not used after $t = 5$ s to demonstrate the performance of the estimator.

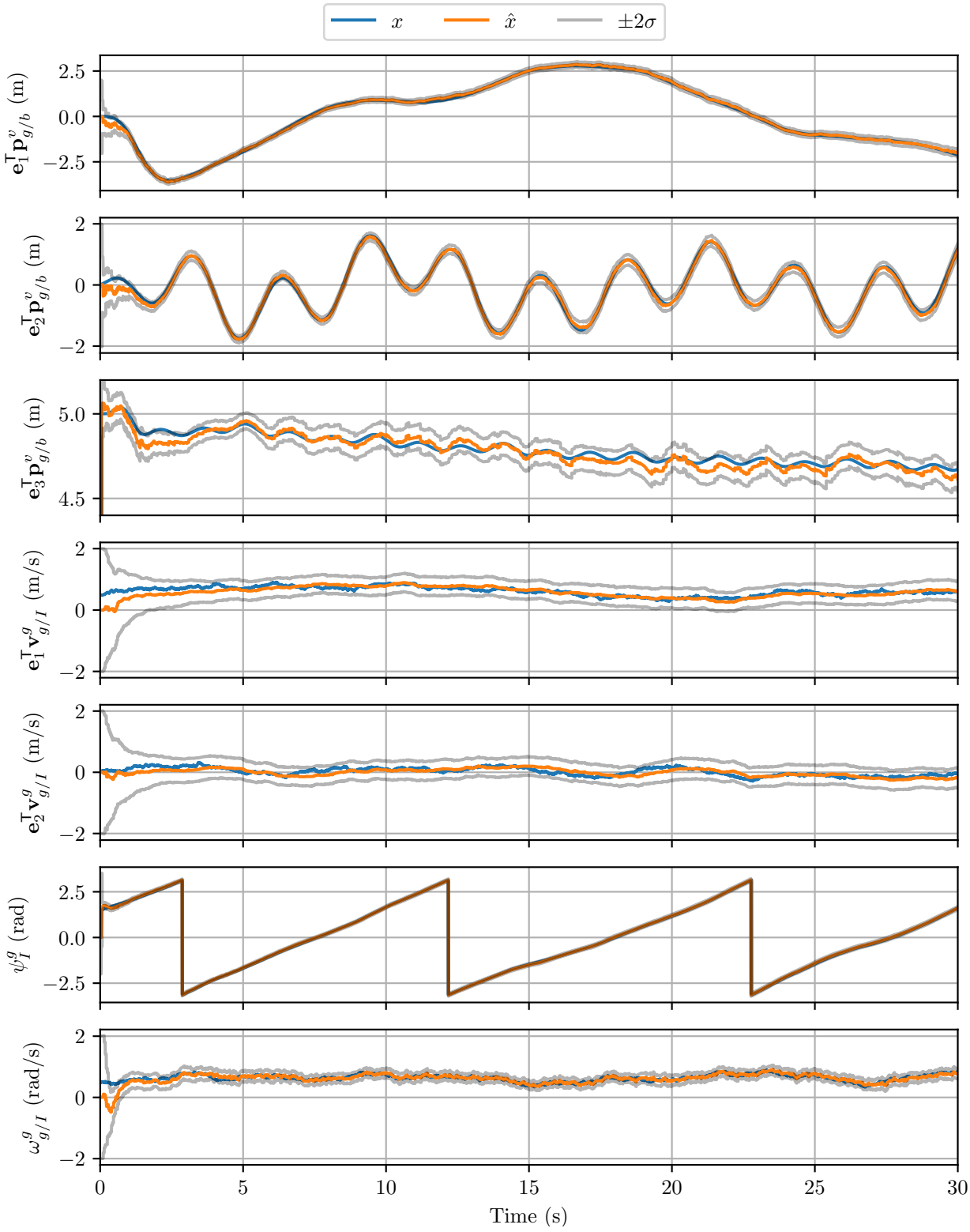

Figure 3.2: Simulation results when the ESKF estimated the positions of *ten* visual features. The blue line represents the true state while the orange line represents the estimated state. The two grey lines show $\pm 2\sigma$ bounds for the estimate based on the estimated covariance. Measurements from the fiducial marker were not used after $t = 5$ s to demonstrate the performance of the estimator.

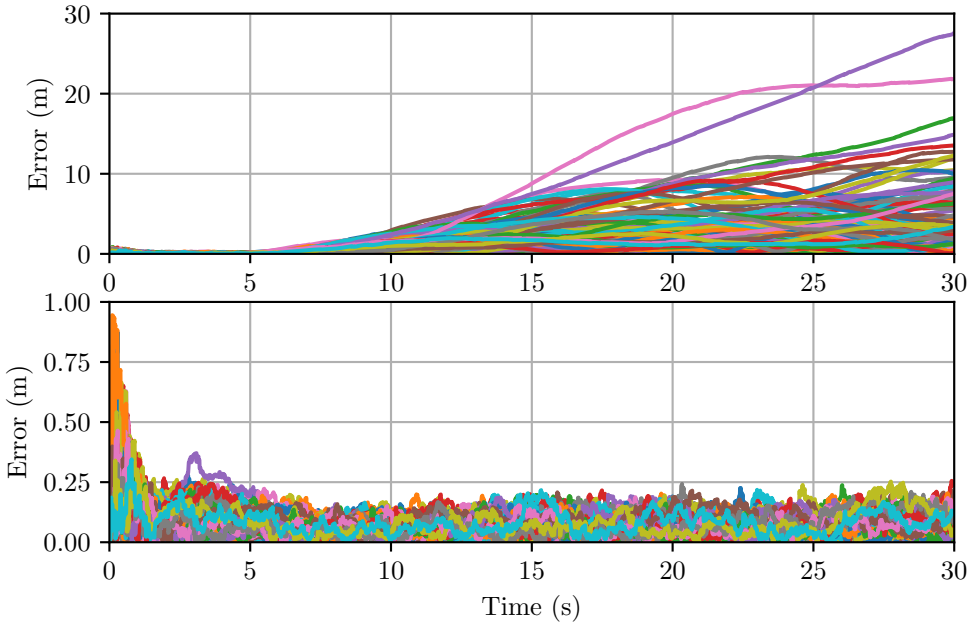

Figure 3.3: Error of the estimated position of the target vehicle in the xy plane. The top plot shows the error with respect to time for 100 simulations in which the ESKF estimates the positions of zero visual features. The bottom plot shows the error with respect to time for 100 simulations in which the ESKF estimates the positions of ten visual features. Measurements from the fiducial marker are not used after $t = 5$ s to demonstrate the performance of the estimator.

on a DJI 450 Flamewheel frame. All computation was done onboard the UAV on an NVIDIA Jetson TX2 using the Robot Operating System[2]. An ELP USB Camera with a 2.1 mm lens was mounted to the bottom of the UAV such that the camera faced downward during flight. The image from this camera was used for visual feature tracking and fiducial marker detection.

The multirotor UAV was manually flown until the fiducial landing marker was detected. Upon detection, a successive-loop PID control scheme took full control of the UAV, closing the loop around the estimated states. Throughout the flight, the UAV was controlled to maintain a 0.5 m altitude directly above the landing target such that $\hat{\mathbf{p}}^{v}_{g/b} = \begin{bmatrix} 0 & 0 & 0.5 \end{bmatrix}$ m. The relative yaw angle between the UAV and the goal frame was also controlled to zero. Commands resulting from this control scheme were sent from the onboard computer to a CC3D Revolution 32bit F4 flight controller running the ROSflight firmware [13].

[2]Robot Operating System: `www.ros.org`

3.5.2 Fiducial Landing Marker

The fiducial landing marker used in our flight experiments was a 6.1 cm $\times$ 6.1 cm ArUco marker [27], as pictured in the top, right corner of Fig. 3.4. When detected in a camera image, a relative translation and rotation from the camera frame to the marker was estimated using the detected positions of the corners of the marker in the camera image and the known size of the marker.

3.5.3 Feature Tracking

As mentioned in Sec. 3.3, the estimation algorithm uses measurements of visual features that are rigidly attached to the landing target vehicle. It is important to note that determining which visual features in a camera image are attached to the target vehicle is not a trivial problem. We leave this problem as future work and circumvent this problem by flying low enough to the target vehicle such that it occupies the entire field of view of the camera.

Visual features were first detected using a FAST feature detector [28]. The detected features were then tracked from one frame to the next using optical flow [29]. To remove features that had been poorly tracked, we periodically estimated the essential matrix between the current camera image and a stored keyframe. Outliers to this estimated essential matrix were discarded, and new FAST features were detected to replace them. For our experiments, the feature tracker attempted to maintain 250 tracked features at all times. As the proposed estimator only used ten visual features at a time, a subset of the tracked features which had persisted the longest were provided to the estimator for each camera image.

Each visual feature that was acquired and tracked was assigned a unique integer identification number. The estimator used these identification numbers to determine when visual features were no longer tracked and when new visual features were acquired. An example camera image from the UAV showing the subset of tracked features provided to the estimator with the corresponding identification numbers is seen in Fig. 3.4.

Figure 3.4: A processed camera image from the multirotor UAV's camera where the landing target vehicle occupies the entire image. The ArUco marker is pictured in the top, right corner of the image. Each green circle shows the tracked location of a visual feature used by the estimator. The red number associated with each visual feature is the unique integer ID assigned by the feature tracker.

3.5.4 Indoor Motion Capture

The hardware flight experiments were conducted in the indoor motion capture room in the MAGICC Lab at Brigham Young University. An Optitrack motion capture system provided meaurements of the global position and attitude of the UAV throughout the flights. The motion capture system was also used as ground truth for the position and attitude of the target vehicle.

3.5.5 Landing Target Vehicle

As flight tests were conducted in a small indoor environment, the landing target vehicle was designed to be small in an attempt to better extend to outdoor scenarios in which the UAV is to land on larger vehicles such as trucks or boats. The target vehicle, pictured in Fig. 3.5, was manually driven during the experiments, roughly following an oval.

3.5.6 Experiment Results

The results of the flight experiment are shown in Fig. 3.6. The fiducial landing marker was first detected at $t = 20$ s, where the plots begin. To demonstrate the ability of the system to maintain

Figure 3.5: Multirotor UAV shown autonomously tracking the landing target vehicle.

good tracking of a target vehicle when a fiducial marker is not detected for long periods of time, the fiducial marker detection was turned off ten seconds after initial detection, at $t = 30$ s. It is clear that the estimates of the position, velocity, attitude, and angular velocity of the target vehicle remained accurate and consistent for the duration of the experiment. These accurate estimates allowed the UAV to continue to control relative to the target vehicle, tracking closely above the landing target as the target vehicle moved around the room. After the target vehicle completed two full laps around the room, at approximately $t = 102$ s, manual control of the UAV was resumed, ending the experiment. A video of the flight experiment can be found at `https://youtu.be/3AyjCI0c1Nc`.

3.6 Conclusion

The proposed ESKF provides a method for maintaining accurate and consistent estimates of the state of a landing target vehicle when a fiducial landing marker is not detected for significant periods of time. This improvement is achieved by tracking and estimating the locations of unknown visual features on the target vehicle. The presented experiments demonstrated that the addition of just ten visual feature positions to the estimated state is sufficient to allow for reliable operation with respect to a target vehicle for more than one minute without detection of a fiducial landing marker.

