Computer Vision-Based Motion Control and State Estimation for Unmanned Aerial Vehicles (UAVs)

by

Geoffrey Fink

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Control Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

To achieve a fully autonomous unmanned aerial vehicle (UAV) the vehicle needs a high level of self awareness. At a minimum it needs to know where it is and where it wants to go. Computer vision (CV) is a logical solution to this problem. However, using CV to solve motion control problems for UAVs is a challenging problem as both the quadrotor dynamics and camera kinematics are nonlinear. Although both CV and UAVs are not new topics there are relatively few fully autonomous experimental results due to the difficulty in building and maintaining an experimental platform and the high computation power required by many CV algorithms. Recent advances in embedded computers have notably changed the field. This thesis focuses on three important areas: developing a reliable and powerful UAV platform for testing bleeding edge CV algorithms, dynamic image-based visual servoing (DIBVS), and monocular visual state estimation (VSE).

In order to efficiently perform experimental research on nonlinear control and CV, we developed the indoor Applied Nonlinear Control Lab (ANCL) quadrotor platform. The design for our experimental platform was inspired from other indoor flight test stands and we have prioritized open-source hardware and open-source software. In addition to building the platform we derived a kinematic and dynamic model of a quadrotor and then experimentally identified the system parameters. Also, using a series of experiments we characterized the performance of the wireless communication network between the quadrotor and the motion capture system (MCS) to improve the onboard position control.

This thesis proposes several DIBVS control laws for a quadrotor equipped with a single fixed onboard camera. The motion control problem is to regulate the relative

position and yaw of the vehicle to a target located on the ground. The control law is termed *dynamic* as it based on the dynamics and kinematics of the vehicle. The proposed designs use a nonlinear change of state coordinates, the virtual camera method, adaptive control techniques, and image features. The control laws developed have proven convergence rates. Simulation and experimental results demonstrate the methods' ease of onboard implementation, performance, and robustness.

Next, in this thesis we develop a visual odometry (VO) system which is an onboard CV-based navigation system. An important feature of VO systems is that they are independent of a global navigation satellite system (GNSS). Our approach uses inertial sensor measurements along with scaled position measurements from an onboard CV system which implements a visual simultaneous localization and mapping (VSLAM) system. We study the observability of the visual inertial simultaneous localization and mapping (VISLAM) problem using a state transformation that puts the system into linear time-varying (LTV) form and simplifies the observability analysis. This leads to an observer design with sufficient conditions for convergence. The observer fuses an accelerometer measurement from an inertial measurement unit (IMU) with scaled position measurements from a VSLAM system to estimate vehicle position and velocity. Our approach does not require an approximate linearization of the model equations whereas typical solutions in the literature use an extended Kalman filter (EKF) that linearizes the model equations. Simulations and experimental results onboard a quadrotor UAV validate the proposed designs.

Preface

- The research conducted for this thesis in Chapter 5 forms part of an international research collaboration led by Professor Alan F. Lynch at the University of Alberta, with Professor Klaus Röbenack and Mirko Franke at Technische Universität Dresden.

- Section 4.1 has been published as [G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Nonlinear dynamic visual servoing of a quadrotor," *Journal of Unmanned Vehicle Systems*, vol. 3, no. 1, pp. 1–21, 2015]. I was responsible for the data collection and analysis as well as the manuscript composition. H. Xie assisted with the analysis and contributed to manuscript edits. A. F. Lynch and M. Jagersand were the supervisory authors and were involved with the concept formation and manuscript composition.

- Section 4.2 with the exception of Section 4.2.1 has been published as [G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Experimental validation of dynamic visual servoing for a quadrotor using a virtual camera," in Proceedings of the *International Conference on Unmanned Aircraft Systems*, Denver, CO, Jun. 2015, pp. 1231–1240]. I was responsible for the data collection and analysis as well as the manuscript composition. H. Xie assisted with the analysis and contributed to manuscript edits. A. F. Lynch and M. Jagersand were the supervisory authors and were involved with the concept formation and manuscript composition.

- Section 4.3 has been published as [H. Xie, G. Fink, A. F. Lynch, and M. Jagersand, "Adaptive dynamic visual servoing of a UAV," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2529–2538, 2016]. H. Xie was responsible for the data collection and analysis as well as the manuscript composition. I assisted with the data collection and analysis, and contributed to manuscript edits. A. F. Lynch and M. Jagersand were the supervisory authors and were involved with the concept formation and manuscript composition.

- Section 4.4 has been published as [G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Dynamic visual servoing for a quadrotor using a virtual camera," *Unmanned Systems*, vol. 5, no. 1, pp. 1–17, 2017]. I was responsible for the data collection and analysis as well as the manuscript composition. H. Xie assisted with the analysis and contributed to manuscript edits. A. F. Lynch and M. Jagersand were the supervisory authors and were involved with the concept formation and manuscript composition.

- Parts of Chapter 5 have been published as [G. Fink, M. Franke, A. F. Lynch, K. Röbenack, and B. Godbolt, "Visual inertial SLAM: Application to unmanned aerial vehicles," in Proceedings of the *IFAC World Congress* Toulouse, France, Jul. 2017, pp. 2001–2006]. I was responsible for the data collection and analysis as well as the manuscript composition. M. Franke was responsible for the analysis and contributed to manuscript edits. A. F. Lynch and K. Röbenack were the supervisory authors and were involved with the concept formation and manuscript composition. B. Godbolt was involved with the concept formation.

- Parts of Chapter 5 have been published as [G. Fink, M. Franke, A. F. Lynch, K. Röbenack, and B. Godbolt, "Observer design for visual inertial SLAM scale on a quadrotor UAV," in Proceedings of the *International Conference on Unmanned Aircraft Systems*, Miami, FL, Jun. 2017, pp. 830–839]. I was responsible for the data collection and analysis as well as the manuscript composition. M. Franke was responsible for the analysis and contributed to manuscript edits. A. F. Lynch and K. Röbenack were the supervisory authors and were involved with the concept formation and manuscript composition. B. Godbolt was involved with the concept formation.

- Parts of Chapter 5 have been published as [G. Fink, M. Franke, A. F. Lynch, and K. Röbenack, "Observer design for monocular visual inertial slam," *Automatisierungstechnik*, vol. 66, no. 3, pp. 246–257, 2018]. I was responsible for analysis as well as the manuscript composition. M. Franke was responsible for the analysis and contributed to manuscript edits. A. F. Lynch and K. Röbenack were the supervisory authors and were involved with the concept formation and manuscript composition.

- The experimental platform as described in Chapter 3 was designed by myself.

- All the additional analysis and conclusions included in this thesis are my original work.

To Abril

Table of Contents

List of Tables

List of Figures

List of Symbols

Ω_i	Angular speed of propeller i
Θ_i	The angle from the basis vector b_1 to the arm i
ℓ_i	Distance from the motor i to the CoM
η	ZYX Euler angles $\eta = [\phi, \theta, \psi]^T \in \mathbb{R}^3$
κ_i	Radial Distortion model coefficients
λ_i	Focal length $\lambda_i > 0$, $i \in [1, 2]$ (Chapter 2 & 4)
λ	VSLAM map scale $\lambda > 0$ (Chapter 5)
ν	Velocity screw $\nu = [v^T, \omega^T]^T \in \mathbb{R}^6$
ω^x	Angular velocity of the vehicle in $\mathcal{X}$ frame
ϕ	Roll angle
π	Image Plane
π_d	Radial distortion Function
ψ	Yaw angle
ρ_i	Size of pixel $\rho_i > 0$
τ^x	External torques acting on the vehicle in $\mathcal{X}$ frame
θ	Pitch angle
$\sim$	Equality up to non-zero scaling
A	Camera calibration matrix
a^x	Acceleration in $\mathcal{X}$ frame
$\mathcal{B}$	Body frame with basis $\{b_1, b_2, b_3\}$
b^x	Accelerometer bias in $\mathcal{X}$ frame
$\mathcal{C}$	Camera frame with basis $\{c_1, c_2, c_3\}$
c_θ	$\mathrm{c}_\theta = \cos\theta$
F^x	External forces acting on the vehicle in $\mathcal{X}$ frame
f_ℓ	Focal length
g^x	Acceleration due to gravity in $\mathcal{X}$ frame
H	Homography matrix
J^x	Inertia Matrix in $\mathcal{X}$ frame
k_τ	Aerodynamic coefficient for torque

k_f	Aerodynamic coefficient for force
L	The interaction matrix or image Jacobian
l^x	Line in $\mathcal{X}$ frame
m	Mass of the vehicle
$\mathcal{N}$	Navigation frame with basis $\{n_1, n_2, n_3\}$
$\mathcal{O}$	A dense object with a continuous surface.
p^x	Position (body or point) in $\mathcal{X}$ frame
p_c^b	Position of origin of frame $\mathcal{C}$ relative to origin of frame $\mathcal{B}$
q	Unit quaternion
R_y^x	Rotation from frame $\mathcal{Y}$ to frame $\mathcal{X}$, $R_y^x \in \mathrm{SO}(3)$
s	Skew parameter of camera
s_θ	$\mathrm{s}_\theta = \sin\theta$
$\mathrm{sk}(x)$	Skew symmetric operator
SO	Special Orthogonal Group
$\mathcal{T}$	Target frame with basis $\{t_1, t_2, t_3\}$
t_θ	$\mathrm{t}_\theta = \tan\theta$
$\mathcal{V}$	Vision frame with basis $\{v_1, v_2, v_3\}$
v^x	Linear velocity of the vehicle in $\mathcal{X}$ frame
Y^x	Image coordinates (Homogeneous) from the $\mathcal{X}$ camera
y^x	Image coordinates (Cartesian) from the $\mathcal{X}$ camera
y_0	Principal point
y_d	Distorted image coordinates
y_u	Undistorted image coordinates

List of Acronyms

AGAST	Adaptive and Generic Accelerated Segment Test
AHRS	Attitude and Heading Reference System
ANCL	Applied Nonlinear Control Lab
AO	Adaptive Observer
BA	Bundle Adjustment
BIBO	Bounded-Input, Bounded-Output
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
CenSurE	Center Surround Extremas
CoM	Centre of Mass
CV	Computer Vision
DIBVS	Dynamic Image-Based Visual Servoing
DIY	Do-It-Yourself
DoF	Degree of Freedom
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FAST	Features from Accelerated Segment Test
FoV	Field of View
fps	Frames per Second
GAS	Globally Asymptotically Stable
GES	Globally Exponentially Stable
GFTT	Good Features to Track
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
IBVS	Image-Based Visual Servoing
IMU	Inertial Measurement Unit
KF	Kalman filter

LATCH	Learned Attachment of Three Patch Codes
LES	Locally Exponentially Stable
LTV	Linear Time-Varying
MCS	Motion Capture System
n D	n-Dimensional
ORB	Orientated FAST and rotated BRIEF
P	Proportional
PBVS	Position-Based Visual Servoing
PD	Proportional-Derivative
PE	Persistently Exciting
PID	Proportional-Inegral-Derivative
PTAM	Parallel Tracking and Mapping
PWM	Pulse Width Modulation
RMSE	Root Mean Square Error
ROS	Robot Operating System
SFM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Feature
UAV	Unmanned Aerial Vehicle
UCO	Uniform Complete Observability
UD	Uniformly Detectable
uORB	Micro-Object Request Broker
USB	Universal Synchronous Bus
VISLAM	Visual Inertial Simultaneous Localization and Mapping
VO	Visual Odometry
VS	Visual Servoing
VSE	Visual State Estimation
VSLAM	Visual Simultaneous Localization and Mapping

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) are used for a number of indoor and outdoor applications such as search and rescue, surveillance, and infrastructure inspection. To achieve a fully autonomous UAV the vehicle needs a high level of self awareness. At a minimum it needs to know where it is and where it wants to go. A range of sensors exist to try to fulfill this need, e.g., light detection and ranging (lidar) systems, global navigation satellite systems (GNSSs), and cameras. For UAVs the most commonly used positioning systems are GNSSs and motion capture systems (MCSs) for outdoors and indoors, respectively. While there has been a lot of success with these sensors they have some important limitations. Both GNSSs and MCSs are external sensors and hence rely on a solid communication link between the vehicle and the external sensor. This link can easily be broken due to occlusion or intentional signal jamming. Furthermore, they can be inaccurate and do not provide a position estimate relative to a target of interest. This has led to research on computer vision (CV) to provide an alternate or augmentation source for performing state estimation. CV provides a number of benefits over conventionally used sensors: it can be placed onboard, it can run at a high frequency, and can provide rich information about a scene such as relative position to a visual target. Additionally, cameras have the advantage of being lightweight, small, low cost, low power, and passive.