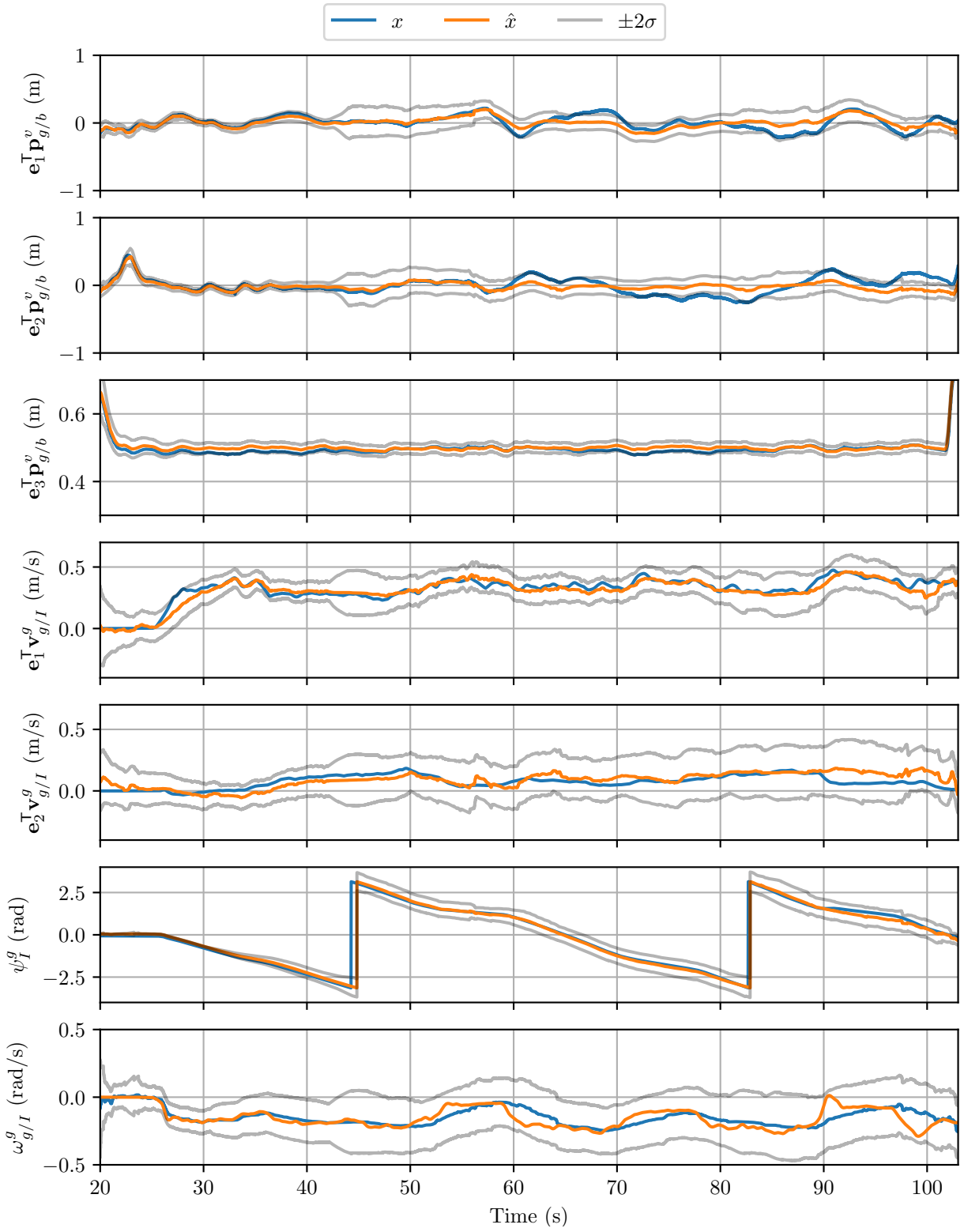

Figure 3.6: Hardware results when the ESKF estimated the positions of *ten* visual features. The blue line represents the true state while the orange line represents the estimated state. The two grey lines show $\pm 2\sigma$ bounds for the estimate based on the estimated covariance. Measurements from the fiducial marker were not used after $t = 30$ s to demonstrate the performance of the estimator.

CHAPTER 4. ERROR-STATE LQR CONTROL OF A MULTIROTOR UAV[1]

4.1 Introduction

Over the past two decades, multirotor unmanned air vehicles (UAVs) have become a popular platform for robotics research and the base for a variety of consumer and commercial products. UAVs are currently used all over the world for everything from military surveillance to package delivery. Larger multirotor vehicles have even been recently introduced to transport humans. Whatever the application, multirotors must but able to safely navigate in their environment, requiring a combination of complex perception, motion planning, and control algorithms. This paper describes a novel control algorithm that allows a multirotor UAV to accurately track a desired trajectory in time and space.

The Linear Quadratic Regulator (LQR) is a well-known feedback controller that computes the optimal feedback gains for a linear time-invariant (LTI) system given a quadratic cost function. LQR has been used to control multirotor UAVs with a variety of approaches. Almost all of these approaches linearize the system at a given stable state and use a fixed LQR gain [31]. Some have used a gain scheduling approach with a library of LQR gains for different magnitudes of deviation from the desired state [32]. Recently, an approach was proposed that relinearizes the system at a fixed rate, slower than the control loop, and then recomputes the LQR gains at that rate [33]. Our proposed solution takes a similar approach while relinearizing and recomputing the LQR gains at every control step.

Recently there has been a movement in the robotics community to appropriately deal with the evolution of a robot's state along a manifold using Lie theory [10]. Though these methods have widely been used in the field of state estimation [11, 12], a few methods have emerged that also apply Lie theory to control [34, 35]. We propose a formulation of the LQR problem that properly

[1]This chapter is a modified version of the paper published in the 2019 International Conference on Unmanned Aircraft Systems, written by Michael Farrell, James Jackson, Jerel Nielsen, Craig Bidstrup, and Tim McLain [30]. The coauthors assisted with the mathematical notation and with the trajectory generation section.

deals with the manifold nature of the state, specifically the attitude component. Most previous LQR solutions for a multirotor UAV use an Euler angle representation of attitude and treat the tuple of ZYX Euler angles as if it were a vector space [31], even though it is not [36]. While some methods use unit quaternions or rotation matrices to properly represent attitude, these are also not inherently a vector space and extra steps are required to orthonormalize or otherwise force the attitude to stay on the manifold [32, 33]. The proposed solution is derived from Lie theory and care is taken to ensure that all vector manipulations are done with appropriate vector quantities so that the state remains on the manifold.

Sec. 4.2 explains the multirotor UAV model used in the proposed controller formulation. Sec. 4.3 presents traditional LQR theory and shows how the proposed LQR formulation is a natural extension when care is taken to apply Lie theory to the multirotor problem. Sec. 4.4 describes the experiments used to demonstrate the proposed control scheme both in simulation and in hardware. Sec. 4.5 discusses the results of the experiments and Sec. 4.6 provides concluding remarks.

4.2 Model

4.2.1 Notation

We define some common notation used throughout the paper, first noting that vectors are represented with a bold letter (e.g., $\mathbf{v}$) and matrices with a capital letter (e.g., A).

R_a^b	Rotation matrix from reference frame a to b
$\mathbf{v}_{a/b}^c$	Vector state $\mathbf{v}$ of frame a w.r.t. frame b, expressed in frame c
$\check{a}$	Desired value of a
$\dot{a}$	Time derivative of a
$\tilde{a}$	Error of variable a, i.e., $\tilde{a} \triangleq a - \check{a}$

We also define the following coordinate frames:

I	The inertial coordinate frame in north-east-down
ℓ	The aircraft's vehicle-1 (body-level) coordinate frame
b	The aircraft's body-fixed coordinate frame

We make frequent use of the skew-symmetric matrix operator defined by

$$[\mathbf{v}]_\times \triangleq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{4.1}$$

which is related to the cross-product between two vectors as

$$\mathbf{v} \times \mathbf{w} = [\mathbf{v}]_\times \mathbf{w}. \tag{4.2}$$

We also use the standard basis vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N$, where $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T}$ and so forth.

4.2.2 Quaternion Representation

A quaternion $\mathbf{q}$ is a hyper-complex number of rank four. It is well known that a unit quaternion $\in S^3$ can be used to efficiently represent attitude, as S^3 is a double cover of $SO(3)$. Quaternions have the advantage over $SO(3)$ of being more efficient to implement on modern hardware [37], therefore in the software implementation of the described algorithm, we use quaternions, rather than rotation matrices.

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

$$\mathbf{q} = q_0 + q_x i + q_y j + q_z k \tag{4.3}$$

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

$$\begin{aligned} ij &= -ji &&= k, \\ jk &= -kj &&= i, \\ ki &= -ik &&= j, \\ i^2 = j^2 = k^2 &= ijk &&= -1. \end{aligned} \tag{4.4}$$

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

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\mathsf{T} \tag{4.5}$$

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

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{\mathbf{q}} \end{pmatrix}. \tag{4.6}$$

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

$$\mathbf{q}^a \otimes \mathbf{q}^b = \begin{pmatrix} q_0^a & (-\bar{\mathbf{q}}^a)^\mathsf{T} \\ \bar{\mathbf{q}}^a & q_0^a I + [\bar{\mathbf{q}}^a]_\times \end{pmatrix} \begin{pmatrix} q^b \\ \bar{\mathbf{q}}^b \end{pmatrix}. \tag{4.7}$$

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

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0 [\bar{\mathbf{q}}]_\times + 2\bar{\mathbf{q}}\bar{\mathbf{q}}^\top \in SO(3). \tag{4.8}$$

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

$$\begin{aligned} \exp_{\mathbf{q}} &: \mathbb{R}^3 \to S^3 \\ \exp_{\mathbf{q}}(\delta) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\delta\|}{2}\right) \\ \sin\left(\frac{\|\delta\|}{2}\right) \frac{\delta}{\|\delta\|} \end{bmatrix}, \end{aligned} \tag{4.9}$$

with the corresponding logarithmic map defined as

$$\begin{aligned} \log_{\mathbf{q}} &: S^3 \to \mathbb{R}^3 \\ \log_{\mathbf{q}}(\mathbf{q}) &\triangleq 2\,\mathrm{atan2}\left(\|\bar{\mathbf{q}}\|, q_0\right) \frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}. \end{aligned} \tag{4.10}$$

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

$$\exp_{\mathbf{q}}(\delta) \approx \begin{bmatrix} 1 \\ \frac{\delta}{2} \end{bmatrix} \tag{4.11}$$

$$\log_{\mathbf{q}}(\mathbf{q}) \approx 2\,\mathrm{sign}(q_0)\,\bar{\mathbf{q}}. \tag{4.12}$$

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

$$\mathbf{r}^b = R_a^b \mathbf{r}^a. \tag{4.13}$$

4.2.3 Quadrotor Dynamics

If we define the state of a quadrotor as the tuple of position, velocity, and attitude

$$\mathbf{x} = \left(\mathbf{p}_{b/I}^I, \mathbf{v}_{b/I}^b, \mathbf{q}_I^b\right) \in \mathbb{R}^3 \times \mathbb{R}^3 \times S^3$$

and the input to our system as the tuple of the throttle signal, s, and angular velocity, $\boldsymbol{\omega}_{b/I}^b$,

$$\mathbf{u} = \left(s, \boldsymbol{\omega}_{b/i}^b\right) \in \mathbb{R}^1 \times \mathbb{R}^3,$$

then the rigid body dynamics of a multirotor UAV are as follows [38]:

$$\begin{aligned} \dot{\mathbf{p}}_{b/I}^I &= \left(R_I^b\right)^{\mathsf{T}} \mathbf{v}_{b/I}^b \\ \dot{\mathbf{v}}_{b/I}^b &= g R_I^b \mathbf{e}_3 - g\frac{s}{s_e}\mathbf{e}_3 - c_d\left(I - \mathbf{e}_3\mathbf{e}_3^\top\right)\mathbf{v}_{b/I}^b - \left[\boldsymbol{\omega}_{b/I}^b\right]_\times \mathbf{v}_{b/I}^b \\ \dot{\mathbf{q}}_I^b &= \mathbf{q}_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}_{b/I}^b \end{pmatrix}, \end{aligned} \tag{4.14}$$

where c_d is a linear drag constant, s_e is the throttle command required to hover, and g is the magnitude of gravity. This model assumes a linear relationship between throttle signal and thrust,

which is not always the case. Although we use this simple model, more sophisticated approaches, such as [39] estimate this relationship online and compensate for it in real time.

4.2.4 Error-State Dynamics

It is useful to consider what is known as the *error state* of the quadrotor. This concept has a long history in state estimation and is used in the error-state Kalman filter [11, 40]. The error state is used in state estimation as a principled way to represent the covariance about attitude in terms of a vector space, as opposed to some local approximation. This relationship is also useful in control for the same reason. Performing control in the vector space of error state provides a principled way to leverage well-understood and efficient linear algebra machinery to solve control problems over non-vector quantities, such as attitude.

We define the error state of some quantity $\mathbf{y}$ as

$$\tilde{\mathbf{y}} = \mathbf{y} \boxminus \check{\mathbf{y}}, \tag{4.15}$$

where $\boxminus$ is an appropriate difference operator, as described by [41]. For instance, if $\mathbf{y}$, $\check{\mathbf{y}} \in \mathbb{R}^n$, the $\boxminus$ operator may be defined as the vector subtraction operator. However, due to the attitude component of our state, the vector subtraction operator is not defined between $\mathbf{x}$ and $\check{\mathbf{x}}$. We instead define the error state piecewise for each component of the state and combine these into an error-state vector

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{p}}^I_{b/I} & \tilde{\mathbf{v}}^b_{b/I} & \tilde{\mathbf{r}}^b_I \end{bmatrix}^\top \in \mathbb{R}^{9\times 1}, \tag{4.16}$$

where $\tilde{\mathbf{p}}^I_{b/I}$ is the error state associated with position, $\tilde{\mathbf{v}}^b_{b/I}$ is the error state associated with velocity, and $\tilde{\mathbf{r}}^b_I$ is the error state associated with attitude.