1.1 Literature Review

Control of UAVs is an active and broad area of research (e.g., Kendoul [1], Kanellakis and Nikolakopoulos [2]). Hence, this literature review is not meant to be exhaustive, but instead we highlight some of the key works related to how CV can be used to solve motion control problems for UAVs. The advancement of hardware has enabled powerful CV algorithms to be run in real-time and low size, weight, power and cost (SWaP-C) make them ideal for onboard use. However, there are still few published

experimental results with CV running in real-time autonomously onboard UAVs due to the many difficulties of working with an inherently unstable system like a quadrotor. Furthermore, key aspects of CV such as tracking are still not reliable enough for industrial use in unstructured environments. Despite all of the difficulties there have been some strong developments published over the last five to ten years.

Visual servoing (VS) is a control loop with visual feedback. It is divided into two main approaches: position-based visual servoing (PBVS) and image-based visual servoing (IBVS) (Hutchinson et al. [3]). In PBVS the pose of the vehicle is extracted from the image and the motion control of the vehicle can be done using conventional motion control techniques where the vehicle's state is assumed measured. Examples of PBVS methods applied to UAVs include Shakernia et al. [4], Altug et al. [5], Wu et al. [6], Mejias et al. [7], Azrad et al. [8], Garcia Carrillo et al. [9], Fraundorfer et al. [10], Sa and Corke [11]. In general, PBVS requires more a priori knowledge of the target scene or object. For example, to track the relative pose between a UAV and a target, some researchers assume that a computer aided design (CAD) model of the target is known in advance, e.g., Tamadazte et al. [12]. The object corresponding to this CAD model is identified in the environment and tracked. Because the CAD model Euclidean dimensions are known, the relative Euclidean pose between the UAV and the target can be calculated. Often a specially designed target marker (e.g., landing pad pattern or AprilTags) is placed in the environment to facilitate tracking, e.g., Lee et al. [13] or Lin et al. [14]. However, this limits usage in natural unstructured environments. Target based tracking can be done with either static targets placed on the ground or moving targets such as other vehicles. Motion models of the targets may or may not be known. Tracking targets with known motion include friendly vehicles that can communicate with the UAV e.g., a train going at constant velocity on a known track.

Visual state estimation (VSE) can also be used to provide the pose of the vehicle for PBVS (Bonin-Font et al. [15]). VSE methods rely on visual odometry (VO) (Scaramuzza and Fraundorfer [16], Fraundorfer and Scaramuzza [17], Yousif et al. [18]) or visual simultaneous localization and mapping (VSLAM) (Durrant-Whyte and Bailey [19], Bailey and Durrant-Whyte [20], Neira et al. [21], Cadena et al. [22]). The main difference between VO and VSLAM is that in VO just the pose of the vehicle is being calculated whereas in VSLAM a map is being created at the same time. One of the early VSLAM methods that arose from structure from motion (SFM) (Civera et al. [23]) is parallel tracking and mapping (PTAM) (Klein and Murray [24]). In it the authors proposed an algorithm that separates the tracking and mapping into two different threads and the mapping is based on keyframes which are processed using bundle adjustment (BA) or optimization. Examples of VSLAM include: Civera et al. [25] which proposes an inverse depth method for a

single camera; Fraundorfer et al. [10] which uses simultaneous localization and mapping (SLAM) with a front-facing stereo camera to perform mapping and exploration for a quadrotor; S-PTAM (Pire et al. [26]) is modernized version of PTAM that uses stereo vision; monoSLAM (Davison et al. [27]) which uses probabilistic mapping of a sparse map; FastSLAM 2 (Montemerlo et al. [28]) which is based on a particle filter; and ORB-SLAM2 (Mur-Artal and Tards [29]) which uses the oriented FAST and rotated BRIEF (ORB) feature detector. In Geiger et al. [30] and Kümmerle et al. [31] the authors propose standards to be able to compare VSLAM methods and provide standard data sets. As with all real systems the output of VSLAM systems is not perfect: the pose will drift over time and monocular systems will have an unknown scale factor. In a recent comparison of VSLAM algorithms (Quattrini Li et al. [32]) the authors demonstrate the difficulty of applying the algorithms to new datasets. They show sub-optimal performance, loss of localization and other challenges. Out of all of the methods compared none of them could work on more then half of the real world video sequences. It is also interesting to note that PTAM (the VSLAM system used in this thesis) performed just as good as the new methods despite being ten years old.

To overcome the unknown scale, performance, and drift issues data from other sensors can be fused with the VSLAM pose. The most common sensor combination is to combine VSLAM output with inertial measurement unit (IMU) measurements due to their complimentary nature. These sensor fusion algorithms are called visual inertial simultaneous localization and mapping (VISLAM). Loosely coupled VISLAM fuses VSLAM and IMU with a filter (Bryson and Sukkarieh [33], Weiss [34]). If, on the other hand, the algorithm jointly estimates all sensor states then they are called tightly coupled, e.g., Leutenegger [35]. The advantage of loosely coupled methods is that they limit the complexity, however, they disregard the correlations between the sensors. In Concha et al. [36] the authors propose a tightly coupled VISLAM that creates a dense map. Normally the literature focuses on sparse maps such as point clouds.

Another challenge of SLAM-based motion control for flying UAVs in unknown environments is that the coordinate frame recovered by SLAM is arbitrary and needs to be aligned with the world frame. The precision of the executed trajectory will depend both on this external alignment and the internal accuracy of the SLAM model. IBVS on the other hand defines the error function in image coordinates and does not depend on the quality or alignment of a global model. In our anecdotal testing of the two systems we found reprojection errors on the order of ten pixels for 3 D features in the SLAM map. By contrast single feature or patch registration tracking can track to camera pixel precision (Lieberknecht et al. [37], Dick et al. [38]). This means that for a given camera sensor resolution IBVS has the potential to

provide more accurate positioning. An additional challenge for monocular VSLAM methods is that they need a large visual area to extract and compute coordinates for 3 D points. Normally the map is made of a static scene terrain, although a large moving object such as a ship would be possible.

We identify a number of dynamic IBVS approaches for UAVs that have appeared in the literature: the spherical image moment based design, homography based methods, the virtual spring approach, the virtual camera approach and the state transformation approach. It is interesting to note that all the abovementioned methods involve trying to recover a passivity property of the image feature kinematics to prove stability of the control.

One of the earliest approaches is the spherical image moment method (Hamel and Mahony [39]). Here a dynamic IBVS controller based on a backstepping design is proposed. First order spherical image moments are used as a visual feature and the resulting interaction matrix gives the system dynamics important passivity properties. Further work in Hamel and Mahony [40] removes the requirement of an attitude sensor which makes the stabilization problem more challenging. An important extension to the work uses optical flow to estimate scaled linear velocity (Bras et al. [41]).

Another popular tool in CV is the homography matrix. A homography matrix relates the pose of two camera views of the same planar target (Ma et al. [42]). In Metni and Hamel [43] the authors use a homography decomposition to obtain a new image feature whose kinematics have a passivity property. In this approach only the position of the UAV is controlled and control of the yaw orientation is not addressed. Work in de Plinval et al. [44] proposes a homography-based feature error vector that is diffeomorphically related to camera pose. A local stabilizing control law is developed that only needs vehicle angular velocity and visual measurement of a planar target.

Work in Ozawa and Chaumette [45] introduced a virtual spring approach where image feature moments lead to an interaction matrix with an identity matrix in its translational component. The work assumes knowledge of the desired height and that feature points are coplanar and parallel to the image plane. This approach is motivated by the UAV's underactuation where increasing image feature error reduces lateral position error. The control law is independent of linear velocity and its asymptotic stability is proven. The virtual spring method is limited by the assumption that the image plane is parallel to the target plane. Lateral motion of a traditional underactuated quadrotor UAV requires roll and pitch motion which means the image and target planes are no longer parallel. Even a small change in roll or pitch may cause a large change in the interaction matrix.

Work in Lee et al. [13, 46] introduced an IBVS design based on a virtual image

plane. This plane has zero-roll and pitch and has the same position and yaw of the real camera's image plane. This virtual plane facilitates the estimation of depth of image points. A classical IBVS method is used to generate a reference velocity screw, and an adaptive sliding mode control is used in the inner loop. Work in Jabbari Asl et al. [47] uses image moments in the same virtual plane and the feature kinematics has a passivity like property. An adaptive backstepping controller is developed to stabilize the feature error. Virtual camera methods require UAV attitude and linear velocity measurements. In Jabbari Asl and Yoon [48] the authors combine optical flow with the virtual camera approach to eliminate the need for an velocity estimate.

Work in Xie [49] proposed a state transformation to eliminate the angular velocity dependence in the image kinematics. The state transformation was obtained by solving a system of first-order homogeneous partial differential equations. It can be shown that the virtual camera approach is a particular solution from the general solution. Hence, this approach provides alternatives to the virtual camera transformation for eliminating angular velocity from the image kinematics. In Xie et al. [50] the authors propose an adaptive output feedback-based visual servoing law that removes the requirement of translational velocity measurements.

Both PBVS and IBVS require the online tracking of real targets in real environments. Tracking is a difficult problem and the tracking of multiple points might restrict applicability to an artificial environment given current tracker performance. Recent surveys (Galoogahi et al. [51], Liang et al. [52]) benchmarked state-of-the-art object tracking algorithms. This work demonstrates tracking is a computationally expensive process and lacks robust performance. The outputs of most modern trackers are a single point and a bounding box, with the point feature typically being more robust than the bounding box.

CV algorithms running onboard unmanned vehicles is an active area of research. The combined use of these technologies has led to some impressive accomplishments such as the Mars rover project (Maimone et al. [53]). As new algorithms are being developed UAVs are becoming more and more autonomous. This review has highlighted some of the important literature that has been published to date.

1.2 Overview of Thesis

This thesis is divided into three main parts: experimental platform design, VS, and VSE. Before delving into the main content, we present an established nonlinear quadrotor model in Chapter 2. We derive the kinematics, dynamics, force and torque models of a quadrotor. Using this model, we present our inner-outer loop control strategy for the vehicle. Next we examine the pinhole camera model. From this model we define a point feature and derive its image kinematics. Lastly, we

present a brief introduction on VS. This chapter also helps to define the notation that will be used throughout this thesis.

In Chapter 3 we present our experimental platform. Our indoor quadrotor platform has gone through many hardware and software iterations over the course of this thesis. Everything has been improved upon including the frame, autopilot, electronic speed controllers (ESCs), motors, propellers, and companion computers. As part of the our model-based design we have also experimentally identified many of the quadrotor model parameters. As our test stand is indoors, our Vicon MCS plays an important role in both the control of the quadrotor and for verification of the CV algorithms. In an effort optimize the performance of the network we propose experiments to fully characterize the performance of the network used to transmit MCS data to the vehicle.

In Chapter 4 we present state transformation and virtual camera dynamic image-based visual servoing (DIBVS) methods. Additionally, we derive the image moments features and their kinematics and we present the homography matrix and its decomposition. We propose five different DIBVS control laws. The first virtual camera-based method is a simple control law using a state transformation and just one image feature point to control the lateral motion of the quadrotor. Then we propose four control laws using the virtual camera method with image moment features. The first is a straight forward application of the theory to validate that the method is sufficiently robust to be used in experiment. Up until Fink et al. [54], only simulated results of the approach were available. Next we extend the control law using an adaptive scheme. Lastly, using the homography matrix we extend the method for moving and non-horizontal targets. All of the control laws with the exception of the homography matrix approach are experimentally implemented and validated. Thorough experimental validation is a main theme of this thesis as it ensures the theory can be used in practice.

In Chapter 5 we propose an observer design to estimate vehicle position, linear velocity, and accelerometer bias. The observer fuses an accelerometer measurement from an IMU and a scaled position estimate from a VSLAM system. The observer depends on an attitude estimate from an attitude and heading reference system (AHRS). A change of coordinates is used to transform the system into linear time-varying (LTV) form. Using these coordinates we consider the observability of the VISLAM problem. We prove that the observer is globally exponentially stable (GES). Lastly, we simplify the assumptions on the system and provide an improved stability proof. Finally, in Chapter 6 we present our conclusions and future work.

1.3 Contributions

The contributions of this thesis are summarized as follows:

Dynamic Image-Based Visual Servoing

- In Fink et al. [54] we proposed a DIBVS control for a quadrotor UAV equipped with a single fixed onboard camera facing downward. The motion control problem was to regulate the relative lateral position of the vehicle to a stationary target located on the ground. The proposed design uses nonlinear input-dependent change of state coordinates and its error dynamics were proven to be locally exponentially stable (LES).

- Next, in Fink et al. [55] we expanded the motion control problem to regulate the full 3 D relative position and yaw of the vehicle to a stationary target located on the ground. The proposed design relies on the notion of a virtual camera and image moment visual features to simplify the kinematics and dynamics. The convergence of the closed-loop was proven to be globally asymptotically stable (GAS).

- Later, in Xie et al. [56] we proposed an adaptive DIBVS control. The motion control problem was as in [54], however, the adaptive design accounted for unknown thrust constant, mass, gravity constant, attitude estimation errors, and desired depth. The proposed design relies on the notion of a virtual camera and image moment visual features to simplify the kinematics and dynamics. The convergence of the closed-loop was proven to be GAS.