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

$$\tilde{\mathbf{p}}^I_{b/I} = \mathbf{p}^I_{b/I} - \check{\mathbf{p}}^I_{b/I} \tag{4.17}$$

$$\tilde{\mathbf{v}}^b_{b/I} = \mathbf{v}^b_{b/I} - \check{\mathbf{v}}^b_{b/I}, \tag{4.18}$$

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

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

$$\mathbf{r}_I^b(t) = \mathbf{r}_I^b(t_0) + \int_{t_0}^{t} \boldsymbol{\omega}_{b/I}^b(\tau)\, d\tau, \tag{4.19}$$

such that $\mathbf{r}_I^b(t_0) = 0$ and $\dot{\mathbf{r}}_I^b = \boldsymbol{\omega}_{b/I}^b$. With this definition, we can use (4.9) and (4.10) to express

$$\mathbf{q}_I^b = \check{\mathbf{q}}_I^b \otimes \exp_{\mathbf{q}}(\tilde{\mathbf{r}}) \tag{4.20}$$

$$\tilde{\mathbf{r}} = \log_{\mathbf{q}}\left(\left(\check{\mathbf{q}}_I^b\right)^{-1} \otimes \mathbf{q}_I^b\right), \tag{4.21}$$

as described by [41].

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

$$\dot{\tilde{\mathbf{r}}}_I^b = \dot{\mathbf{r}}_I^b - R_I^b\left(\check{R}_I^b\right)^\top \dot{\check{\mathbf{r}}}_I^b, \tag{4.22}$$

where $R_I^b\left(\check{R}_I^b\right)^\top$ moves the desired vector derivative, $\dot{\check{\mathbf{r}}}_I^b$, from its own tangent space to the tangent space of $\dot{\mathbf{r}}_I^b$. With both vectors in the same tangent space, the vector subtraction in (4.22) is valid.

For use in control, we similarly define an error state for the control input with the error state being the difference between the current control input and some reference input. Using the same definition as in (4.15), we can see that

$$\tilde{s} = s - \check{s} \tag{4.23}$$

$$\tilde{\boldsymbol{\omega}}_{b/I}^b = \boldsymbol{\omega}_{b/I}^b - \check{\boldsymbol{\omega}}_{b/I}^b \tag{4.24}$$

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

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

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}}^I_{b/I} &= \left(R^b_I\right)^\mathsf{T} \tilde{\mathbf{v}}^b_{b/I} - \left(R^b_I\right)^\mathsf{T} \left[\mathbf{v}^b_{b/I}\right]_\times \tilde{\mathbf{r}}^b_I \\
\dot{\tilde{\mathbf{v}}}^b_{b/I} &= g\left[R^b_I \mathbf{e}_3\right]_\times \tilde{\mathbf{r}}^b_I - g\frac{\tilde{s}}{s_e}\mathbf{e}_3 - c_d\left(I - \mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I} - \left[\omega^b_{b/I}\right]_\times \tilde{\mathbf{v}}^b_{b/I} + \left[\mathbf{v}^b_{b/I}\right]_\times \tilde{\omega}^b_{b/I} \\
\dot{\tilde{\mathbf{r}}}^b_I &= \tilde{\omega}^b_{b/I} - \left[\omega^b_{b/I}\right]_\times \tilde{\mathbf{r}}^b_I,
\end{aligned} \tag{4.25}$$

or succinctly,

$$\dot{\tilde{\mathbf{x}}} = f\left(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}\right). \tag{4.26}$$

The derivation of these error-state dynamics can be found in Appendix C.

4.3 LQR Control

4.3.1 Traditional LQR

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

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{4.27}$$

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

$$J(\mathbf{x}, \mathbf{u}) = \int_0^\infty \left(\mathbf{x}^\top Q\mathbf{x} + \mathbf{u}^\top R\mathbf{u}\right) dt \tag{4.28}$$

with Q and R being positive definite matrices that define the costs associated with the state and the input. The cost function given in (4.28) is minimized by the control input

$$\mathbf{u} = -K\mathbf{x}, \tag{4.29}$$

where K is given by

$$K = R^{-1}B^{\top}P, \tag{4.30}$$

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

$$A^{\top}P + PA - PBR^{-1}B^{\top}P + Q = 0. \tag{4.31}$$

It should be noted that in its basic form, an LQR controller is simply a regulator and the control input $\mathbf{u}$ will only attempt to drive the state to zero in an optimal way. If the desire is for the system to reach a desired state, $\check{\mathbf{x}}$, one can start by defining the error-state as

$$\tilde{\mathbf{x}} = \mathbf{x} - \check{\mathbf{x}} \tag{4.32}$$

and redefining the control input as

$$\tilde{\mathbf{u}} = -K\tilde{\mathbf{x}}. \tag{4.33}$$

This technique, however, will generally result in steady-state error between the state $\mathbf{x}$ and the reference trajectory $\check{\mathbf{x}}$. The steady-state error can be removed by augmenting the state with an integrator or by applying a model-based feed-forward control input,

$$\mathbf{u} = \tilde{\mathbf{u}} + \check{\mathbf{u}} = -K\tilde{\mathbf{x}} + \check{\mathbf{u}}. \tag{4.34}$$

A direct application of (4.32) in our case is not defined because the multirotor state is not a vector quantity. To compensate for this, we propose to compute control based on the error-state dynamics of the system, where the error-state is purely a vector quantity.

4.3.2 Error-State LQR

We can apply the same LQR approach to the error-state dynamics from (4.25). Since LQR control is a regulator, it will drive the error-state to zero, or our current state to our desired state. Since LQR is only defined for an LTI system, we can approximate the error-state system as an LTI system by linearizing about the current state at each time step. This gives us the system

$$\dot{\tilde{\mathbf{x}}} = A\tilde{\mathbf{x}} + B\tilde{\mathbf{u}} \tag{4.35}$$

with the matrices A and B given by

$$A(\mathbf{x},\mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{x}}} f(\mathbf{x},\tilde{\mathbf{x}},\mathbf{u},\tilde{\mathbf{u}}) \tag{4.36}$$

$$B(\mathbf{x},\mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{u}}} f(\mathbf{x},\tilde{\mathbf{x}},\mathbf{u},\tilde{\mathbf{u}}). \tag{4.37}$$

Using the error-state dynamics in (4.25) and dropping the subscripts and superscripts for compactness it can be seen that

$$A(\mathbf{x},\mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{x}}} f(\mathbf{x},\tilde{\mathbf{x}},\mathbf{u},\tilde{\mathbf{u}}) \tag{4.38}$$

$$= \begin{bmatrix} \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{v}}} & \frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{r}}} \\ \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{v}}} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{r}}} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\mathbf{r}}} \end{bmatrix} \tag{4.39}$$

with the individual components given by

$$\frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{v}}} = \left(R_I^b\right)^{\mathsf{T}} \tag{4.40}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{r}}} = -\left(R_I^b\right)^{\mathsf{T}} \left[\mathbf{v}_{b/I}^b\right]_\times \tag{4.41}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{v}}} = -c_d \left(I - \mathbf{e}_3\mathbf{e}_3^{\mathsf{T}}\right) - \left[\omega_{b/I}^b\right]_\times \tag{4.42}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{r}}} = \left[gR_I^b\mathbf{e}_3\right]_\times \tag{4.43}$$

$$\frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\mathbf{r}}} = -\left[\omega_{b/I}^b\right]_\times. \tag{4.44}$$

It can similarly be seen that

$$B(\mathbf{x},\mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{u}}} f(\mathbf{x},\tilde{\mathbf{x}},\mathbf{u},\tilde{\mathbf{u}}) \tag{4.45}$$

$$= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{s}} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\omega}} \\ \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\omega}} \end{bmatrix} \tag{4.46}$$

with the individual components given by

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{s}} = -\frac{g}{s_e}\mathbf{e}_3 \tag{4.47}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\omega}} = \left[\mathbf{v}^b_{b/I}\right]_\times \tag{4.48}$$

$$\frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\omega}} = I_{3\times 3}. \tag{4.49}$$

By linearizing at every time step, the CARE must be solved at each time step with the current A and B matrices. We use the closed-form, Schur decomposition method described in [42]. This method allows us to relinearize and recompute the optimal control at full rate in our experiments.

Although we relinearize and solve the CARE at each time step, the Q and R weighting matrices are fixed. We choose these gains based on Bryson's rule [43]. In addition, we have found that better results are achieved by saturating the error-state in accordance with the maximum error terms used to choose the gains with Bryson's rule.

4.4 Experiment

To test the proposed error-state LQR controller, we designed two experiments to be performed in simulation and hardware: (i) tracking step inputs in the desired position of the UAV and (ii) tracking time-dependent full-state trajectories. We first explain how we generate these time-dependent trajectories for the experiments and then detail the experimental setup for both simulation and hardware.

4.4.1 Trajectory Generation

One important consideration in high-performance control of quadrotors is the generation of smooth, feasible trajectories. The quadrotor has the benefit of being *differentially flat* which means that the required inputs to the quadrotor can be fully defined using derivatives of the outputs of the system, the desired position and heading [44]. If we are given some smooth, differentiable trajectory of our desired position and heading then we can compute the full state and required inputs as a function of time

$$\begin{pmatrix} \check{\mathbf{p}}^I_{b/I}(t) \\ \check{\mathbf{q}}^b_I(t) \\ \check{\mathbf{v}}^I_{b/I}(t) \\ \check{s}(t) \\ \check{\omega}^I_{b/I}(t) \end{pmatrix} = \check{f} \begin{pmatrix} \check{\mathbf{p}}^I_{b/I}(t) \\ \dot{\check{\mathbf{p}}}^I_{b/I}(t) \\ \ddot{\check{\mathbf{p}}}^I_{b/I}(t) \\ \check{\psi}^I_{b/I}(t) \end{pmatrix}. \tag{4.50}$$

We derive the general case where the desired yaw angle of the multirotor UAV is a function of time. However, since it is well known that the yaw angle of a multirotor UAV is easily controllable independent of the other states [44], we simply command a constant zero yaw in our experiments.

We now derive the differentially flat outputs. First, desired position is given to us directly

$$\check{\mathbf{p}}^I_{b/I} = \check{\mathbf{p}}^I_{b/I}. \tag{4.51}$$

To derive desired attitude and throttle signal we start by applying Newton's second law, and consider the rotation from the heading-rotated, body-level coordinate frame, ℓ, to the body frame, b. Note that to avoid the need for an iterative solution, we neglect the forces due to drag that are accounted for in the quadrotor dynamics in (4.14). Newton's second law is given by

$$\sum F^I = m\ddot{\check{\mathbf{p}}}^I_{b/I} \tag{4.52}$$

$$-T\left(\check{R}^b_\ell\right)^{\mathsf{T}} + mg\mathbf{e}_3 = m\ddot{\check{\mathbf{p}}}^I_{b/I} \tag{4.53}$$

$$\frac{T}{m}\left(\check{R}^b_\ell\right)^{\mathsf{T}}\mathbf{e}_3 = g\mathbf{e}_3 - \ddot{\check{\mathbf{p}}}^I_{b/I}. \tag{4.54}$$

If we then define $\check{\mathbf{a}} = g\mathbf{e}_3 - \ddot{\check{\mathbf{p}}}^I_{b/I}$, then we get the following expression

$$\frac{T}{m}\left(\check{R}^b_\ell\right)^\mathsf{T}\mathbf{e}_3 = \check{\mathbf{a}} \tag{4.55}$$

where T and $\check{R}^b_\ell$ must satisfy the following conditions:

$$\frac{T}{m} = \|\check{\mathbf{a}}\| \tag{4.56}$$

$$\check{R}^I_\ell\mathbf{e}_3 = \frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}. \tag{4.57}$$

Because $T = \frac{s}{s_e}gm$, then

$$\check{s} = \frac{s_e}{g}\|\check{\mathbf{a}}\| \tag{4.58}$$

and (4.57) can be solved with

$$\check{\mathbf{q}}^b_\ell = \exp_{\mathbf{q}}(\theta\delta) \tag{4.59}$$

where

$$\theta = \cos^{-1}\left(\mathbf{e}_3^\mathsf{T}\frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}\right) \tag{4.60}$$

$$\delta = [\mathbf{e}_3]_\times\frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}. \tag{4.61}$$

The heading portion of attitude can now be applied to give us our full desired attitude

$$\check{\mathbf{q}}^b_I = \exp_{\mathbf{q}}(\check{\psi}\mathbf{e}_3)\otimes\check{\mathbf{q}}^b_\ell. \tag{4.62}$$

Desired velocity can be found using the desired attitude

$$\check{\mathbf{v}}^b_{b/I} = \check{R}^b_I\check{\dot{\mathbf{p}}}^I_{b/I}, \tag{4.63}$$

and the required angular rate is found by taking the time derivative of our desired attitude

$$\check{\omega}^b_{b/I} = \frac{d}{dt}\check{\mathbf{q}}^b_I. \tag{4.64}$$

In our implementation, we do this numerically with central differencing on the manifold.