- Lastly, in Fink et al. [57] we further expanded the motion control problem for the cases where the target is moving and for when the target is non-horizontal. Similarly, the proposed design relies on the notion of a virtual camera and image moment visual features. Furthermore, it introduced a homography matrix decomposition to compensate for non-horizontal targets.

Monocular Visual Inertial Simultaneous Localization and Mapping

- In Fink et al. [58, 59, 60] we proposed an observer design to estimate vehicle position, linear velocity, and accelerometer bias. The observer fuses an accelerometer measurement from an IMU and scaled position estimates from a VSLAM system. The observer depends on an attitude estimate from an AHRS. A change of coordinates is used to transform the system into LTV form. Using these coordinates we consider the observability of the VISLAM problem. We prove that the observer is GES.

- In Fink et al. [61] we provided sufficient conditions for observer estimate error convergence.

Experimental Platform

- The Applied Nonlinear Control Lab (ANCL) quadrotor platform was designed and implemented in Fink et al. [54]. We remark that implementation could be considered too mundane to be classified as a scientific contribution. However, we argue that research on platform development is novel work which is required to develop relevant control theory. Thanks in part to open-source projects, the platform was developed with full configurability and this was essential to develop a wide range of theory.

- The DIBVS controllers (Fink et al. [54, 55], Xie et al. [56], Fink et al. [57]) were experimentally validated on the ANCL platform. This is particularly noteworthy due to the lack of experimental DIBVS results in the literature.

- The observers in Fink et al. [59, 60] were experimentally validated on the ANCL platform. This is also of interest due to the CV algorithms being implemented on embedded hardware.

- In Fink et al. [62] we characterized and then optimized the communication between the quadrotor and the MCS.

Chapter 2

Quadrotor and Camera Modelling

This chapter presents the mathematical models and defines the notation that will be used throughout this thesis. In Section 2.1 we derive the quadrotor kinematics and dynamics and then present a model-based proportional-integral-derivative (PID) control law. In Section 2.2 we present a commonly used camera model and image features, detectors, and descriptors. Lastly, we introduce the concept of visual servoing (VS).

2.1 Quadrotor Kinematics, Dynamics and Control

This section presents an established nonlinear quadrotor model. The first two reference frames of interest are the navigation frame $\mathcal{N}$ and the body frame $\mathcal{B}$. The navigation frame is assumed inertial and has its origin fixed to the surface of the earth. Its basis is the orthonormal set of vectors $\{n_1, n_2, n_3\}$ that are oriented north, east, and down, respectively. The body frame has its origin fixed to the quadrotor's centre of mass (CoM) and its basis $\{b_1, b_2, b_3\}$ is oriented forward, right, and down, respectively. Figure 2.1 shows frames $\mathcal{N}$ and $\mathcal{B}$.

2.1.1 Representation of the Orientation

The rotation matrix $R_b^n \in \mathrm{SO}(3)$ describes the relative orientation of $\mathcal{N}$ and $\mathcal{B}$ [63]. The columns of R_b^n are the direction cosines of the basis vectors of $\mathcal{B}$ and $\mathcal{N}$. We have

$$R_b^n = \begin{bmatrix} n_1 \cdot b_1 & n_2 \cdot b_1 & n_3 \cdot b_1 \\ n_1 \cdot b_2 & n_2 \cdot b_2 & n_3 \cdot b_2 \\ n_1 \cdot b_3 & n_2 \cdot b_3 & n_3 \cdot b_3 \end{bmatrix}$$

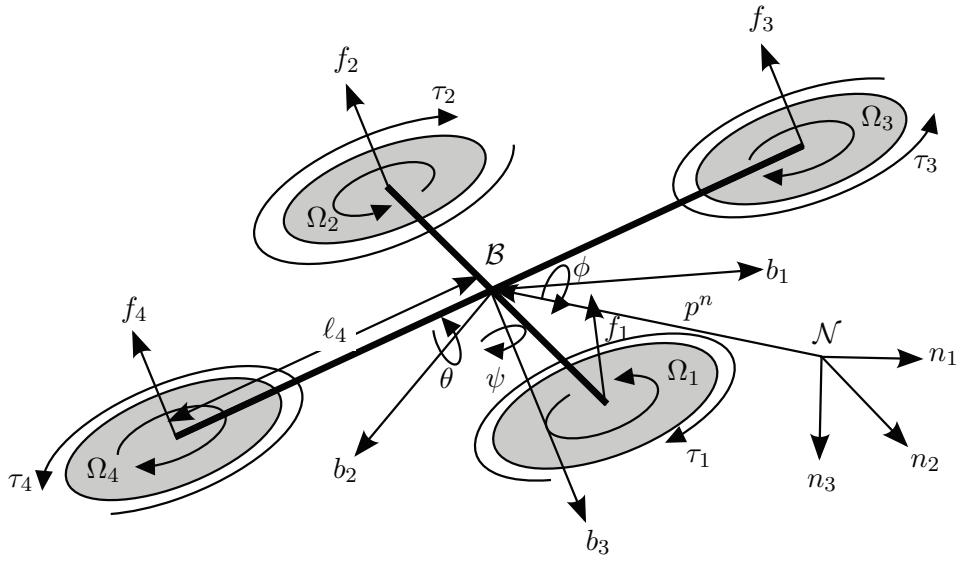


Figure 2.1: .
Diagram of a quadrotor showing frames $\mathcal{N}$ and $\mathcal{B}$, Euler angles, and the forces and torques generated by the propellers.

The elementary rotation matrices are rotation matrices that rotate about just a single axes. They are

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\theta & -\mathrm{s}_\theta \\ 0 & \mathrm{s}_\theta & \mathrm{c}_\theta \end{bmatrix}, \quad R_2(\theta) = \begin{bmatrix} \mathrm{c}_\theta & 0 & \mathrm{s}_\theta \\ 0 & 1 & 0 \\ -\mathrm{s}_\theta & 0 & \mathrm{c}_\theta \end{bmatrix}, \quad R_3(\theta) = \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & 0 \\ \mathrm{s}_\theta & \mathrm{c}_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

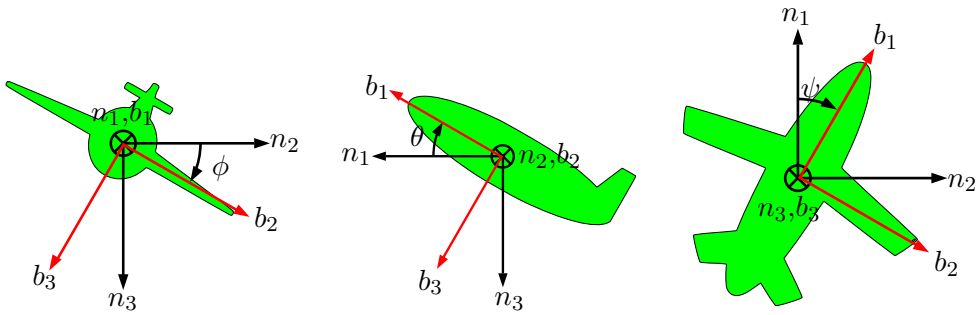
where $\mathrm{c}_\theta = \cos\theta$ and $\mathrm{s}_\theta = \sin\theta$. This thesis typically parametrizes the rotation between the navigation frame $\mathcal{N}$ and the body frame $\mathcal{B}$ with the so-called roll-pitch-yaw (ϕ-θ-ψ) or ZYX Euler angles. The rotation matrix when parametrized by the Euler angles is

$$R_n^b(\eta) = R_3(\psi)R_2(\theta)R_1(\phi) \tag{2.1}$$

$$= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \tag{2.2}$$

$$= \begin{bmatrix} \mathrm{c}_\theta\mathrm{c}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi & \mathrm{c}_\psi\mathrm{s}_\theta\mathrm{c}_\phi + \mathrm{s}_\psi\mathrm{s}_\phi \\ \mathrm{c}_\theta\mathrm{s}_\psi & \mathrm{s}_\psi\mathrm{s}_\theta\mathrm{s}_\phi + \mathrm{c}_\psi\mathrm{c}_\phi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{c}_\theta\mathrm{s}_\phi & \mathrm{c}_\theta\mathrm{c}_\phi \end{bmatrix} \tag{2.3}$$

where $\eta = [\phi, \theta, \psi]^T$. The downside of using the Euler angle approximation is the gimbal lock problem. There is a singularity when $\theta = \pi/2 + k\pi$, $k \in \mathbb{Z}$. Normally this



(a) Roll (ϕ) (b) Pitch (θ) (c) Yaw (ψ)

Figure 2.2: The roll-pitch-yaw (ϕ-θ-ψ) or ZYX Euler angles.

singularity is only encountered in acrobatic flight. However, typical visual servoing (VS) applications do not approach the singularity. The advantage of the Euler angles is that they can be easily visualized as seen in Figure 2.2.

The unit quaternion q is used to represent rotations. Quaternions are a minimal representation of a rotation without singularities, however, each rotation does not have a unique quaternion, as there is double coverage. In the quaternion $q = [q_0, q_1, q_2, q_3]^T$, q_0 is the scalar part of the quaternion and $[q_1, q_2, q_3]$ is the vector part. The rotation matrix parametrized by a unit quaternion is

$$R_n^b(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.4}$$

2.1.2 Quadrotor Rigid Body Dynamics

The rigid body kinematics and dynamics of the vehicle expressed in $\mathcal{N}$ are

$$\dot{p}^n = v^n \tag{2.5a}$$
$$m\dot{v}^n = R_b^n F^b + mg^n \tag{2.5b}$$
$$\dot{R}_b^n = \text{sk}(\omega^n) R_b^n \tag{2.5c}$$
$$J^n \dot{w}^n = -\omega^n \times J^n \omega^n + R_b^n \tau^b \tag{2.5d}$$

where $p^n \in \mathbb{R}^3$ is the position of the vehicle in $\mathcal{N}$, $v^n \in \mathbb{R}^3$ is the linear velocity of the vehicle in $\mathcal{N}$, $\omega^n \in \mathbb{R}^3$ is the angular velocity of the vehicle in $\mathcal{N}$, $m \in \mathbb{R}$ is the mass of the vehicle, $g^n \in \mathbb{R}^3$ is the acceleration due to gravity, $J^n \in \mathbb{R}^{3\times3}$ is the inertia matrix of the vehicle in $\mathcal{N}$. The external force and torque acting on the vehicle are denoted $F^b \in \mathbb{R}^3$, $\tau^b \in \mathbb{R}^3$, respectively, and the skew operator $\text{sk}(x)$

maps a vector to a skew symmetric matrix

$$\mathrm{sk}(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

where $x = [x_1, x_2, x_3]^T$. We use the notation x_i^y to denote the ith component of the vector x in the $\mathcal{Y}$ frame. Depending on the control law it is sometimes convenient to examine the dynamics in $\mathcal{B}$.

$$\dot{p}^b = -\omega^b \times p^b + v^b \tag{2.6a}$$
$$m\dot{v}^b = -m\omega^b \times v^b + F^b + mR_n^b g^n \tag{2.6b}$$
$$\dot{R}_b^n = R_b^n \,\mathrm{sk}(\omega^b) \tag{2.6c}$$
$$J^b\dot{\omega}^b = -\omega^b \times J^b\omega^b + \tau^b \tag{2.6d}$$

where $p^b \in \mathbb{R}^3$ is the position of the vehicle in $\mathcal{B}$, $v^b \in \mathbb{R}^3$ is the linear velocity of the vehicle in $\mathcal{B}$, $\omega^b \in \mathbb{R}^3$ is the angular velocity of the vehicle in $\mathcal{B}$, and $J^b \in \mathbb{R}^{3\times 3}$ is the inertia matrix of the vehicle in $\mathcal{B}$.

The moment of inertia in $\mathcal{N}$ is time-varying, however, it can be calculated from the constant moment of inertia in $\mathcal{B}$. The relationship between the inertia in $\mathcal{N}$ and $\mathcal{B}$ is

$$J^n = R_b^n J^b R_n^b \tag{2.7}$$

If the Euler angles are chosen then the rotational kinematics is

$$\dot{\eta} = W(\eta)\omega^b \tag{2.8}$$

where

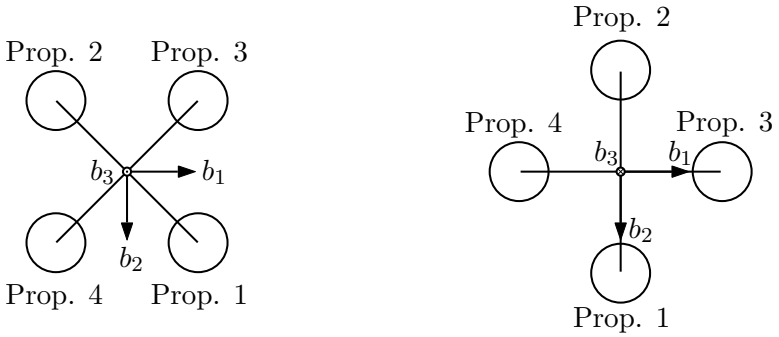
$$W(\eta) = \begin{bmatrix} 1 & \mathrm{s}_\phi \mathrm{t}_\theta & \mathrm{c}_\phi \mathrm{t}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \dfrac{\mathrm{s}_\phi}{\mathrm{c}_\theta} & \dfrac{\mathrm{c}_\phi}{\mathrm{c}_\theta} \end{bmatrix} \tag{2.9}$$

where $\mathrm{t}_\xi = \tan\xi$. For small θ and ϕ,

$$W(\eta) \approx I$$
$$\dot{\eta} \approx \omega^b$$

If quaternions are chosen then the rotational kinematics is

$$\dot{q} = \frac{1}{2}W(q)\omega^b \tag{2.10}$$



(a) "×" Quadrotor Configuration (b) "+" Quadrotor Configuration

Figure 2.3: The two main configurations of quadrotors.

where

$$W(q) = \begin{bmatrix} q_3 & -q_2 & q_1 \\ q_2 & q_3 & -q_0 \\ -q_1 & q_0 & q_3 \\ -q_0 & -q_1 & -q_2 \end{bmatrix} \tag{2.11}$$

2.1.3 Force and Torque Model

Figure 2.1 illustrates the positive directions of F^b and τ^b. We have

$$F^b = [0, 0, -\sum_{i=1}^{4} f_i]^T \qquad \text{and} \qquad f_i = k_{fi}\Omega_i^2 \tag{2.12}$$

where $\Omega_i > 0$ is the rotational speed of Propeller i and $k_{fi} > 0$ is an aerodynamic coefficient. Quadrotors can have arbitrary arm length and motor positions, but are typically built in one of two main physical configurations. Figure 2.3 shows these two configurations. In the generic case the torque applied to the vehicle is calculated by projecting the torque of each propeller into the directions b_1 and b_2.

$$\tau^b = \begin{bmatrix} \sum_{i=1}^{4} \tau_{1i} \\ \sum_{i=1}^{4} \tau_{2i} \\ k_{\tau 1}\Omega_1^2 + k_{\tau 2}\Omega_2^2 - k_{\tau 3}\Omega_3^2 - k_{\tau 4}\Omega_4^2 \end{bmatrix}^T \tag{2.13}$$

where

$$\begin{aligned} \tau_{1i} &= \ell_i f_i \sin\Theta_i \\ \tau_{2i} &= \ell_i f_i \cos\Theta_i \end{aligned} \tag{2.14}$$

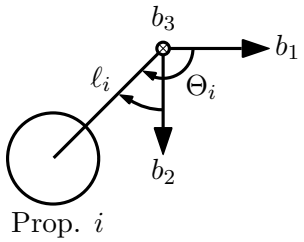


Figure 2.4: Angle definitions for torque calculations in non-square vehicles.

and ℓ_i is the distance from the motor to the centre of mass of the vehicle, $k_{\tau i} > 0$ is an aerodynamic constant, and Θ_i is the angle from the basis vector b_1 to the arm i as seen in Figure 2.4. If the vehicle is in one of the common configurations then the expression for torque can be simplified. The most common configuration used in the literature is the "+" design where the front of the vehicle (i.e., in the b_1 direction) is orientated towards Propeller 3. Then the torque is

$$\tau^b = \begin{bmatrix} \ell(f_2^b - f_1^b) \\ \ell(f_3^b - f_4^b) \\ k_{\tau 1}\Omega_1^2 + k_{\tau 2}\Omega_2^2 - k_{\tau 3}\Omega_3^2 - k_{\tau 4}\Omega_4^2 \end{bmatrix}^T \tag{2.15}$$

where $\ell = \ell_1 = \ell_2 = \ell_3 = \ell_4$, $\Theta_1 = 90°$, $\Theta_2 = -90°$, $\Theta_3 = 0$, and $\Theta_4 = 180°$. The second configuration is the "×" design which is used in this thesis. Here, the front of the vehicle is oriented between Propellers 3 and 1, and the right of the vehicle lies between Propellers 1 and 4. The torque applied to the vehicle is

$$\tau^b = \begin{bmatrix} \frac{\ell}{\sqrt{2}}(f_2 + f_3 - f_1 - f_4) \\ \frac{\ell}{\sqrt{2}}(f_1 + f_3 - f_2 - f_4) \\ k_{\tau 1}\Omega_1^2 + k_{\tau 2}\Omega_2^2 - k_{\tau 3}\Omega_3^2 - k_{\tau 4}\Omega_4^2 \end{bmatrix}^T \tag{2.16}$$

where $\Theta_1 = 45°$, $\Theta_2 = -135°$, $\Theta_3 = -45°$, and $\Theta_4 = 135°$.

The commonly used simple input models for F^b and τ^b are chosen to obtain a simple control law. These models could be generalized to include aerodynamic effects such as blade flapping and drag force which have been proven to be important experimentally [64, 65]. Furthermore in practice k_f and k_τ are not constants, but they are approximately linear close to hover. Lastly, this model neglects any gyroscopic torques due to variation in the angular momentum of the propellers. Further details regarding modelling can be found in [66–70].

2.1.4 Quadrotor Control

The control of unmanned aerial vehicles (UAVs) is typically divided up into an inner and outer loop. We adopt this structure here as shown in Figure 2.5. This two-loop

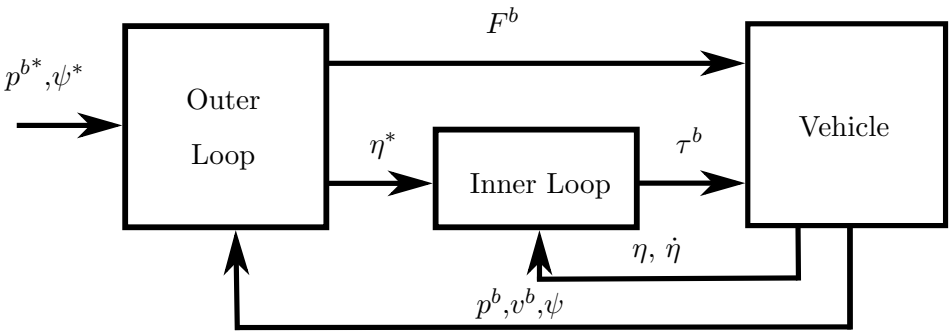


Figure 2.5: Block diagram of the inner-outer control loop structure. The inner loop controls the orientation η of the vehicle, and the outer loop provides a roll and pitch reference η^* to the inner loop in order to achieve control of lateral motion.

structure is often used independent of the sensors employed (e.g., camera or global navigation satellite system (GNSS)). The inner loop tracks an orientation reference of the vehicle and the outer loop provides a roll and pitch reference to the inner loop in order to achieve control of lateral motion. The two loop structure arises in part from a difference in the structural properties of UAVs: the outer loop has slow dynamics and the inner loop has fast dynamics. The two loop structure also provides a practical solution to motion control found in many autopilots [71, 72]. A theoretical analysis of this structure is in [73]. We define the state vector

$$x = [(p^b)^T, (v^b)^T, \eta^T, (\omega^b)^T]^T$$

At hover, the state is $x_{\text{hover}} = [(p^b)^T, 0, [0, 0, \psi], 0]^T$ and the linearized dynamics in (2.6) are

$$\dot{p^b} = v^b \tag{2.17a}$$
$$m\dot{v^b} = F^b + mg^n \tag{2.17b}$$
$$\dot{\eta} = \omega^b \tag{2.17c}$$
$$J\dot{\omega^b} = \tau^b \tag{2.17d}$$

The outer loop controls the vehicle's position p^n by applying a thrust F^b in the desired direction (θ^*, ϕ^*). It can be controlled with a PID controller with gravity

compensation.

$$\begin{aligned}\theta^* &= k_{p1}e_{p1} + k_{i1}\int_0^t e_{p1}(\tau)d\tau + k_{d1}\dot{e}_{p1} \\ \phi^* &= k_{p2}e_{p2} + k_{i2}\int_0^t e_{p2}(\tau)d\tau + k_{d2}\dot{e}_{p2} \\ F^b &= k_{p3}e_{p3} + k_{i3}\int_0^t e_{p3}(\tau)d\tau + k_{d3}\dot{e}_{p3} + mg^n\end{aligned} \tag{2.18}$$

where the error signal is

$$e_p = R_3(\psi)\left(p^{n*} - p^n\right)$$

and $p^{n*} \in \mathbb{R}^3$ is the desired position in $\mathcal{N}$, and $k_{pi} > 0$, $k_{ii} > 0$, $k_{di} > 0$, $i \in \{1, 2, 3\}$ are constant scalar control gains.

The inner loop controls the orientation η of the vehicle by applying torque τ^b. It is often controlled with a PID controller:

$$\begin{aligned}\tau_1^b &= k_{p\phi}e_\phi + k_{i\phi}\int_0^t e_\phi(\tau)d\tau + k_{d\phi}\dot{e}_\phi \\ \tau_2^b &= k_{p\theta}e_\theta + k_{i\theta}\int_0^t e_\theta(\tau)d\tau + k_{d\theta}\dot{e}_\theta \\ \tau_3^b &= k_{p\psi}e_\psi + k_{i\psi}\int_0^t e_\psi(\tau)d\tau + k_{d\psi}\dot{e}_\psi\end{aligned} \tag{2.19}$$

where $e_\eta = \eta^* - \eta$ is the error signal and $k_{pi} > 0$, $k_{ii} > 0$, $k_{di} > 0$, $i \in \{\phi, \theta, \psi\}$ are constant scalar control gains. Typically in practice, a proportional-derivative (PD) controller is used for roll and pitch i.e., $k_{i\theta} = k_{i\phi} = 0$, a proportional (P) controller is used for yaw i.e., $k_{i\psi} = k_{d\psi} = 0$. A UAV generates the force from each propeller by commanding a propeller speed Ω_i. The relationship between the propeller velocity to torque and propeller velocity to force is assumed known. The control signals are mapped to the individual propeller speeds using the vehicle's geometry as previously defined. When the vehicle is in the "+" configuration, the propeller velocities are

$$\begin{bmatrix}\Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2\end{bmatrix} = \begin{bmatrix}-\ell k_f & \ell k_f & 0 & 0 \\ 0 & 0 & \ell k_f & -\ell k_f \\ k_\tau & k_\tau & -k_\tau & -k_\tau \\ k_f & k_f & k_f & k_f\end{bmatrix}^{-1} \begin{bmatrix}\tau_1^b \\ \tau_2^b \\ \tau_3^b \\ F^b\end{bmatrix} \tag{2.20}$$

When the vehicle is in the "×" configuration, the propeller velocities are

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{-\ell k_f}{\sqrt{2}} & \frac{\ell k_f}{\sqrt{2}} & \frac{\ell k_f}{\sqrt{2}} & \frac{-\ell k_f}{\sqrt{2}} \\ \frac{\ell k_f}{\sqrt{2}} & \frac{-\ell k_f}{\sqrt{2}} & \frac{\ell k_f}{\sqrt{2}} & \frac{-\ell k_f}{\sqrt{2}} \\ k_\tau & k_\tau & -k_\tau & -k_\tau \\ k_f & k_f & k_f & k_f \end{bmatrix}^{-1} \begin{bmatrix} \tau_1^b \\ \tau_2^b \\ \tau_3^b \\ F^b \end{bmatrix} \tag{2.21}$$

We note that the above matrices are always invertible as they are full rank when $\ell, k_\tau, k_f > 0$. In summary the model assumes that the relationships from Ω_i^2 to F^b and τ^b are known and linear. We assume the propeller speeds can be controlled arbitrarily fast (i.e., there are no propellor motor system dynamics).

2.2 Computer Vision

Computer vision has been well studied in the literature, e.g., [42, 74]. This section presents the most common model for image formation and defines the image features, detectors, and descriptors. Lastly, it provides a brief introduction on VS and the homography matrix.