In summary, the differentially flat output of a quadrotor is given as follows:

$$\check{\mathbf{p}}^I_{b/I} = \check{\mathbf{p}}^I_{b/I} \tag{4.65}$$

$$\check{\mathbf{q}}^b_I = \exp_{\mathbf{q}}(\check{\psi}\mathbf{e}_3) \otimes \check{\mathbf{q}}^b_\ell \tag{4.66}$$

$$\check{\mathbf{v}}^b_{b/I} = \check{R}^b_I \check{\dot{\mathbf{p}}}^I_{b/I} \tag{4.67}$$

$$\check{s} = \frac{s_e}{g}\left\| g\mathbf{e}_3 - \check{\ddot{\mathbf{p}}}^I_{b/I} \right\| \tag{4.68}$$

$$\check{\omega}^b_{b/I} = \frac{d}{dt}\check{\mathbf{q}}^b_I. \tag{4.69}$$

The examples in this work all reference the same figure-eight trajectory defined by

$$\check{\mathbf{p}}^I_{b/I}(t) = \mathbf{p}^I_{b/I}(t_0) + \begin{bmatrix} \delta_x \sin\left(\frac{T}{2\pi}t\right) \\ \delta_y \sin\left(\frac{T}{\pi}t\right) \\ \delta_z \sin\left(\frac{T}{2\pi}t\right) \end{bmatrix} \tag{4.70}$$

$$\check{\psi}^I_{b/I}(t) = 0 \tag{4.71}$$

where the $\delta_{(\cdot)}$ parameters define the dimensions of the trajectory, and T defines the period. While a trajectory defined by periodic functions is useful for simple demonstrations such as what we perform in this work, we direct the reader to more sophisticated methods of differentiable trajectory generation such as [44] for practical application.

4.4.2 Simulation

For simulation, we used Gazebo[2] and ROS[3] with the ROSflight software-in-the-loop (SIL) simulation [13]. This simulation setup allowed us to test the exact code that also ran in hardware.

[2]Gazebo: `www.gazebosim.org`

[3]Robot Operating System: `www.ros.org`

4.4.3 Hardware

A custom multirotor UAV built on a DJI 450 Flamewheel frame with an STM32F1 microcontroller running the ROSflight [13] flight control firmware was flown for the hardware experiments. The algorithm was implemented and run in real time onboard on an NVIDIA Jetson TX2. Though the TX2 has a GPU, all computation was done using only the ARM CPU, showing that this algorithm can also run at full rate on a variety of popular onboard computers. The multirotor UAV was flown in a small motion capture room with feedback from an OptiTrack[4] motion tracking system. The global position and attitude measurements from the motion capture system were fused in real time with the onboard IMU of the UAV using an extended Kalman filter (EKF) to produce full state estimates.

For added safety, the computed control inputs were saturated before being sent to the flight controller. The throttle signal, s, was saturated to a maximum value of 0.85 and the angular rate commands, $\omega^b_{b/I}$, were saturated such that each component $|\omega| \leq 2\,{}^{\text{rad}}/_{\text{s}}$.

4.5 Results

4.5.1 Simulation

Figs. 4.1 and 4.2 show simulation results of the controller following desired position step inputs (waypoints) and a figure-eight trajectory. The multirotor began the simulations at rest on the ground and converged to the desired trajectory within a few seconds. In Fig. 4.1, we see that the simulated UAV significantly overshot the initial desired altitude of 5 m, but relatively smoothly reached the following step inputs. These results are noteworthy as the desired step inputs caused the error state of the system to be large, breaking a key assumption of the controller.

In Fig. 4.2, we see that the simulated UAV smoothly converged to the desired trajectory. After convergence, the figure-eight trajectory tracking was near perfect even though the feed forward inputs computed from (4.65)– (4.69) did not account for the force due to drag and the controller did not model thrust dynamics nor torque dynamics.

[4]OptiTrack: `www.optitrack.com`

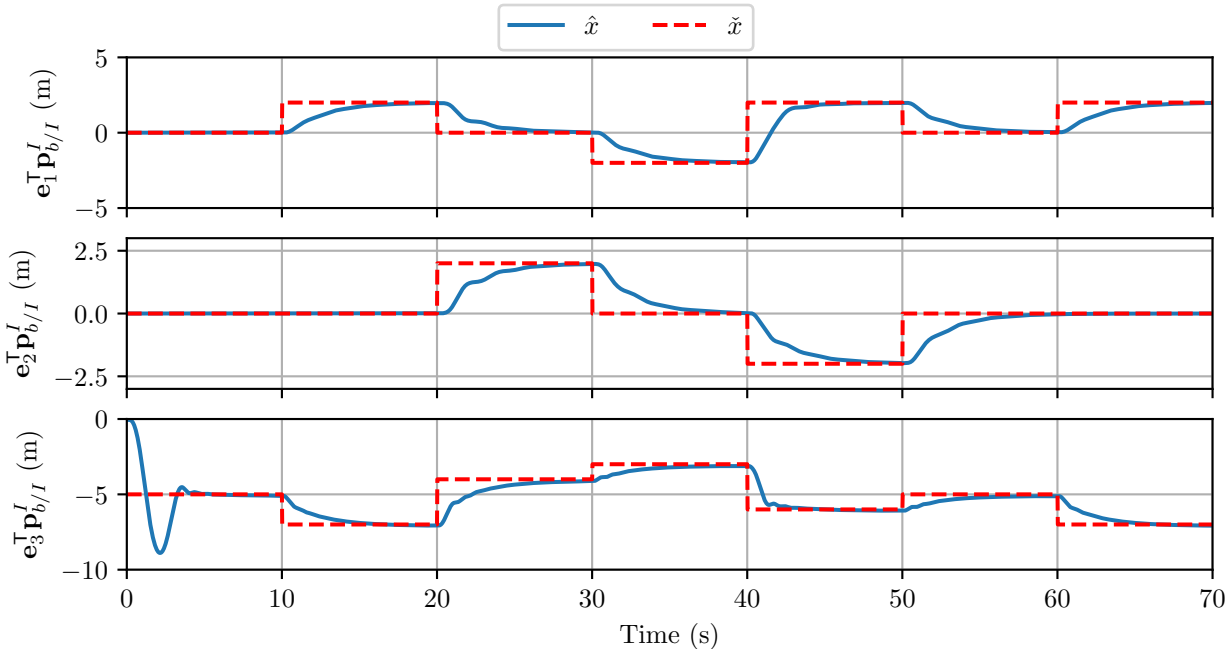

Figure 4.1: Simulation results for the position of the multirotor UAV given step inputs in position. The red dotted line is the desired position and the blue solid line is the estimated position.

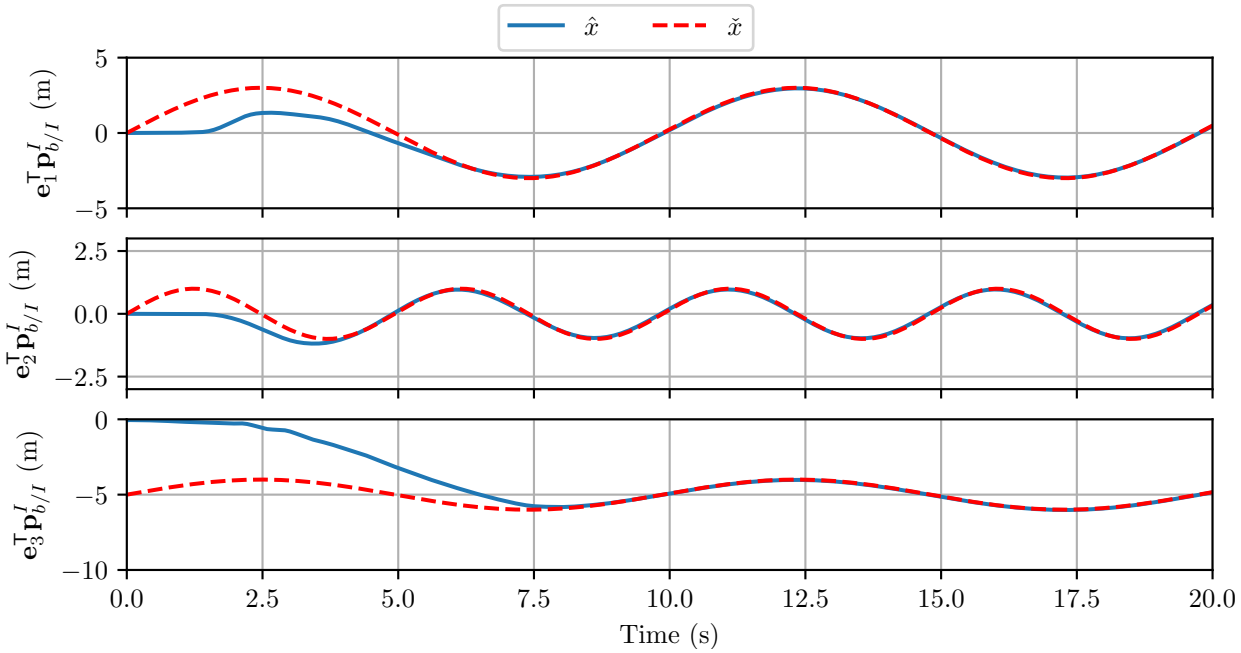

Figure 4.2: Simulation results of a multirotor UAV tracking a figure eight trajectory. The red dotted line is the desired position and the blue solid line is the estimated position.

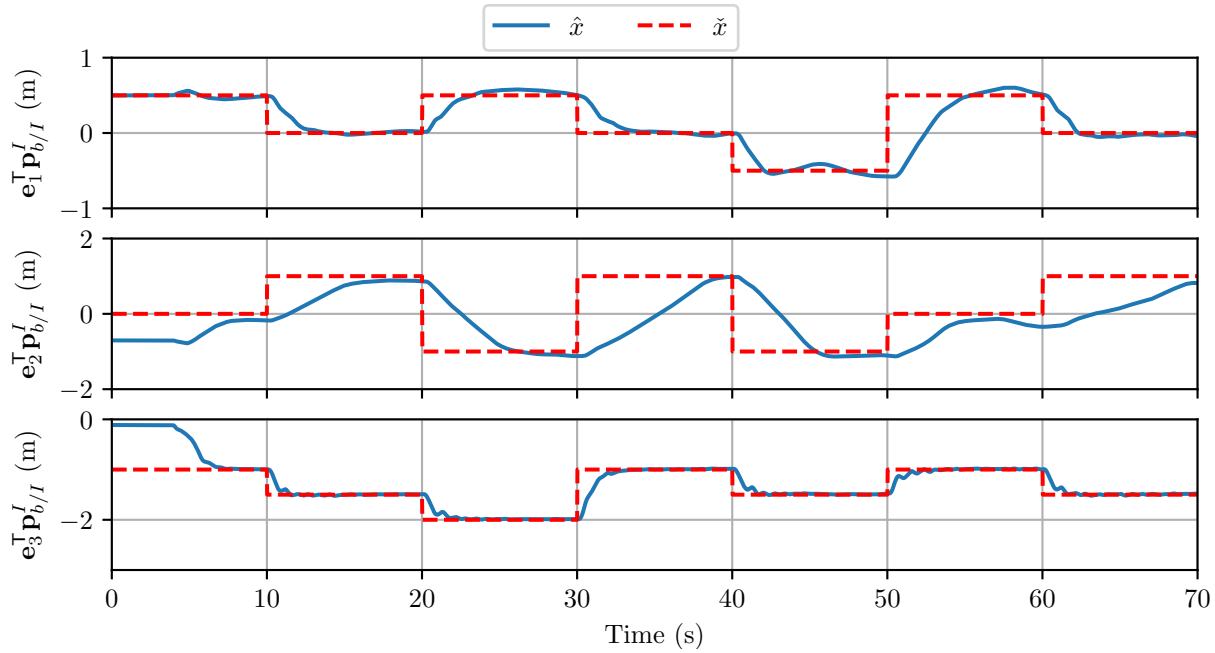

Figure 4.3: Hardware results for the position of the multirotor UAV given step inputs in position. The red dotted line is the desired position and the blue solid line is the estimated position.

4.5.2 Hardware

During the hardware experiments, the entire control algorithm ran at the full streaming rate of the onboard IMU, which was set to 250 Hz. The computation time of the algorithm was shown to have a mean of 274.6 µs and a standard deviation of 43.84 µs. This shows that the proposed LQR formulation can run at full rate even on computationally constrained platforms.

Fig. 4.3 shows the multirotor position along with the commanded positions for a waypoint path. The clean step response with minimal overshoot is notable considering we did not hand tune the LQR weighting matrices beyond an initial value derived from Bryson's rule. Fig. 4.4 shows how well the multirotor was able to follow the figure-eight trajectory, and Fig. 4.5 depicts the same flight plotted in two dimensions. The major deviations visible in this plot depict the take-off and landing portions of the flight. Though the trajectory tracking was close, we hypothesize that the visible error resulted from unmodeled dynamics of the system. To minimize these errors, the internal gains of the flight controller could potentially be tuned to produce quicker and more accurate tracking of the commanded angular rates.

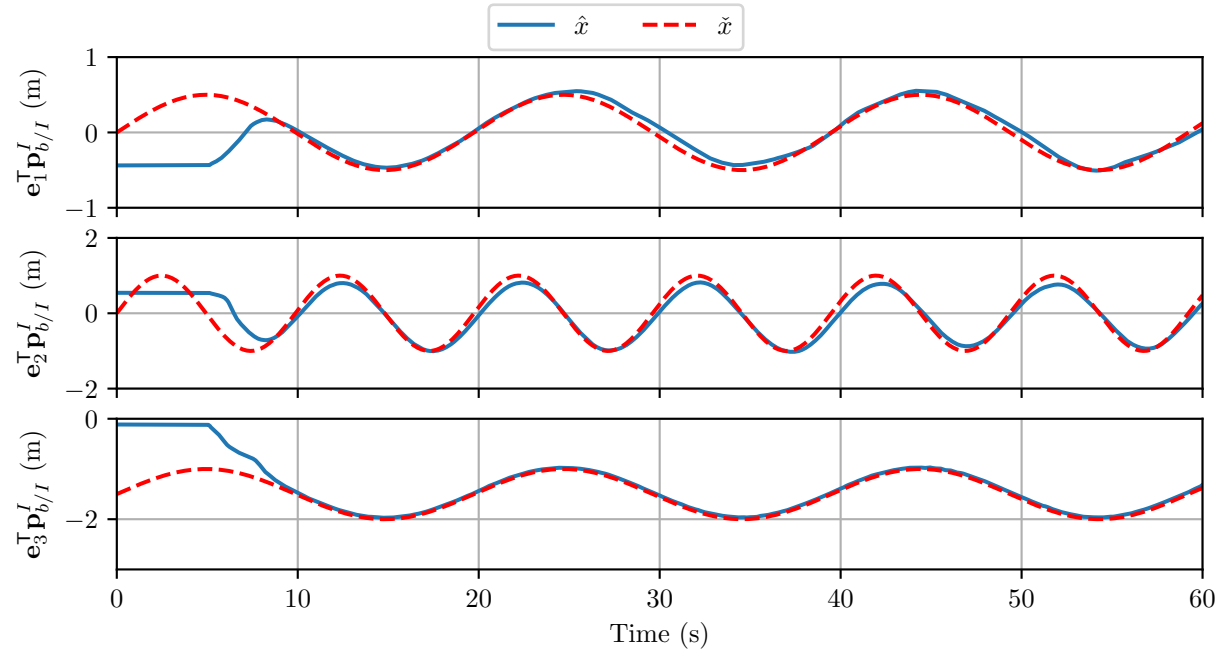

Figure 4.4: Hardware results of a multirotor UAV tracking a figure eight trajectory. The red dotted line is the desired position and the blue solid line is the estimated position.

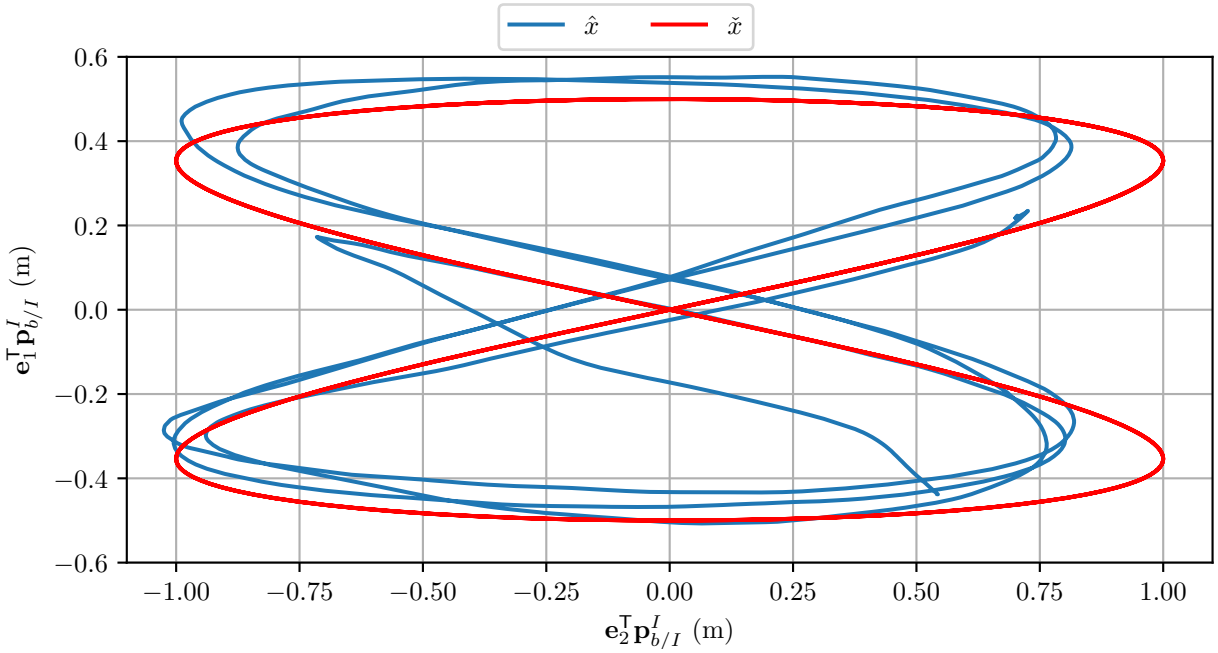

Figure 4.5: Top-down view of a multirotor UAV tracking a figure eight trajectory in hardware. The major deviation in position indicates the time during take-off when the multirotor must get to the trajectory before following it. The red solid line is the desired position and the blue solid line is the estimated position.

4.6 Conclusion

In this work we propose a novel LQR formulation derived from Lie theory. We show that by using the error-state dynamics, we can properly compute control using standard linear algebra techniques on vector quantities. Our implementation is efficient enough to relinearize the system and update the LQR gains in less than one millisecond. Simulation and hardware experiments show the effectiveness and simplicity of this control approach.

CHAPTER 5. CONCLUSION

The robust autonomous operation of multirotor UAVs from moving-vehicle base stations could benefit a variety of new industries. In an effort to make this a possibility, this thesis presents improvements to state estimation and control methods used by UAVs during landing on moving vehicles. The estimation algorithm proposed in Chapter 3 allows a UAV to continue to operate reliably with respect to a target vehicle even when a fiducial landing marker is not detected for significant periods of time. The control algorithm proposed in Chapter 4 provides a principled and computationally efficient method for a UAV to track a feasible, time-dependent trajectory. Simulation and hardware tests demonstrated the accurate and robust performance of these two algorithms. These results suggest that a reliable system for UAVs landing on moving vehicles could be developed by combining these proposed state estimation and control algorithms with an adequate motion planning algorithm.

5.1 Future Work

While the research in this thesis presents improvements to state-of-the-art methods, there remain many opportunities for future improvements. The following subsections outline recommendations and directions for future work related to the autonomous landing of multirotor UAVs on moving vehicles. These recommendations are divided into the specific fields of state estimation, motion planning, and control.

5.1.1 State Estimation

The estimation algorithm presented in Chapter 3 details a method of tracking and estimating the locations of visual features to aid in estimation when a fiducial landing marker is not detected. The results of this work were quite remarkable, showing that a UAV can continue to operate accurately with respect to a target vehicle without detecting a fiducial marker for more than

a minute. However, as mentioned in Sec. 3.3, this estimation algorithm assumes that all tracked features are rigidly attached to the target vehicle. Therefore, before the presented estimation algorithm can be used in a commercial product, a feature tracker must be developed that can reliably filter out visual features that are not part of the target vehicle. Recently, there has been significant research done in the fields of image and video segmentation that could be leveraged for such a task [45]. Alternative methods that depend on mathematical models of rigid body motion could also be explored as a possible solution to this problem.

While the proposed estimation method is easily extensible to a variety of target vehicles, experiments to test this algorithm were only conducted with a simple ground vehicle. In practice, many use cases require UAVs to land on vehicles with much more complex motion models such as trucks driving on mountainous terrain or boats on rough seas. Further experiments should be conducted to test the performance of the presented estimation algorithm with target vehicles of complex motion models. To achieve better results in these scenarios, additional measurement updates from sensors mounted to the target vehicle, such as GPS or IMU, could be added to the algorithm.

5.1.2 Motion Planning

While not discussed in great detail in this thesis, the motion planning problem associated with multirotor UAVs operating from moving vehicles is challenging. Dynamically feasible trajectories must be planned for UAVs to avoid obstacles during their approach to the landing target. This problem is especially important when a UAV must land on a large boat with significant superstructure or when a UAV must re-enter and land inside of a package delivery truck. While planning trajectories for UAVs with respect to static obstacles has been widely researched, planning trajectories to avoid dynamic obstacles that are rigidly attached to the desired landing target is its own unique problem to be solved.

When operating from vehicles with complex motion, it can also be important to precisely time the touchdown of the UAV. For instance, when landing on a boat in rough waters, it may be desireable that the UAV touches down when the heave motion of the boat reaches a peak to minimize potential damage to the UAV. Further research should be conducted to plan feasible, time-dependent trajectories for these scenarios.

5.1.3 Control

As discussed in Chapter 4, the Lie-theory-based controller operates on the error state of the system. The main advantage to this approach is its mathematically principled nature. This, in theory, allows for more accurate tracking during extreme maneuvers, where less principled methods, such as typical Euler-angle formulations, struggle. This controller should be tested with feasible trajectories of acrobatic maneuvers to demonstrate this advantage.

One key assumption made by the controller is that the error-state of the system is always small. This was especially not true in the flight experiments during takeoff, when the UAV was far from the desired sinusoidal trajectory. This resulted in unsteady transient behavior when the error-state values were not saturated to be small. Instead of a static trajectory, the controller should be tested with dynamically planned trajectories which begin at the current state of the UAV. With these trajectories, the error state of the system would be forced to be small, satisfying this important assumption.