2.2.1 Camera Model

Figure 2.6 illustrates the commonly used pinhole camera model. The relationship between a Euclidean point p^n in $\mathcal{N}$ expressed in $\mathcal{C}$ to the projective coordinates (or homogeneous coordinates) Y^c is

$$\pi : (p^c \in \mathbb{R}^3 - \{0\} : p_3^c > 0) \to \mathbb{R}^2 \tag{2.22}$$

$$: p^c \to Ap^c \tag{2.23}$$

where $A \in \mathbb{R}^{3\times 3}$ is the intrinsic camera calibration matrix that is defined as

$$A = \begin{bmatrix} \lambda_1 & s & y_{10} \\ 0 & \lambda_2 & y_{20} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

where $\lambda_i > 0$, $i \in [1, 2]$ is the focal length of the camera in pixels, $s \in \mathbb{R}$ is the skew parameter, $y_0 \in \mathbb{R}^2$ is the principal point, and $f_\ell \in \mathbb{R}^2$ is the focal length in pixels. In camera literature the navigation frame is often referred to as *world coordinates*. The focal length in pixels is related to the focal length in meters by

$$\lambda_1 = \frac{f_\ell}{\rho_1}, \quad \text{and} \quad \lambda_2 = \frac{f_\ell}{\rho_2} \tag{2.25}$$

where $f_\ell \in \mathbb{R}$ is the focal length, and $\rho_i > 0$, $i \in [1, 2]$ is the size of the pixel in

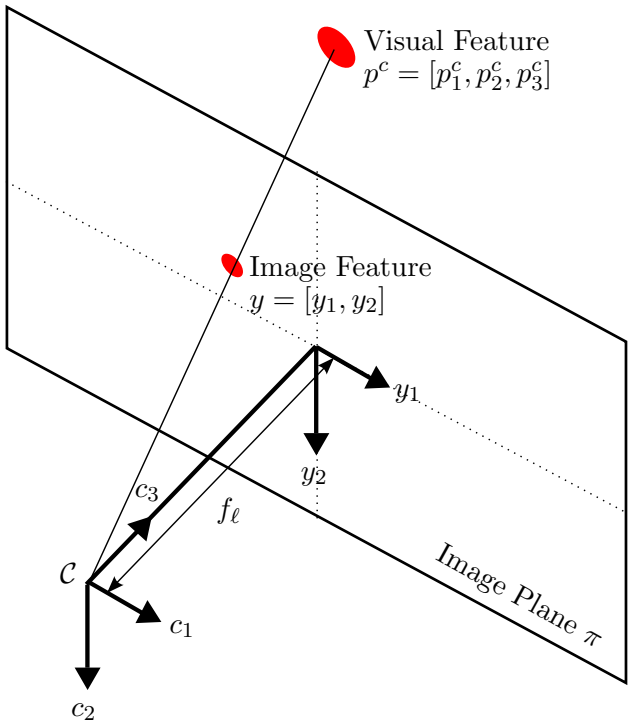


Figure 2.6: Model of a pinhole camera with a visual feature point p^c and the image feature y on the image plane π.

meters. The aspect ratio of the camera is λ_1/λ_2. In homogeneous coordinates the image coordinate Y^c is

$$Y^c \sim Ap^c$$

where $\sim$ represents equality up to a scaling of a non-zero number. In Cartesian coordinates the image coordinate Y^c is

$$\begin{aligned} y_1^c &= \frac{\lambda_1 p_1^c + s p_2^c}{p_3^c} + y_{10}^c \\ y_2^c &= \frac{\lambda_2 p_2^c}{p_3^c} + y_{20}^c \end{aligned} \tag{2.26}$$

Lens Distortion

For applications that require higher accuracy the projection model is not sufficient and lens distortion should be taken into account [75, 76]. The most common lens distortion model is radial distortion. Let π_d be the radial distortion function

$$\pi_d : r_u \to r_d \tag{2.27}$$

with

$$\frac{\partial \pi_d}{\partial r_u}(0) = 1 \tag{2.28}$$

and the function is invertible over the entire image where $r_u \in \mathbb{R} = \|y_u\|_2$, $r_d \in \mathbb{R} = \|y_d\|_2$, y_u is the undistorted image coordinate and y_d is the distorted image coordinate. The undistorted image is

$$y_u = \frac{\pi_d^{-1}(r_d)}{r_d} y_d \tag{2.29}$$

The polynomial model is (2.29) re-written as an infinite series:

$$y_u = \left(1 + \sum_{i=1}^{\infty} \kappa_i r_d^{2i}\right) y_d \tag{2.30}$$

where $\kappa_i \in \mathbb{R}$ are constant parameters. It has been shown that using just the first parameter κ_1 can achieve of accuracy of less then 0.1 pixels. Hence the truncated undistorted coordinates are

$$y_u \approx \left(1 + \kappa_1 r_d^2\right) \tag{2.31}$$

For high-distortion lenses or fish-eye lenses it may be necessary to take into account the higher order terms.

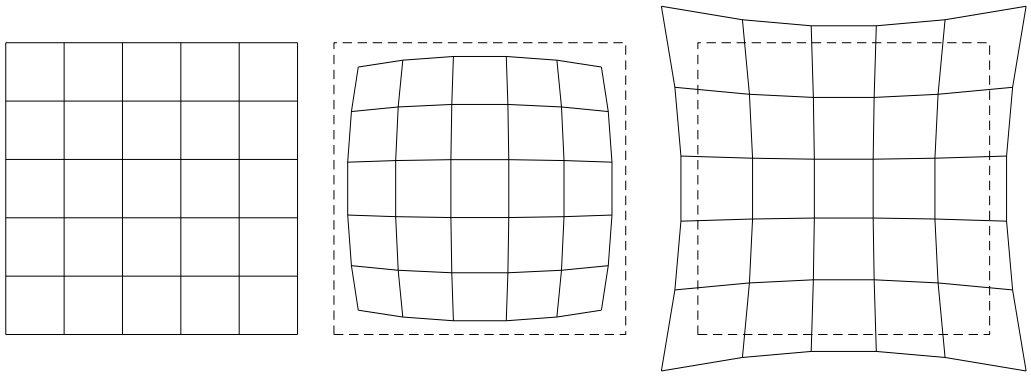
Another widely used model for fish-eye lenses is the field of view (FoV) model:

$$y_u = \frac{\tan(\kappa r_d)}{2\tan(\kappa/2)} \tag{2.32}$$

where κ is the field of view of the lens. The FoV model can also be extended by adding the higher order terms of the polynomial model. Figure 2.7 demonstrates the affect of radial distortion on straight lines in an image.

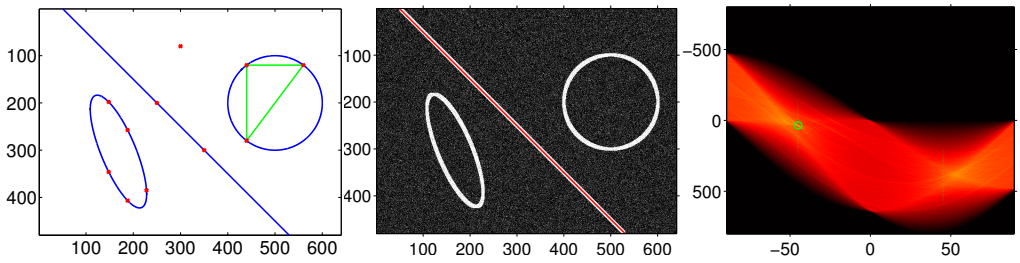
2.2.2 Image Features

Image features are the building blocks for solving a computer vision task. They contain information about an image such as point, lines, or more complicated objects. This section defines the most common image features used in VS and visual state estimation. Figure 2.8a shows an example of point features, lines features, circle features and conic features. To extract these features you need five points to determine a conic, three for a circle, two for a line, and one for a point. A point feature is extracted from an image by using colour thresholding. Figure 2.8b shows simulated colour blue thresholding of the image features with additive Gaussian noise. In the figure anywhere there is blue colour it is marked in white and everywhere else is black. After thresholding, various computer vision algorithms can determine how



(a) Undistributed Image $\kappa_1 = 0$ (b) Barrel Distortion $\kappa_1 > 0$ (c) Pincushion Distortion $\kappa_1 < 0$

Figure 2.7: Radial distortion of an image.



(a) Image Features (b) Image Feature Detection (c) Hough Transform

Figure 2.8: Some common image features are conics, circle, lines and points. Colour thresholding along with the Hough transform can be used to detect line features.

many blobs there are, the size of the blobs and other useful information such as their moments. A point image feature can be centre of a colour blob. Similarly, a line feature can be computed as the line between the centre of two colour blobs. However, if there is an actual line in the image is it more common to extract the line feature using a Hough transform. Hough transforms can also be used to extract circles and conics, however with each degree of freedom (DoF) the complexity increases exponentially. Hence, the Hough transform is typically not used for anything but lines or circles. Figure 2.8c shows the Hough transform of the image using a "hot" colour map, i.e., areas where it is more probable to have a line will be white and least probable will be black. In Figure 2.8c the highest probable spot where there is a line is highlighted in green. This line is then redrawn on top of Figure 2.8b in red. More advanced image features are typically calculated from basic image features. For example, you can obtain a plane or a cube from a set of points. Similarly, if the physical dimensions of a complex object are known, e.g., a car, it can be extracted from just a few points.

Point Features

One of the simplest image features is a point. From (2.26), given a 3 D point p^c expressed in the camera frame, the projection of p^c onto the image plane π is a 2 D point feature y with coordinates

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 \frac{p_1^c}{p_3^c} + y_{10} \\ \lambda_2 \frac{p_2^c}{p_3^c} + y_{20} \end{bmatrix} \tag{2.33}$$

Here we have take the camera skew $s = 0$ which is a good approximation for most modern cameras. The single point image feature kinematics can be found by differentiating (2.33) with respect to time [3]:

$$\dot{y} = L\nu = \begin{bmatrix} L_v & L_\omega \end{bmatrix} \begin{bmatrix} v^c \\ \omega^c \end{bmatrix} \tag{2.34}$$

where L is the interaction matrix or image Jacobian, ν is the velocity screw of the camera in $\mathcal{C}$, and

$$L_v = \begin{bmatrix} -\frac{\lambda_1}{p_3} & 0 & \frac{y_1 - y_{10}}{p_3} \\ 0 & -\frac{\lambda_2}{p_3} & \frac{y_2 - y_{20}}{p_3} \end{bmatrix} \tag{2.35}$$

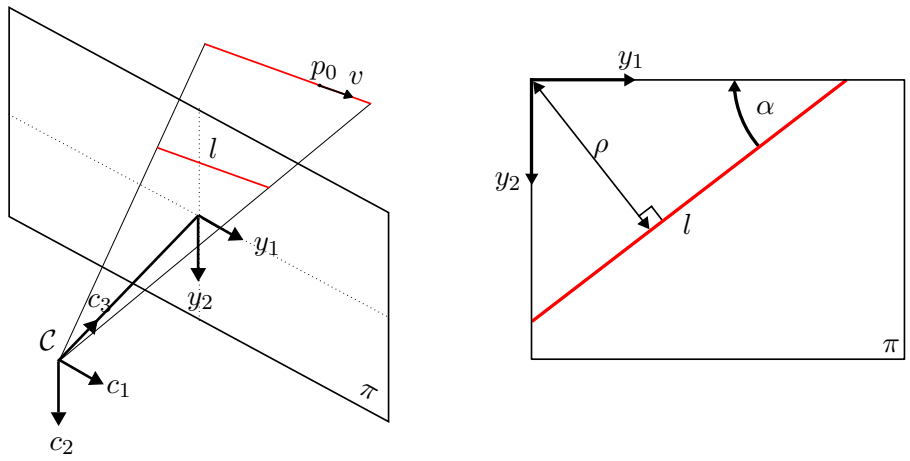
$$L_\omega = \begin{bmatrix} \frac{(y_1 - y_{10})(y_2 - y_{20})}{\lambda_2} & -\frac{(y_1 - y_{10})^2 + \lambda^2}{\lambda_1} & \frac{\lambda_1}{\lambda_2}(y_2 - y_{20}) \\ \frac{(y_2 - y_{20})^2 + \lambda^2}{\lambda_2} & -\frac{(y_1 - y_{10})(y_2 - y_{20})}{\lambda_1} & -\frac{\lambda_2}{\lambda_1}(y_1 - y_{10}) \end{bmatrix} \tag{2.36}$$

Line Features

Another basic feature is the line feature. In projective geometry a line is the dual of a point. i.e., the role of points and lines can be swapped in axioms or theories about the properties of lines and points. A 3 D line can be represented by a base point p_0 and a vector v that represents the direction of the line. Equivalently, this can also be seen as the intersection of two planes as seen in Figure 2.9a.

$$\begin{cases} A_1 p_1 + B_1 p_2 + C_1 p_3 + D_1 = 0 \\ A_2 p_1 + B_2 p_2 + C_2 p_3 + D_2 = 0 \end{cases} \tag{2.37}$$

We exclude the degenerate case when $D_1 = D_0$. This happens when the line passes through the origin of the camera and the camera would only see a point in the



(a) A 3 D line projected onto the image plane π.

(b) A 2 D line l on the image plane π.

Figure 2.9: Line image feature

image. Any homogeneous image point $Y \in \mathbb{R}^3$ that is on the line $\bar{l} \in \mathbb{R}^3$ satisfies the orthogonality equation.

$$\bar{l}^T Y = 0 \tag{2.38}$$

We choose to parameterize the line as $l = (\alpha, \rho)$ where

$$\bar{l}^T Y = \begin{bmatrix} \mathrm{s}_\alpha & \mathrm{c}_\alpha & -\rho \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = y_1 \mathrm{s}_\alpha + y_2 \mathrm{c}_\alpha - \rho = 0 \tag{2.39}$$

We note that this representation is ambiguous as the same line can be parameterized by $(\alpha + 2k\pi, \rho)$ or $(\alpha + (2k+1)\pi, -\rho)$ where $k \in \mathbb{Z}$. Without loss of generality α can be restricted to lie in $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$. The positive direction of α is from the line to the axis y_1 as is shown in Figure 2.9b. We note that the typical parameterization of a line as $y_2 = my_1 + b$ is a poor choice of parameterization as it cannot represent vertical lines.