REFERENCES

[1] Planck Aerosystems, "Ground vehicle based drones," https://www.planckaero.com/truckdrone, accessed: 2019-11-14. 1

[2] J. S. Wynn and T. W. McLain, "Visual servoing for multirotor precision landing in daylight and after-dark conditions," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 1242–1248. 1, 10

[3] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny, "Autonomous landing of a quadcopter on a high-speed ground vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2378–2385, 2017. 2, 10

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 2, 10

[5] K. Ling, "Precision landing of a quadrotor UAV on a moving target using low-cost sensors," Master's thesis, University of Waterloo, 2014. 2, 10

[6] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016. 2, 10

[7] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization," in *1999 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1999, pp. 1656–1663. 3

[8] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018. 3, 11

[9] T. Baca, P. Stepan, V. Spurny, D. Hert, R. Penicka, M. Saska, J. Thomas, G. Loianno, and V. Kumar, "Autonomous landing on a moving vehicle with an unmanned aerial vehicle," *Journal of Field Robotics*, vol. 36, no. 5, pp. 874–891, 2019. 3, 10

[10] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018. 4, 43, 73

[11] J. Solà, "Quaternion kinematics for the error-state Kalman filter," *arXiv preprint arXiv:1711.02508*, 2017. 4, 43, 48

[12] D. P. Koch, D. O. Wheeler, R. W. Beard, T. W. McLain, and K. M. Brink, "Relative multiplicative extended Kalman filter for observable GPS-denied navigation," 2017, available at http://scholarsarchive.byu.edu/facpub/1963. 4, 16, 17, 43, 70

[13] J. Jackson, G. Ellingson, and T. McLain, "ROSflight: A lightweight, inexpensive MAV research and development tool," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 758–762. 6, 38, 56, 57

[14] Banggood, "OpenPilot CC3D revolution revo 10DOF STM32F4 flight controller staight pin for RC drone FPV racing multi rotor," https://www.banggood.com/OpenPilot-CC3D-Revolution-Revo-10DOF-STM32F4-Flight-Controller-Staight-Pin-p-1000068.html?cur_warehouse=CN, accessed: 2019-11-14. 8

[15] Connect Tech Inc., "Orbitty Carrier for NVIDIA Jetson TX2/TX2i/TX1," http://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/, accessed: 2019-11-14. 9

[16] Amazon, "ELP USB with camera 2.1mm lens 1080p HD free driver USB camera module,2.0 megapixel(1080p) USB camera,for Linux Windows Android Mac OS," https://www.amazon.com/ELP-Camera-Megapixel-Windows-Android/dp/B00KA7WSSU, accessed: 2019-11-14. 9

[17] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 221–238, 2011. 10

[18] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 3400–3407. 10

[19] P. Marantos, G. C. Karras, P. Vlantis, and K. J. Kyriakopoulos, "Vision-based autonomous landing control for unmanned helicopters," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 1, pp. 145–158, 2018. 10

[20] O. Araar, N. Aouf, and I. Vitanov, "Vision based autonomous landing of multirotor UAV on moving platform," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 369–384, 2017. 10

[21] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 971–976. 10

[22] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013. 11

[23] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572. 11

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. 11

[25] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 200–207. 11

[26] K. M. Brink, "Partial-update Schmidt–Kalman filter," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2214–2228, 2017. 23, 24

[27] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, 2018. 39

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443. 39

[29] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001. 39

[30] M. Farrell, J. Jackson, J. Nielsen, C. Bidstrup, and T. McLain, "Error-state LQR control of a multirotor UAV," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 704–711. 43

[31] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 4001–4008. 43, 44

[32] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, "LQR control for a quadrotor using unit quaternions: Modeling and simulation," in *Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on*. IEEE, 2013, pp. 172–178. 43, 44

[33] P. Foehn and D. Scaramuzza, "Onboard state dependent LQR for agile quadrotors," in *2018 IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 6566–6572. 43, 44

[34] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li, "High performance full attitude control of a quadrotor on SO (3)," in *2015 IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 1698–1703. 43

[35] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425. 43

[36] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006. 44

[37] R. T. Casey, M. Karpenko, R. Curry, and G. Elkaim, "Attitude representation and kinematic propagation for low-cost UAVs," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4615. 45

[38] R. Leishman, J. Macdonald, R. Beard, and T. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems Magazine*, vol. 34, no. 1, pp. 28–41, Feb 2014. 47

[39] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," *arXiv e-prints*, p. arXiv:1812.04755, Dec 2018. 48

[40] F. L. Markley, "Multiplicative vs. additive filtering for spacecraft attitude determination," *Dynamics and Control of Systems and Structures in Space*, no. 467-474, 2004. 48

[41] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013. 48, 49

[42] A. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Transactions on automatic control*, vol. 24, no. 6, pp. 913–921, 1979. 53

[43] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2018. 53

[44] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525. 54, 56

[45] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611 63

APPENDIX A. DERIVATION OF ERROR-STATE DYNAMICS FOR THE ERROR-STATE KALMAN FILTER

A.1 UAV Position

To derive the error-state dynamics for the UAV position state, we start by differentiating the error-state definition given in (3.27) with respect to time to yield

$$\dot{\tilde{\mathbf{p}}}^I_{b/I} = \dot{\mathbf{p}}^I_{b/I} - \dot{\hat{\mathbf{p}}}^I_{b/I}. \tag{A.1}$$

We then substitute in the corresponding dynamics from (3.35) and (3.38)

$$\dot{\tilde{\mathbf{p}}}^I_{b/I} = \left(R^b_I\right)^\mathsf{T} \mathbf{v}^b_{b/I} - \left(\hat{R}^b_I\right)^\mathsf{T} \hat{\mathbf{v}}^b_{b/I}. \tag{A.2}$$

We expand this equation using the approximation for $\left(R^b_I\right)^\mathsf{T}$ given in (3.34) and the error-state definition given in (3.26) resulting in

$$\dot{\tilde{\mathbf{p}}}^I_{b/I} \approx \left(\hat{R}^b_I\right)^\mathsf{T} \left(I + \left[\tilde{\theta}^b_I\right]_\times\right) \left(\hat{\mathbf{v}}^b_{b/I} + \tilde{\mathbf{v}}^b_{b/I}\right) - \left(\hat{R}^b_I\right)^\mathsf{T} \hat{\mathbf{v}}^b_{b/I} \tag{A.3}$$

$$\approx \left(\hat{R}^b_I\right)^\mathsf{T} \left(\hat{\mathbf{v}}^b_{b/I} + \tilde{\mathbf{v}}^b_{b/I} + \left[\tilde{\theta}^b_I\right]_\times \hat{\mathbf{v}}^b_{b/I} + \left[\tilde{\theta}^b_I\right]_\times \tilde{\mathbf{v}}^b_{b/I}\right) - \left(\hat{R}^b_I\right)^\mathsf{T} \hat{\mathbf{v}}^b_{b/I} \tag{A.4}$$

$$\approx \left(\hat{R}^b_I\right)^\mathsf{T} \tilde{\mathbf{v}}^b_{b/I} + \left(\hat{R}^b_I\right)^\mathsf{T} \left[\tilde{\theta}^b_I\right]_\times \hat{\mathbf{v}}^b_{b/I} + \left(\hat{R}^b_I\right)^\mathsf{T} \left[\tilde{\theta}^b_I\right]_\times \tilde{\mathbf{v}}^b_{b/I}. \tag{A.5}$$

We assume the error-state components to be small, neglecting the second-order terms to get the final expression

$$\dot{\tilde{\mathbf{p}}}^I_{b/I} \approx \left(\hat{R}^b_I\right)^\mathsf{T} \tilde{\mathbf{v}}^b_{b/I} - \left(\hat{R}^b_I\right)^\mathsf{T} \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tilde{\theta}^b_I. \tag{A.6}$$

A.2 UAV Attitude

To derive the error-state dynamics for the UAV attitude state, we follow [12], starting with (3.30) and letting $\tilde{\mathbf{q}}_I^b = \exp_{\mathbf{q}}\left(\tilde{\theta}_I^b\right)$ such that

$$\mathbf{q}_I^b = \hat{\mathbf{q}}_I^b \otimes \tilde{\mathbf{q}}_I^b. \tag{A.7}$$

Differentiating with respect to time results in

$$\dot{\mathbf{q}}_I^b = \dot{\hat{\mathbf{q}}}_I^b \otimes \tilde{\mathbf{q}}_I^b + \hat{\mathbf{q}}_I^b \otimes \dot{\tilde{\mathbf{q}}}_I^b \tag{A.8}$$

which we multiply all terms on the left by $\left(\hat{\mathbf{q}}_I^b\right)^{-1}$ and simplify to yield

$$\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\mathbf{q}}_I^b = \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\hat{\mathbf{q}}}_I^b \otimes \tilde{\mathbf{q}}_I^b + \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \hat{\mathbf{q}}_I^b \otimes \dot{\tilde{\mathbf{q}}}_I^b \tag{A.9}$$

$$\left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\mathbf{q}}_I^b = \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\hat{\mathbf{q}}}_I^b \otimes \tilde{\mathbf{q}}_I^b + \dot{\tilde{\mathbf{q}}}_I^b \tag{A.10}$$

We then rearrange the above equation and simplify using the true-state and estimated-state dynamics of the UAV attitude state given in (3.35) and (3.38) along with the error-state definition from (3.29):

$$\dot{\tilde{\mathbf{q}}}_I^b = \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\mathbf{q}}_I^b - \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \dot{\hat{\mathbf{q}}}_I^b \otimes \tilde{\mathbf{q}}_I^b \tag{A.11}$$

$$= \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \mathbf{q}_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right) \end{pmatrix} - \left(\hat{\mathbf{q}}_I^b\right)^{-1} \otimes \hat{\mathbf{q}}_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\left(\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right) \end{pmatrix} \otimes \tilde{\mathbf{q}}_I^b \tag{A.12}$$

$$= \frac{1}{2}\tilde{\mathbf{q}}_I^b \otimes \begin{pmatrix} 0 \\ \left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right) \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 0 \\ \left(\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right) \end{pmatrix} \otimes \tilde{\mathbf{q}}_I^b. \tag{A.13}$$

Applying (3.9) and (3.10) we can expand (A.13) as matrix-like products

$$\dot{\tilde{\mathbf{q}}}_I^b = \frac{1}{2}\begin{pmatrix} 0 & -\left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right)^\mathsf{T} \\ \left(\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right) & -\left[\bar{\omega}_{b/I}^b - \beta_\omega - \upsilon_\omega\right]_\times \end{pmatrix} \tilde{\mathbf{q}}_I^b \\ -\frac{1}{2}\begin{pmatrix} 0 & -\left(\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right)^\mathsf{T} \\ \left(\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right) & \left[\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right]_\times \end{pmatrix} \tilde{\mathbf{q}}_I^b. \tag{A.14}$$

We then simplify the previous equation using the error-state definition $\tilde{\beta}_\omega \triangleq \beta_\omega - \hat{\beta}_\omega$ giving

$$\dot{\tilde{\mathbf{q}}}_I^b = \frac{1}{2}\begin{pmatrix} 0 & -\left(-\tilde{\beta}_\omega - \upsilon_\omega\right)^\mathsf{T} \\ \left(-\tilde{\beta}_\omega - \upsilon_\omega\right) & \left[-2\bar{\omega}_{b/I}^b + 2\hat{\beta}_\omega + \tilde{\beta}_\omega + \upsilon_\omega\right]_\times \end{pmatrix} \tilde{\mathbf{q}}_I^b, \tag{A.15}$$

which implies that

$$\begin{pmatrix} 1 \\ \frac{1}{2}\dot{\tilde{\theta}}_I^b \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 0 & -\left(-\tilde{\beta}_\omega - \upsilon_\omega\right)^\mathsf{T} \\ \left(-\tilde{\beta}_\omega - \upsilon_\omega\right) & \left[-2\bar{\omega}_{b/I}^b + 2\hat{\beta}_\omega + \tilde{\beta}_\omega + \upsilon_\omega\right]_\times \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1}{2}\tilde{\theta}_I^b \end{pmatrix}. \tag{A.16}$$

To get the final expression, we drop the scalar equation and neglect second-order terms to yield

$$\dot{\tilde{\theta}}_I^b = \left(-\tilde{\beta}_\omega - \upsilon_\omega\right) + \frac{1}{2}\left[-2\bar{\omega}_{b/I}^b + 2\hat{\beta}_\omega + \tilde{\beta}_\omega + \upsilon_\omega\right]_\times \tilde{\theta}_I^b \tag{A.17}$$

$$\approx -\left[\bar{\omega}_{b/I}^b - \hat{\beta}_\omega\right]_\times \tilde{\theta}_I^b - \tilde{\beta}_\omega - \upsilon_\omega. \tag{A.18}$$

A.3 UAV Velocity

Similar to the approach for the UAV position state, to derive the error-state dynamics for the UAV velocity state, we begin with the time derivative of the error-state definition

$$\dot{\tilde{\mathbf{v}}}_{b/I}^I = \dot{\mathbf{v}}_{b/I}^I - \dot{\hat{\mathbf{v}}}_{b/I}^I. \tag{A.19}$$