Next the line kinematics are derived by differentiating (2.39) with respect to time.

$$\dot{l} = L_\ell \nu = \begin{bmatrix} L_{v\ell} & L_{\omega\ell} \end{bmatrix} \begin{bmatrix} v^c \\ \omega^c \end{bmatrix} \tag{2.40}$$

where

$$L_{v\ell} = \left(\frac{1}{D_i}\right) \begin{bmatrix} -A_i \mathrm{c}_\alpha + B_i \mathrm{s}_\alpha & 0 \\ 0 & A_i \rho \mathrm{s}_\alpha + B_i \rho \mathrm{c}_\alpha + C_i \end{bmatrix} \begin{bmatrix} \mathrm{s}_\alpha & \mathrm{c}_\alpha & -\rho \\ \mathrm{s}_\alpha & \mathrm{c}_\alpha & -\rho \end{bmatrix}$$

$$L_{\omega\ell} = \begin{bmatrix} \rho \mathrm{s}_\alpha & \rho \mathrm{c}_\alpha & 1 \\ (1+\rho^2)\mathrm{c}_\alpha & -(1+\rho^2)\mathrm{s}_\alpha & 0 \end{bmatrix}$$

where

$$i = \begin{cases} 1, & \text{if } D_1 \neq 0 \\ 2, & \text{otherwise} \end{cases}$$

Feature Detectors and Descriptors

A feature is a point of interest in the image based on some criterion, e.g., a corner. There many different features used in computer vision (CV). Some of the most common features detectors are good features to track (GFTT) [77], scale-invariant feature transform (SIFT) [78], features from accelerated segment test (FAST) [79], speeded up robust feature (SURF) [80], center surround extremas (CenSurE) [81], adaptive and generic accelerated segment test (AGAST) [82], and oriented FAST and rotated BRIEF (ORB) [83]. Very closely related to the feature detectors are feature descriptors. A feature descriptor is a vector of values that describe an image patch around a image feature. A descriptor can be as simple as the image coordinates of the feature or it can be more complicated. Some of the most used descriptors are SIFT, SURF, ORB, binary robust independent elementary features (BRIEF) [84], binary robust invariant scalable keypoints (BRISK) [85], and learned attachment of three patch codes (LATCH) [86]. As can be seen from the list some of the algorithms (SIFT, SURF, and ORB) are both feature detectors and descriptors. Often in the literature both the feature detector and descriptors will be referred to together as simply an image feature.

2.2.3 Visual Servoing

It is well established that computer vision is a natural sensor for increasing a robot's autonomy in uncertain environments as it provides information about the actual situation [42]. In the case of UAVs, this has led to research on using computer vision as a replacement for GNSS for localizing the vehicle and motion control. *Visual servoing* (VS) refers to motion control which relies on computer vision. For example, visual servoing was applied to autonomous landing of a UAV in [87] and to pole inspection in [88].

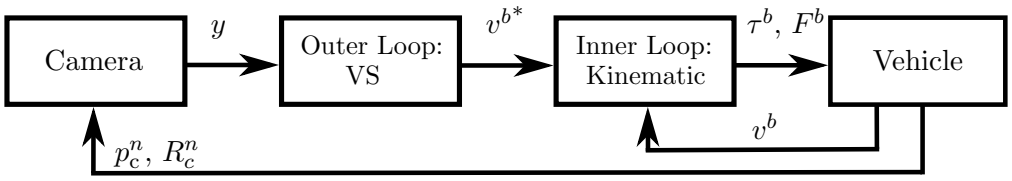


Figure 2.10: Block diagram of the traditional inner-outer control loop structure for visual servoing.

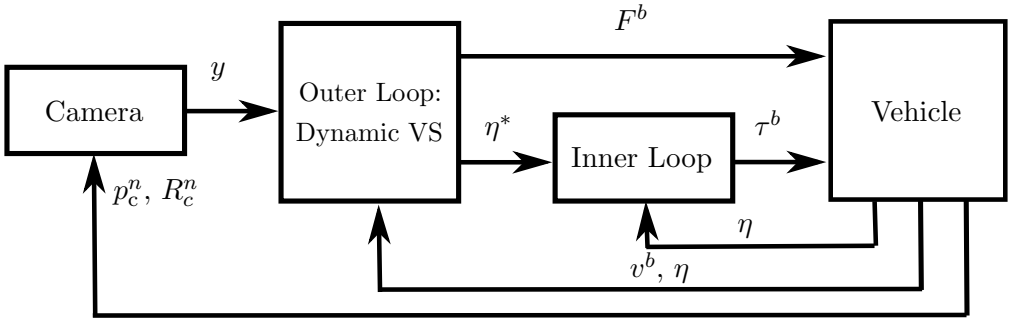


Figure 2.11: Block diagram of dynamic visual servoing with a inner-outer control loop structure.

Typically, the control structure of VS is divided into two loops [3]. As seen in Figure 2.10, an inner loop accepts a reference velocity and determines force and torque inputs to control the robot's velocity at a relatively high bandwidth (e.g., 200 Hz). An outer loop uses video camera feedback to generate reference velocities for the inner loop. It operates at a relatively low frequency due to the low rate at which images are acquired and processed (e.g., 30 Hz). As discussed in Chapter 1, VS is divided into two main approaches: position-based visual servoing (PBVS) and image-based visual servoing (IBVS) [3]. In PBVS the pose of the vehicle is extracted from the image and the motion control of the vehicle can be done using typical motion control techniques. In this case the vision system can be thought of as a generic position sensor similar to GNSS or a laser proximity sensor. On the other hand, IBVS minimizes an error function which is directly computed from features in the image plane. Both IBVS and PBVS approaches have distinct advantages and disadvantages depending on the application considered.

A classical IBVS for a fully actuated 6 DoF vehicle solves for the desired velocity screw of the vehicle by using at least three point features. The interaction matrix in (2.34) for each point are stacked and the stacked matrix is inverted. Such an approach can lead to singularities and it neglects the dynamics of the vehicle. For the case of an underactuated UAV, angular and linear velocity cannot be independently specified. It is assumed that the vehicle can perfectly track the reference velocity and design is based on a purely kinematic vehicle model. However, the importance

of including the dynamics of the vehicle is underlined in [47, 89] for both high speed tasks and underactuated systems.

The configuration space of a quadrotor is six-dimensional whereas it only has four inputs. Hence, it is underactuated. The roll and pitch are directly coupled to translational acceleration which makes makes it difficult to apply a conventional kinematics-based visual servoing design. Hence, we include the vehicle's linear velocity dynamics in the control design. We refer to visual servoing that directly accounts for vehicle dynamics as *dynamic visual servoing*. A block diagram of dynamic VS is shown in Figure 2.11. Dynamic image-based visual servoing (DIBVS) approaches for UAVs that have appeared in the literature include spherical image moment-based designs [39, 41, 90–92], homography based methods [43, 44, 93], a virtual spring approach [45], virtual camera approaches [13, 46, 47], and a nonlinear state transformation approach [54]. Although dynamic visual servoing has clear practical significance, few published works have thorough experimental results. In Chapter 4 we propose a number of IBVS controls and present their experimental validation.

2.2.4 Homography

A useful tool for IBVS is the homography matrix which embeds information about the relative pose of a camera given two images of the same planar object [42]. Following [42, 74], we consider two views of a point on a planar target as shown in Figure 2.12. The two camera frames $\mathcal{C}^1 = \{c_1^1, c_2^1, c_3^1\}$ and $\mathcal{C}^2 = \{c_1^2, c_2^2, c_3^2\}$ are related by a translation r^1 and rotation R_1^2. We denote the image plane coordinates of n points in the two cameras to be $y_i^1, y_i^2 \in \mathbb{R}^2, 1 \leq i \leq n$. The homogeneous representations of these vectors are denoted $Y_i^1, Y_i^2 \in \mathbb{R}^3, 1 \leq i \leq n$. A homography mapping $H \in \mathbb{R}^{3\times 3}$ is a matrix which transforms Y_i^1 to Y_i^2 according to

$$Y_i^2 \sim HY_i^1$$

where $\sim$ represents equality up to a nonzero scaling. We note that the matrix H can be arbitrarily scaled, therefore it has eight degrees of freedom. When a homography is induced by a static plane it can be decomposed into a translation r^1 and a rotation R_1^2 assuming that the optical centre of the camera never passes through the plane. Such a situation is physically impossible as the camera would have to pass through the target. That is, we have

$$H = A\left(R_1^2 + \frac{1}{d^1} r^1 {n^1}^T\right) A^{-1} \tag{2.41}$$

where A is the matrix of intrinsic camera parameters, d^1 is the perpendicular dis-

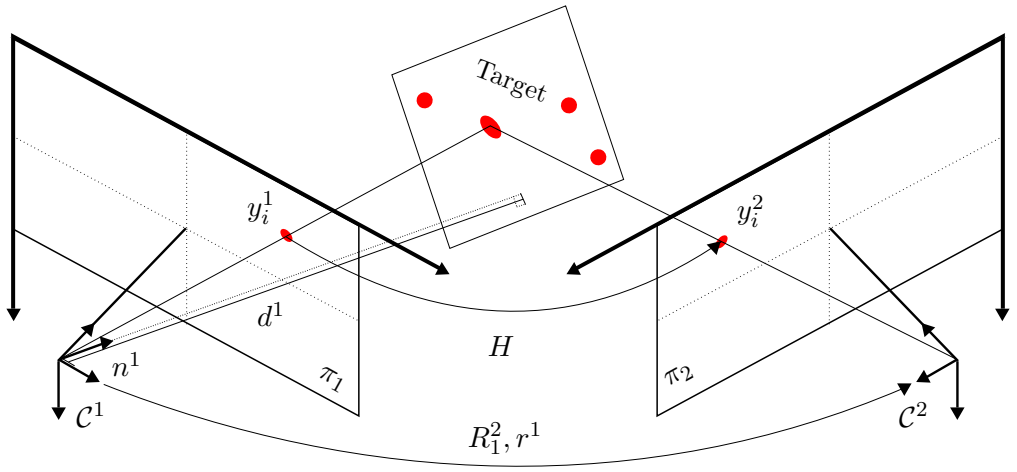


Figure 2.12: A homography matrix H transforms the image coordinates $Y_i^1 \in \mathbb{R}^3$ into $Y_i^2 \in \mathbb{R}^3$ of two views of the same 3 D point.

tance from the camera frame $\mathcal{C}^1$ to the target plane, n^1 is the unit normal of the target plane expressed in $\mathcal{C}^1$, r^1 is the displacement of the origin of $\mathcal{C}^2$ relative to the origin of $\mathcal{C}^1$ expressed in $\mathcal{C}^1$, and R_1^2 is the rotation matrix which describes the relative orientation of $\mathcal{C}^1$ and $\mathcal{C}^2$. Without loss of generality we take $A = I$ which assumes camera calibration has been performed. In practice, cameras can be easily calibrated to subpixel error. As an alternative, the points in the images can be normalized using a similarity transform that moves the centroid of the points in the images to the origin and scales the average distance of the points from the origin to $\sqrt{2}$. This approach yields acceptable results in practice. Normalization, via either the camera's intrinsic parameters or by a similarity transform, is an important step of the algorithm to provide suitable accuracy in the estimate of the homography [74].

Homography Composition

The homography

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

can be calculated using at least $n \geq 4$ co-planar point correspondences in the two images using the planar homography constraint

$$\text{sk}(Y_i^2) H Y_i^1 = 0, \quad 1 \leq i \leq n \tag{2.42}$$

Since the constraint is linear we can rewrite it as

$$\chi H^s = 0 \tag{2.43}$$

where $H^s = [H_{11}, H_{21}, H_{31}, H_{12}, H_{22}, H_{32}, H_{13}, H_{23}, H_{33}]^T$,

$$\chi = \begin{bmatrix} \text{kron}(Y_1^1, \text{sk}(Y_1^2))^T \\ \text{kron}(Y_2^1, \text{sk}(Y_2^2))^T \\ \vdots \\ \text{kron}(Y_n^1, \text{sk}(Y_n^2))^T \end{bmatrix} \in \mathbb{R}^{3n \times 9} \tag{2.44}$$

and $\text{kron}(A, B) \in \mathbb{R}^{mk \times nl}$ is the Kronecker product of $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{k \times l}$:

$$\text{kron}(A, B) = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \tag{2.45}$$

To solve (2.43) uniquely (up to a scale factor) we need $\text{rank}(\chi) = 8$. This is only possible if there exists a set of 4 points out of n on the plane such that no three of them are co-linear. Since Y_i^j contains noise we find the optimal χ by solving

$$\min_{||H^s||_2=1} ||\chi H^s||_2 \tag{2.46}$$

using linear least squares to obtain H^s up to a scale factor. To solve (2.46), we compute the SVD of $\chi = U\Sigma V^T$ where $U, V \in \mathbb{R}^{9 \times 9}$ are orthogonal,

$$\Sigma = \text{diag}([\sigma_1, \ldots, \sigma_9]) \in \mathbb{R}^{9 \times 9}$$

and $\sigma_1 \geq \cdots \geq \sigma_9 > 0$. Since V is orthogonal and defining $\xi = V^T H^s$ then

$$||H^s||_2 = ||VV^T H^s||_2 = ||V\xi||_2 = ||\xi||_2$$

Hence,

$$\begin{aligned} \min_{||H^s||_2=1} ||\chi H^s||_2^2 &= \min_{||\xi||_2^2=1} ||U\Sigma\xi||_2^2 = \min_{||\xi||_2^2=1} ||\Sigma\xi||_2^2 \\ &= \min_{||\xi||_2^2=1} \sigma_1^2\xi_1^2 + \cdots + \sigma_9^2\xi_9^2 = \sigma_9^2 \end{aligned}$$

Thus the minimizing $\xi = [0, 0, \ldots, 0, 1]^T$ and

$$H_l^s = V[0, 0, \ldots, 0, 1]^T, \tag{2.47}$$

i.e., the ninth column of V. We form a scaled homography matrix H_l from solution H_l^s, i.e.,

$$H = \lambda H_l \tag{2.48}$$

The magnitude of scale factor λ is given by

$$|\lambda| = \sigma_2(H_l) \tag{2.49}$$

where $\sigma_2(H_l)$ is the second largest singular value of H_l. To ensure the sign of H is correct we verify the constraint

$$(Y_i^2)^T H Y_i^1 > 0, \quad 1 \leq i \leq K \tag{2.50}$$

Homography Decomposition

There are multiple algorithms to decompose a homography matrix [42, 74, 94]. Here we will use a modified version of the SVD algorithm. There are four mathematical solutions to the decomposition of H into R_1^2, $\frac{1}{d^1}r^1$, and n^1. Only two of these solutions are physically meaningful, and to choose the correct solution additional information about the scene is required.

Given that $H^T H$ is symmetric we can diagonalize it as

$$H^T H = V \Sigma V^T \tag{2.51}$$

where $V = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \in \mathrm{SO}(3), v_i \in \mathbb{R}^3$, and $\Sigma = \mathrm{diag}([\sigma_1^2, \sigma_2^2, \sigma_3^3])$, where $\sigma_i, 1 \leq i \leq 3$ denote the singular values of the estimated H. Next, we define two unit length vectors u_1 and u_2 such that their length is preserved under the map H and that $\{v_2, u_1, \mathrm{sk}(v_2)u_1\}$ and $\{v_2, u_2, \mathrm{sk}(v_2)u_2\}$ are orthonormal bases of $\mathbb{R}^3$. We have the expressions

$$\begin{aligned} u_1 &= \frac{\sqrt{1-\sigma_3^2}v_1 + \sqrt{\sigma_1^2 - 1}v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \\ u_2 &= \frac{\sqrt{1-\sigma_3^2}v_1 - \sqrt{\sigma_1^2 - 1}v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \end{aligned} \tag{2.52}$$

We define the following matrices

$$\begin{aligned} U_1 &= \begin{bmatrix} v_2 & u_1 & \mathrm{sk}(v_2)u_1 \end{bmatrix} \\ U_2 &= \begin{bmatrix} v_2 & u_2 & \mathrm{sk}(v_2)u_2 \end{bmatrix} \\ W_1 &= \begin{bmatrix} Hv_2 & Hu_1 & \mathrm{sk}(Hv_2)Hu_1 \end{bmatrix} \\ W_2 &= \begin{bmatrix} Hv_2 & Hu_2 & \mathrm{sk}(Hv_2)Hu_2 \end{bmatrix} \end{aligned} \tag{2.53}$$

$\boldsymbol{i}$	$\boldsymbol{R^2_{1i}}$	$\boldsymbol{n^1_i}$	$\boldsymbol{(1/d^1)r^1_i}$
1	$W_1U_1^T$	$\mathrm{sk}(v_2)u_1$	$(H - W_1U_1^T)\,\mathrm{sk}(v_2)u_1$
2	$W_2U_2^T$	$\mathrm{sk}(v_2)u_2$	$(H - W_2U_2^T)\,\mathrm{sk}(v_2)u_2$
3	$W_1U_1^T$	$\mathrm{sk}(u_1)v_2$	$(W_1U_1^T - H)\,\mathrm{sk}(v_2)u_1$
4	$W_2U_2^T$	$\mathrm{sk}(u_2)v_2$	$(W_2U_2^T - H)\,\mathrm{sk}(v_2)u_2$

Table 2.1: Four solutions of the decomposition of $H = R_1^2 + \frac{1}{d^1} r n^{1T}$ into $\{R^2_{1i}, \frac{1}{d^1} r^1_i, n^1_i\}, i = 1, 2, 3, 4$.

such that

$$R_1^2 U_1 = W_1$$
$$R_1^2 U_2 = W_2$$

Due to the sign ambiguity in $\frac{1}{d^1} r^1 n^{1T}$ we obtain four decompositions of $H = R_1^2 + \frac{1}{d^1} r^1 n^{1T}$ denoted $\{R_i, \frac{1}{d^1} r^1_i, n^1_i\}, i = 1, 2, 3, 4$ as shown in Table 2.1. From these four solutions we can extract two relevant solutions using the positive depth constraint

$$(n^1)^T e_3 > 0 \tag{2.54}$$

to obtain two solutions.

The control method requires the scaled relative pose between the desired and current frames. We describe two methods for picking the correct solution from the homography decomposition data. The first method computes the homography between consecutive images and relies on the fact that images vary continuously in time. This implies the correct solution for R_1^2, n^1 and $\frac{1}{d^1} r^1$ are also continuous in time. The incorrect solution has no guarantee of continuity and can make large jumps. Initially we pick one of the solutions arbitrarily which we assume is correct. We monitor both solutions for discontinuities in the parameters. If one of the solutions exhibits a discontinuity, we can determine the correct solution. This method suffers from the two limitations. First, it requires slow movement in the image and a fast frame rate. Second, the two solutions can meet making it impossible to determine the correct solution for subsequent time. In this case, the method is reset and we must arbitrarily choose a solution. The method relies on the robustness of the control law to incorrect choices of the solution.

In this work we chose a simpler method to picking the correct solution which relies on a known target plane normal vector n^1 in the initial image. Rather than calculate the homography between consecutive frames we compute the homography between the initial and current images. This helps eliminate any accumulation of error in the estimate of the scaled relative pose. The choice of correct solution is determined by comparing the estimated normal vector with its known value. Assum-

ing a known target image and normal is not restrictive in that tracking algorithms require an initial image. We require additionally that the user specifies the initial target orientation. We remark that the method does not require any geometric information about the target and maintains the benefits of IBVS given that the control law is still calculated directly in the image plane.

Chapter 3

Experimental Platform

In order to efficiently perform experimental research on nonlinear control and visual servoing of unmanned aerial vehicles (UAVs), we developed the indoor Applied Nonlinear Control Lab (ANCL) quadrotor platform at the University of Alberta in Edmonton, Canada. The design for our experimental platform was inspired by:

- Eidgenössische Technische Hochschule Zürich's (ETHZ's) "The Flying Machine Arena" [95, 96]
- University of Pennsylvania's GRASP Laboratory [97]
- Massachusetts Institute of Technology's (MIT's) Aerospace Controls Laboratory (ACL) [98]

Over the course of this PhD the hardware has undergone various revisions. In general an emphasis has been focused on open-source and open-hardware platforms that have been developed as research platforms.

3.1 Indoor Quadrotor Platform

The ANCL indoor quadrotor UAV platform consists of a laboratory, a motion capture system (MCS), various autopilots, quadrotors, computer vision systems, and cameras. The flight volume is 4 m x 5 m x 2 m and has an eight camera Bonita 3 Vicon MCS surrounding it. Each MCS camera has a frame rate of 240 frames per second (fps) and a resolution of 640×480 pixel resolution. The overall Vicon system has a 2 ms delay and can calculate the vehicle pose at up to 200 Hz with millimetre accuracy. A ground station computer runs QGroundControl (QGC) [99] which is an open-source ground control software which is intended to be used with any Mavlink enable drone. QGC is used to setup the UAV (e.g., parameter values and attitude and heading reference system (AHRS) calibration), visualize UAV sensor data, and perform mission planning.

Quadrotor I

Parameter	Value
m	1.6 kg
J_1	0.03 kg·m^2
J_2	0.03 kg·m^2
J_3	0.05 kg·m^2
ℓ	0.25 m
$\Theta_1, -\Theta_3$	45°
$-\Theta_2, \Theta_4$	135°

Quadrotor II

Parameter	Value
m	2.3 kg
ℓ	0.28 m
$\Theta_1, -\Theta_3$	57°
$-\Theta_2, \Theta_4$	123°

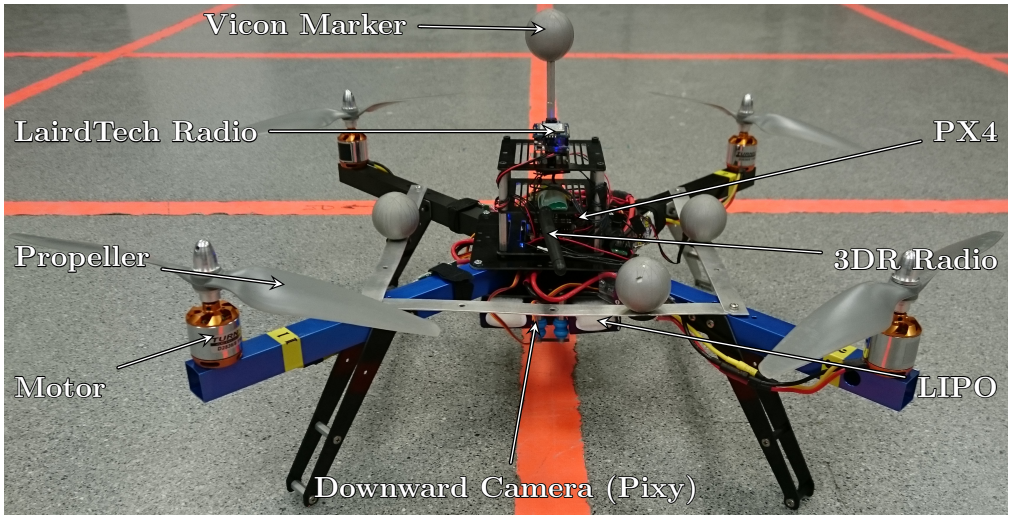
Table 3.1: Experimental platform quadrotor parameters.

3.1.1 Quadrotors

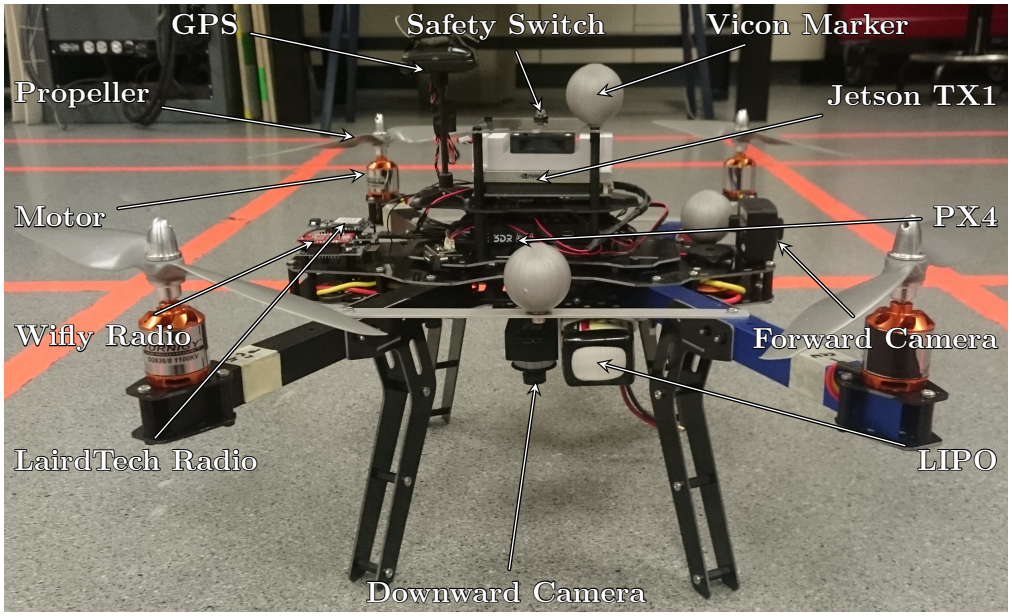
We custom built two quadrotor systems using 3D Robotics quadrotor do-it-yourself (DIY) frames. The vehicles are shown in Figure 3.1. The arms are aluminum, and the legs and body are carbon fibre. The quadrotors are equipped with 8 channel Spektrum satellite receiver paired with a Spektrum DX8 radio transmitter for manual control. The radio is shown in Figure 3.2. Pulse width modulation (PWM) outputs drive four Afro 30 A electronic speed controllers (ESCs) which power four Turnigy 1100 KV Brushless Outrunner Motors and four APC 11 inch multi-rotor propellers. The ESCs run SimonK firmware which is open-source ESC firmware designed for faster response time and improved multi-rotor performance [100]. Power is supplied by a Turnigy 3 cell 5000 mAh lithium polymer battery (LiPo) battery. An onboard LairdTech 2.4 GHz radio communicates with a personal computer (PC) running the Vicon Tracker software which computes the vehicle pose and computes a using a low pass-filtered finite difference velocity estimate by low pass differentiating. This data is received by the autopilot which is interfaced to a matching onboard LairdTech radio. The ANCL quadrotors I and II, as seen in Figure 3.1, weigh about 1.6 kg and 2.3 kg, respectively, including the battery, the vision computer and cameras. They measure 25 cm and 28 cm, respectively, from the centre of the vehicle to the motor shafts. Table 3.1 summarizes the quadrotor model parameters.