We then substitute in the dynamics from (3.35) and (3.38) and expand using the error-state definitions and approximations

$$\begin{aligned}\dot{\tilde{\mathbf{v}}}^I_{b/I} =& R^b_I \mathbf{g}^I + \left[\mathbf{v}^b_{b/I}\right]_\times \left(\bar{\omega}^b_{b/I} - \beta_\omega - \upsilon_\omega\right) + \left(\bar{\mathbf{a}}^b_{b/I} - \beta_a - \upsilon_a\right) \qquad \text{(A.20)}\\ &- \left(\hat{R}^b_I \mathbf{g}^I + \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \left(\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right) + \left(\bar{\mathbf{a}}^b_{b/I} - \hat{\beta}_a\right)\right)\\ \approx& \left(I - \left[\tilde{\theta}^b_I\right]_\times\right) \hat{R}^b_I \mathbf{g}^I + \left[\hat{\mathbf{v}}^b_{b/I} + \tilde{\mathbf{v}}^b_{b/I}\right]_\times \left(\bar{\omega}^b_{b/I} - \hat{\beta}_\omega - \tilde{\beta}_\omega - \upsilon_\omega\right) \qquad \text{(A.21)}\\ &+ \left(\bar{\mathbf{a}}^b_{b/I} - \hat{\beta}_a - \tilde{\beta}_a - \upsilon_a\right) - \left(\hat{R}^b_I \mathbf{g}^I + \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \left(\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right) + \left(\bar{\mathbf{a}}^b_{b/I} - \hat{\beta}_a\right)\right),\end{aligned}$$

which we simplify and neglect high-order terms to yield the final expression:

$$\dot{\tilde{\mathbf{v}}}^I_{b/I} \approx \left[\hat{R}^b_I \mathbf{g}^I\right]_\times \tilde{\theta}^b_I - \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \tilde{\beta}_\omega - \left[\hat{\mathbf{v}}^b_{b/I}\right]_\times \upsilon_\omega - \left[\bar{\omega}^b_{b/I} - \hat{\beta}_\omega\right]_\times \tilde{\mathbf{v}}^b_{b/I} - \tilde{\beta}_a - \upsilon_a. \qquad \text{(A.22)}$$

A.4 Sensor Biases

To derive the error-state dynamics for the UAV bias states, we start with the time derivative of their error-state definitions

$$\dot{\tilde{\beta}}_a = \dot{\beta}_a - \dot{\hat{\beta}}_a \qquad \text{(A.23)}$$

$$\dot{\tilde{\beta}}_\omega = \dot{\beta}_\omega - \dot{\hat{\beta}}_\omega. \qquad \text{(A.24)}$$

We then substitute in the dynamics from (3.35) and (3.38) to yield the final expression

$$\dot{\tilde{\beta}}_a = \eta_{\beta_a} \qquad \text{(A.25)}$$

$$\dot{\tilde{\beta}}_\omega = \eta_{\beta_\omega}. \qquad \text{(A.26)}$$

A.5 Target Vehicle Position

Similar to the method used for the UAV states, to derive the error-state dynamics for the target vehicle position state, we start with the time derivative of the error-state definition

$$\dot{\tilde{\mathbf{p}}}^{v}_{g/b} = \dot{\mathbf{p}}^{v}_{g/b} - \dot{\hat{\mathbf{p}}}^{v}_{g/b}. \tag{A.27}$$

We then substitute in the dynamics from (3.36) and (3.38) and expand using the error-state definitions and approximations

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}}^{v}_{g/b} &= I_{3\times 2}\left(R^{g}_{I}\right)^{\mathsf{T}}\mathbf{v}^{g}_{g/I} - \left(R^{b}_{I}\right)^{\mathsf{T}}\mathbf{v}^{b}_{b/I} - \left(I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{g}_{g/I} - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{b}_{b/I}\right) && \text{(A.28)}\\
&\approx I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\left(R\left(\tilde{\psi}^{g}_{I}\right)\right)^{\mathsf{T}}\left(\hat{\mathbf{v}}^{g}_{g/I}+\tilde{\mathbf{v}}^{g}_{g/I}\right) - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\left(I+\left[\tilde{\theta}^{b}_{I}\right]_{\times}\right)\left(\hat{\mathbf{v}}^{b}_{b/I}+\tilde{\mathbf{v}}^{b}_{b/I}\right) && \text{(A.29)}\\
&\quad - \left(I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{g}_{g/I} - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{b}_{b/I}\right).
\end{aligned}$$

From [10], by assuming $\tilde{\psi}^{g}_{I}$ is small, we can approximate

$$\left(R\left(\tilde{\psi}^{g}_{I}\right)\right)^{\mathsf{T}} \approx I + \left[\tilde{\psi}^{g}_{I}\right]_{\times} \tag{A.30}$$

where the two-dimensional skew-symmetric matrix is given by

$$\left[\tilde{\psi}^{g}_{I}\right]_{\times} \triangleq \begin{bmatrix} 0 & -\tilde{\psi}^{g}_{I} \\ \tilde{\psi}^{g}_{I} & 0 \end{bmatrix}. \tag{A.31}$$

We then substitute (A.30) into (A.29) and simplify, neglecting higher-order terms to get the final expression

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}}^{v}_{g/b} &\approx I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\left(I+\left[\tilde{\psi}^{g}_{I}\right]_{\times}\right)\left(\hat{\mathbf{v}}^{g}_{g/I}+\tilde{\mathbf{v}}^{g}_{g/I}\right) - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\left(I+\left[\tilde{\theta}^{b}_{I}\right]_{\times}\right)\left(\hat{\mathbf{v}}^{b}_{b/I}+\tilde{\mathbf{v}}^{b}_{b/I}\right) && \text{(A.32)}\\
&\quad - \left(I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{g}_{g/I} - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\hat{\mathbf{v}}^{b}_{b/I}\right).\\
&\approx I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\tilde{\mathbf{v}}^{g}_{g/I} + I_{3\times 2}\left(\hat{R}^{g}_{I}\right)^{\mathsf{T}}\left[\tilde{\psi}^{g}_{I}\right]_{\times}\hat{\mathbf{v}}^{g}_{g/I} + \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\left[\hat{\mathbf{v}}^{b}_{b/I}\right]_{\times}\tilde{\theta}^{b}_{I} - \left(\hat{R}^{b}_{I}\right)^{\mathsf{T}}\tilde{\mathbf{v}}^{b}_{b/I}. && \text{(A.33)}
\end{aligned}$$

A.6 Target Vehicle Velocity

Similar to the UAV bias states, the error-state dynamics of the target vehicle velocity state are simply found by starting with the time derivative of the error-state definition

$$\dot{\tilde{\mathbf{v}}}^g_{g/I} = \dot{\mathbf{v}}^g_{g/I} - \dot{\hat{\mathbf{v}}}^g_{g/I} \tag{A.34}$$

and substituting in the dynamics from (3.36) and (3.38) to get the final expression

$$\dot{\tilde{\mathbf{v}}}^g_{g/I} = \eta_{gv}. \tag{A.35}$$

A.7 Target Vehicle Attitude

To derive the error-state dynamics for the target vehicle attitude state, we follow the method used for deriving the error-state dynamics for the previous vector states because we treat ψ^g_I as a vector space. We, therefore, start by taking the time derivative of the error-state definition given in (3.28)

$$\dot{\tilde{\psi}}^g_I = \dot{\psi}^g_I - \dot{\hat{\psi}}^g_I. \tag{A.36}$$

We then substitute in the dynamics from (3.36) and (3.38)

$$\dot{\tilde{\psi}}^g_I = \omega^I_{g/I} - \hat{\omega}^I_{g/I} \tag{A.37}$$

and simplify to get the final expression

$$\dot{\tilde{\psi}}^g_I = \tilde{\omega}^I_{g/I}. \tag{A.38}$$

A.8 Target Vehicle Angular Rate

The error-state dynamics of the target vehicle angular rate state are found by starting with the time derivative of the error-state definition

$$\dot{\tilde{\omega}}^g_{g/I} = \dot{\omega}^g_{g/I} - \dot{\hat{\omega}}^g_{g/I}. \tag{A.39}$$

We then substitute in the dynamics from (3.36) and (3.38) to yield the final expression

$$\dot{\tilde{\omega}}^g_{g/I} = \eta_{g\omega}. \tag{A.40}$$

A.9 Visual Feature Vector

Starting with the time derivative of the error-state definition for the visual feature vector states,

$$\dot{\tilde{\mathbf{r}}}^g_{i/g} = \dot{\mathbf{r}}^g_{i/g} - \dot{\hat{\mathbf{r}}}^g_{i/g}, \tag{A.41}$$

and substituting in the dynamics from (3.37) and (3.38) yields

$$\dot{\tilde{\mathbf{r}}}^g_{i/g} = \mathbf{0}. \tag{A.42}$$

APPENDIX B. DERIVATION OF VISUAL FEATURE MEASUREMENT MODEL FOR THE ERROR-STATE KALMAN FILTER

To derive the measurement residual Jacobian for the visual feature pixel measurements, we first start with the residual model from (3.121)

$$\mathbf{r}_{\text{pix}} = \frac{1}{\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c} I_{2\times 3} K \mathbf{p}_{i/c}^c + \eta_{\text{pix}} - \frac{1}{\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c} I_{2\times 3} K \hat{\mathbf{p}}_{i/c}^c. \tag{B.1}$$

We then note that

$$H_{\text{pix}} = \frac{\partial \mathbf{r}_{\text{pix}}}{\partial \tilde{\mathbf{x}}} \tag{B.2}$$

$$= \frac{\partial}{\partial \tilde{\mathbf{x}}}\left(\frac{1}{\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c} I_{2\times 3} K \mathbf{p}_{i/c}^c\right) + \frac{\partial}{\partial \tilde{\mathbf{x}}}\left(\eta_{\text{pix}}\right) - \frac{\partial}{\partial \tilde{\mathbf{x}}}\left(\frac{1}{\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c} I_{2\times 3} K \hat{\mathbf{p}}_{i/c}^c\right) \tag{B.3}$$

$$= \frac{\partial}{\partial \tilde{\mathbf{x}}}\left(\frac{1}{\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c} I_{2\times 3} K \mathbf{p}_{i/c}^c\right) + \mathbf{0} + \mathbf{0} \tag{B.4}$$

$$= \frac{1}{\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c} I_{2\times 3} K \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}_{i/c}^c - \frac{\mathbf{e}_3^\mathsf{T}\frac{\partial}{\partial \tilde{\mathbf{x}}}\mathbf{p}_{i/c}^c}{\left(\mathbf{e}_3^\mathsf{T}\mathbf{p}_{i/c}^c\right)^2} I_{2\times 3} K \mathbf{p}_{i/c}^c. \tag{B.5}$$

As $\mathbf{p}_{i/c}^c$ appears in (B.5), we substitute in our best possible estimate for this vector, $\hat{\mathbf{p}}_{i/c}^c$, such that

$$H_{\text{pix}} \approx \frac{1}{\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c} I_{2\times 3} K \frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}_{i/c}^c - \frac{\mathbf{e}_3^\mathsf{T}\frac{\partial}{\partial \tilde{\mathbf{x}}}\mathbf{p}_{i/c}^c}{\left(\mathbf{e}_3^\mathsf{T}\hat{\mathbf{p}}_{i/c}^c\right)^2} I_{2\times 3} K \hat{\mathbf{p}}_{i/c}^c. \tag{B.6}$$

To solve for $\frac{\partial}{\partial \tilde{\mathbf{x}}}\mathbf{p}_{i/c}^c$, we must first establish a few approximations related to the landing vehicle attitude state, ψ_I^g. In (A.30) we established that the 2D rotation matrix created from $\tilde{\psi}_I^g$ can be approximated as

$$\left(R_{\text{2D}}\left(\tilde{\psi}_I^g\right)\right)^\mathsf{T} \approx I + \left[\tilde{\psi}_I^g\right]_\times. \tag{B.7}$$

We extend this to the 3D rotation matrix created from $\tilde{\psi}_I^g$ noting that

$$\left(R_{3\mathrm{D}}\left(\tilde{\psi}_I^g\right)\right)^{\mathsf{T}} \approx I + \begin{bmatrix} 0 \\ 0 \\ \tilde{\psi}_I^g \end{bmatrix}_{\times}. \tag{B.8}$$

We now expand (3.115) using the error-state definitions along with the approximations given in (B.8) and (3.33):

$$\mathbf{p}_{i/c}^c = R_b^c \left(R_I^b \left(R_I^g\right)^{\mathsf{T}} \mathbf{r}_{i/g}^g + R_I^b \mathbf{p}_{g/b}^v - \mathbf{p}_{c/b}^b \right) \tag{B.9}$$

$$\approx R_b^c \left(\left(I - \left[\tilde{\theta}_I^b\right]_{\times} \right) \hat{R}_I^b \left(I + \begin{bmatrix} 0 \\ 0 \\ \tilde{\psi}_I^g \end{bmatrix}_{\times} \right) \left(\hat{R}_I^g\right)^{\mathsf{T}} \left(\hat{\mathbf{r}}_{i/g}^g + \tilde{\mathbf{r}}_{i/g}^g \right) \right. \tag{B.10}$$

$$\left. + \left(I - \left[\tilde{\theta}_I^b\right]_{\times} \right) \hat{R}_I^b \left(\hat{\mathbf{p}}_{g/b}^v + \tilde{\mathbf{p}}_{g/b}^v \right) - \mathbf{p}_{c/b}^b \right).$$

(B.11)

By simplifying and ignoring higher-order terms, we get

$$\mathbf{p}_{i/c}^c \approx R_b^c \left(\hat{R}_I^b \left(\hat{R}_I^g\right)^{\mathsf{T}} \hat{\mathbf{r}}_{i/g}^g + \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v - \mathbf{p}_{c/b}^b - \left[\tilde{\theta}_I^b\right]_{\times} \hat{R}_I^b \left(\hat{R}_I^g\right)^{\mathsf{T}} \hat{\mathbf{r}}_{i/g}^g \right. \tag{B.12}$$

$$\left. + \hat{R}_I^b \begin{bmatrix} 0 \\ 0 \\ \tilde{\psi}_I^g \end{bmatrix}_{\times} \left(\hat{R}_I^g\right)^{\mathsf{T}} \hat{\mathbf{r}}_{i/g}^g + \hat{R}_I^b \left(\hat{R}_I^g\right)^{\mathsf{T}} \tilde{\mathbf{r}}_{i/g}^g - \left[\tilde{\theta}_I^b\right]_{\times} \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v + \hat{R}_I^b \tilde{\mathbf{p}}_{g/b}^v \right).$$

$$\approx \hat{\mathbf{p}}_{i/c}^c + R_b^c \left(- \left[\tilde{\theta}_I^b\right]_{\times} \hat{R}_I^b \left(\hat{R}_I^g\right)^{\mathsf{T}} \hat{\mathbf{r}}_{i/g}^g + \hat{R}_I^b \begin{bmatrix} 0 \\ 0 \\ \tilde{\psi}_I^g \end{bmatrix}_{\times} \left(\hat{R}_I^g\right)^{\mathsf{T}} \hat{\mathbf{r}}_{i/g}^g \right. \tag{B.13}$$

$$\left. + \hat{R}_I^b \left(\hat{R}_I^g\right)^{\mathsf{T}} \tilde{\mathbf{r}}_{i/g}^g - \left[\tilde{\theta}_I^b\right]_{\times} \hat{R}_I^b \hat{\mathbf{p}}_{g/b}^v + \hat{R}_I^b \tilde{\mathbf{p}}_{g/b}^v \right).$$

Using (B.13), we define $\frac{\partial}{\partial \tilde{\mathbf{x}}} \mathbf{p}^c_{i/c}$ by its non-zero components,

$$\frac{\partial}{\partial \tilde{\theta}^b_I} \mathbf{p}^c_{i/c} = R^c_b \left[\hat{R}^b_I \left(\left(\hat{R}^g_I \right)^\mathsf{T} \hat{\mathbf{r}}^s_{i/g} + \hat{\mathbf{p}}^v_{g/b} \right) \right]_\times \tag{B.14}$$

$$\frac{\partial}{\partial \tilde{\mathbf{p}}^v_{g/b}} \mathbf{p}^c_{i/c} = R^c_b \hat{R}^b_I \tag{B.15}$$

$$\frac{\partial}{\partial \tilde{\psi}^g_I} \mathbf{p}^c_{i/c} = R^c_b \hat{R}^b_I \left(\hat{R}^g_I \right)^\mathsf{T} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times \hat{\mathbf{r}}^g_{i/g} \tag{B.16}$$

$$\frac{\partial}{\partial \tilde{\mathbf{r}}^g_{i/g}} \mathbf{p}^c_{i/c} = R^c_b \hat{R}^b_I \left(\hat{R}^g_I \right)^\mathsf{T}. \tag{B.17}$$

APPENDIX C. DERIVATION OF ERROR-STATE DYNAMICS FOR LQR CONTROL

To derive the error-state dynamics of the multirotor UAV, we first establish a few identities and approximations. As it is convenient to work with rotation matrices, we first write (4.20) in terms of rotation matrices. Since rotation matrices concatenate in an order opposite to quaterions, we have

$$R_I^b = R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\check{R}_I^b. \tag{C.1}$$

With this, we can express the desired attitude as

$$\check{R}_I^b = R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)^{\mathsf{T}} R_I^b. \tag{C.2}$$

and when $\tilde{\mathbf{r}}_I^b$ is small, we employ (4.11) to get

$$\check{R}_I^b \approx R\left(\begin{bmatrix} 1 \\ \frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right)^{\mathsf{T}} R_I^b \tag{C.3}$$

$$\approx R\left(\begin{bmatrix} 1 \\ -\frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right) R_I^b. \tag{C.4}$$

Now, we can substitute $R\left(\begin{bmatrix} 1 \\ -\frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right)$ into eq. (4.8) to obtain

$$\check{R}_I^b \approx \left(I + \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) R_I^b. \tag{C.5}$$

We also note that the transpose is given by

$$\left(\check{R}_I^b\right)^{\mathsf{T}} = \left(R_I^b\right)^{\mathsf{T}} \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\right) \tag{C.6}$$

and can be similarly approximated as

$$\left(\check{R}_I^b\right)^{\mathsf{T}} \approx \left(R_I^b\right)^{\mathsf{T}} \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right). \tag{C.7}$$

We also employ the skew symmetric identity that

$$[\mathbf{a}]_\times \mathbf{b} = -[\mathbf{b}]_\times \mathbf{a}. \tag{C.8}$$

C.1 Position

Differentiating the error state for the position term given by (4.17) we get

$$\dot{\tilde{\mathbf{p}}}_{b/I}^I = \dot{\mathbf{p}}_{b/I}^I - \dot{\check{\mathbf{p}}}_{b/I}^I. \tag{C.9}$$

Substituting in the multirotor dynamics from (4.14) and simplifying we get

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}}_{b/I}^I &= \left(R_I^b\right)^{\mathsf{T}} \mathbf{v}_{b/I}^b - \left(\check{R}_I^b\right)^{\mathsf{T}} \check{\mathbf{v}}_{b/I}^b && \text{(C.10)}\\
&= \left(R_I^b\right)^{\mathsf{T}} \mathbf{v}_{b/I}^b - \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right) R_I^b\right)^{\mathsf{T}} \check{\mathbf{v}}_{b/I}^b && \text{(C.11)}\\
&= \left(R_I^b\right)^{\mathsf{T}} \mathbf{v}_{b/I}^b - \left(R_I^b\right)^{\mathsf{T}} \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\right)^{\mathsf{T}} \left(\mathbf{v}_{b/I}^b - \tilde{\mathbf{v}}_{b/I}^b\right) && \text{(C.12)}\\
&= \left(R_I^b\right)^{\mathsf{T}} \mathbf{v}_{b/I}^b - \left(R_I^b\right)^{\mathsf{T}} \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) \left(\mathbf{v}_{b/I}^b - \tilde{\mathbf{v}}_{b/I}^b\right). && \text{(C.13)}
\end{aligned}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\dot{\tilde{\mathbf{p}}}_{b/I}^I \approx \left(R_I^b\right)^{\mathsf{T}} \tilde{\mathbf{v}}_{b/I}^b - \left(R_I^b\right)^{\mathsf{T}} \left[\mathbf{v}_{b/I}^b\right]_\times \tilde{\mathbf{r}}_I^b. \tag{C.14}$$

C.2 Velocity

Differentiating the error state for the velocity term given by (4.18) we get

$$\dot{\tilde{\mathbf{v}}}_{b/I}^b = \dot{\mathbf{v}}_{b/I}^b - \dot{\check{\mathbf{v}}}_{b/I}^b. \tag{C.15}$$

Substituting in the multirotor dynamics from (4.14) and simplifying we get

$$\begin{aligned}
\dot{\mathbf{v}}^b_{b/I} = &\left(gR^b_I\mathbf{e}_3 - g\frac{s}{s_e}\mathbf{e}_3 - c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\mathbf{v}^b_{b/I} - \left[\boldsymbol{\omega}^b_{b/I}\right]_\times \mathbf{v}^b_{b/I}\right) \\
&- \left(g\check{R}^b_I\mathbf{e}_3 - g\frac{\check{s}}{s_e}\mathbf{e}_3 - c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\check{\mathbf{v}}^b_{b/I} - \left[\check{\boldsymbol{\omega}}^b_{b/I}\right]_\times \check{\mathbf{v}}^b_{b/I}\right) \qquad \text{(C.16)} \\
= &\left(gR^b_I\mathbf{e}_3 - g\check{R}^b_I\mathbf{e}_3\right) - \left(g\frac{s}{s_e}\mathbf{e}_3 - g\frac{\check{s}}{s_e}\mathbf{e}_3\right) \\
&- \left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\mathbf{v}^b_{b/I} - c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\check{\mathbf{v}}^b_{b/I}\right) \qquad \text{(C.17)} \\
&- \left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times \mathbf{v}^b_{b/I} - \left[\check{\boldsymbol{\omega}}^b_{b/I}\right]_\times \check{\mathbf{v}}^b_{b/I}\right) \\
\approx &\left(gR^b_I\mathbf{e}_3 - g\left(I+\left[\tilde{\mathbf{r}}^b_I\right]_\times\right)R^b_I\mathbf{e}_3\right) \\
&- \left(g\frac{\tilde{s}}{s_e}\mathbf{e}_3\right) - \left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I}\right) \qquad \text{(C.18)} \\
&- \left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times \mathbf{v}^b_{b/I} - \left[\left(\boldsymbol{\omega}^b_{b/I} - \tilde{\boldsymbol{\omega}}^b_{b/I}\right)\right]_\times \left(\mathbf{v}^b_{b/I} - \tilde{\mathbf{v}}^b_{b/I}\right)\right) \\
= &\left(-g\left[\tilde{\mathbf{r}}^b_I\right]_\times R^b_I\mathbf{e}_3\right) - \left(g\frac{\tilde{s}}{s_e}\mathbf{e}_3\right) - \left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I}\right) \\
&- \left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times \mathbf{v}^b_{b/I} - \left[\left(\boldsymbol{\omega}^b_{b/I} - \tilde{\boldsymbol{\omega}}^b_{b/I}\right)\right]_\times \left(\mathbf{v}^b_{b/I} - \tilde{\mathbf{v}}^b_{b/I}\right)\right). \qquad \text{(C.19)}
\end{aligned}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\begin{aligned}
\dot{\tilde{\mathbf{v}}}^b_{b/I} \approx &\, g\left[R^b_I\mathbf{e}_3\right]_\times \tilde{\mathbf{r}}^b_I - g\frac{\tilde{s}}{s_e}\mathbf{e}_3 - c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I} \\
&- \left[\boldsymbol{\omega}^b_{b/I}\right]_\times \tilde{\mathbf{v}}^b_{b/I} + \left[\mathbf{v}^b_{b/I}\right]_\times \tilde{\boldsymbol{\omega}}^b_{b/I}.
\end{aligned} \qquad \text{(C.20)}$$

C.3 Attitude

To derive the error-state dynamics corresponding to attitude, we start with (4.22). From (4.19) we see that $\dot{\mathbf{r}}^b_I = \boldsymbol{\omega}^b_{b/I}$. Substituting this defintion into (4.22) we get

$$\dot{\tilde{\mathbf{r}}}^b_I = \boldsymbol{\omega}^b_{b/I} - R^b_I\left(\check{R}^b_I\right)^\mathsf{T}\check{\boldsymbol{\omega}}^b_{b/I}. \qquad \text{(C.21)}$$

We can simplify this expression and show that

$$\dot{\tilde{\mathbf{r}}}_I^b = \omega_{b/I}^b - R_I^b \left(R_I^b\right)^{\mathsf{T}} \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\right)^{\mathsf{T}} \breve{\omega}_{b/I}^b \tag{C.22}$$

$$\approx \omega_{b/I}^b - R_I^b \left(R_I^b\right)^{\mathsf{T}} \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) \breve{\omega}_{b/I}^b \tag{C.23}$$

$$= \omega_{b/I}^b - \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) \left(\omega_{b/I}^b - \tilde{\omega}_{b/I}^b\right). \tag{C.24}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\dot{\tilde{\mathbf{r}}}_I^b \approx \tilde{\omega}_{b/I}^b - \left[\omega_{b/I}^b\right]_\times \tilde{\mathbf{r}}_I^b. \tag{C.25}$$