3.1.2 Autopilots

The autopilot is based on Pixhawk autopilot hardware which is an open-hardware project which runs the open source PX4 software [72]. In particular we use the PX4FMUv1 (Pixhawk autopilot - flight management unit - version 1) and the PX4IO (Pixhawk Input/Output Module). The PX4FMUv1 contains a 168 MHz ARM processor, a 3 D accelerometer, two 3 D gyroscope, a 3 D magnetometer, and a pressure sensor. The PX4FMU and PX4IO have a number of different hardware interfaces such universal asynchronous receiver/transmitter (UART), inter-integrated circuit



(a) The first quadrotor: Quadrotor I



(b) The second quadrotor: Quadrotor II

Figure 3.1: The ANCL quadrotor platform.

Figure 3.2: The Spektrum DX8 remote transmitter

(I^2C), and PWM outputs. Later on the PX4FMUv1 and PX4IO were repackaged into a single board called the Pixhawk Autopilot or the PX4FMUv2. In this thesis we refer to the PX4FMUv2 as PX4. See Figure 3.3 for an overview of the hardware interconnection of the various components and Appendix A for an explanation of the labels used in the figure. The PX4FMUv1 and PX4FMUv2 are shown in Figure 3.4.

One of the main purposes of an autopilot is implement the vehicle's control laws. In practice it can be difficult to implement control laws exactly the same as in theory. One important difference is that outputs are often normalized. e.g., a quadrotor can not directly apply a force, instead the controller will output a normalized thrust signal. This signal controls each motor between zero and maximum propeller speed. Starting with the outer control loop as in (2.18) the control law becomes

$$u_f = k_{p3} e_{p3} + k_{i3} \int_0^t e_{p3}(\tau) d\tau + k_{d3} \dot{e}_{p3} + u_g \tag{3.1}$$

where $u_f \in [0, 1]$ is the normalized thrust control signal and $u_g \in [0, 1]$ is the normalized gravity compensation. We have reused the gains k_{p3}, k_{i3}, k_{d3} in (2.18) for convenience. The values of the gains in (2.18) and (3.1) will be different for a similar motion control response. Similarly, the inner loop as in (2.19) is normalized.

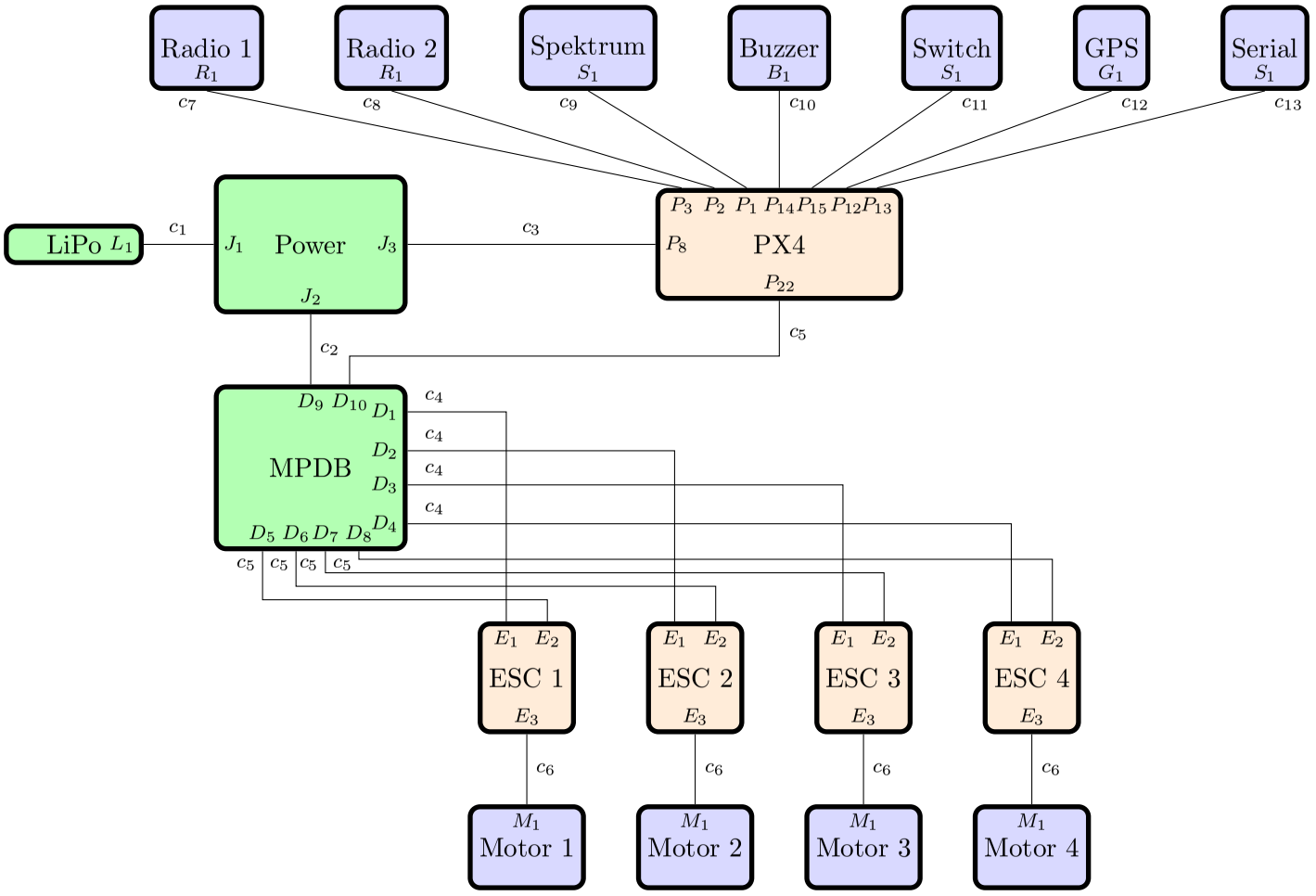
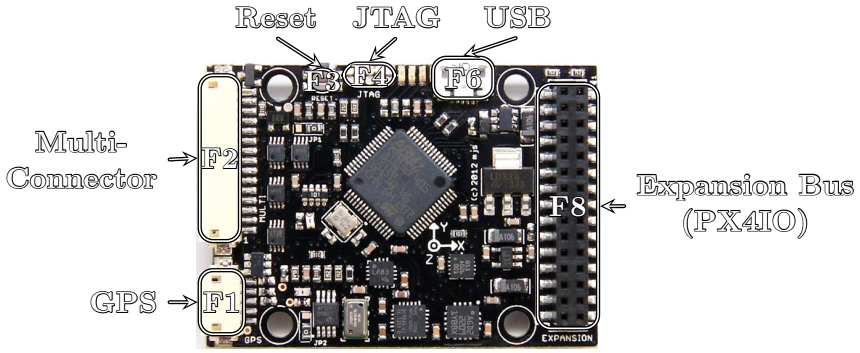
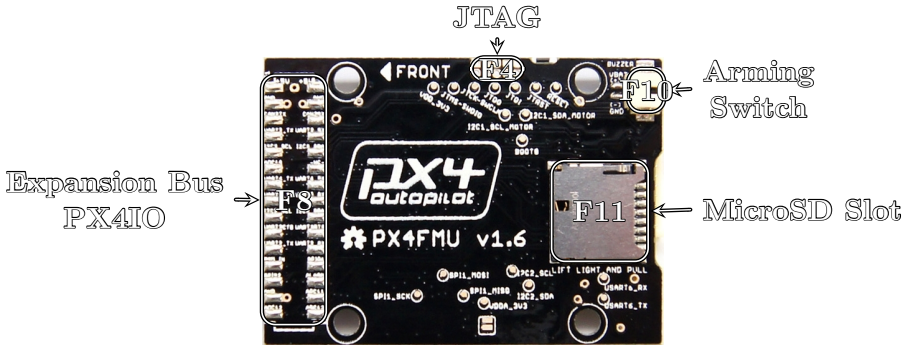


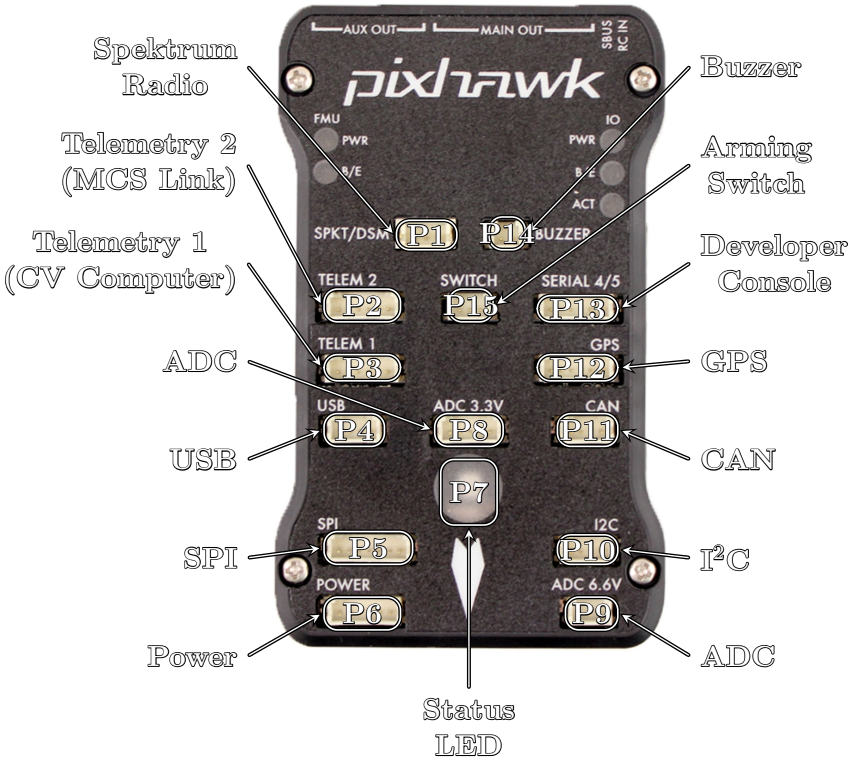
Figure 3.3: Wiring diagram of the hardware interconnection.



(a) PX4FMUv1 - Top



(b) PX4FMUv1 - Bottom



(c) PX4FMUv2

Figure 3.4: Pixhawk autopilot - flight management unit.

$$
\begin{aligned}
u^b_{\tau 1} &= k_{p\phi} e_\phi + k_{i\phi} \int_0^t e_\phi(\tau) d\tau + k_{d\phi} \dot{e}_\phi \\
u^b_{\tau 2} &= k_{p\theta} e_\theta + k_{i\theta} \int_0^t e_\theta(\tau) d\tau + k_{d\theta} \dot{e}_\theta \\
u^b_{\tau 3} &= k_{p\psi} e_\psi + k_{i\psi} \int_0^t e_\psi(\tau) d\tau + k_{d\psi} \dot{e}_\psi
\end{aligned} \tag{3.2}
$$

Next the control signals are mixed together to form the normalized propeller control signals u_Ω. When the vehicle is in the"+" configuration, the propeller velocities are

$$
\begin{bmatrix} u^2_{\Omega 1} \\ u^2_{\Omega 2} \\ u^2_{\Omega 3} \\ u^2_{\Omega 4} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{\tau 1} \\ u_{\tau 2} \\ u_{\tau 3} \\ u_f \end{bmatrix} \tag{3.3}
$$

For the ANCL quadrotor I or any vehicle in the "×" configuration, the propeller velocities are

$$
\begin{bmatrix} u^2_{\Omega 1} \\ u^2_{\Omega 2} \\ u^2_{\Omega 3} \\ u^2_{\Omega 4} \end{bmatrix} = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 1 & 1 \\ 1/\sqrt{2} & -1/\sqrt{2} & 1 & 1 \\ 1/\sqrt{2} & 1/\sqrt{2} & -1 & 1 \\ -1/\sqrt{2} & -1/\sqrt{2} & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{\tau 1} \\ u_{\tau 2} \\ u_{\tau 3} \\ u_f \end{bmatrix} \tag{3.4}
$$

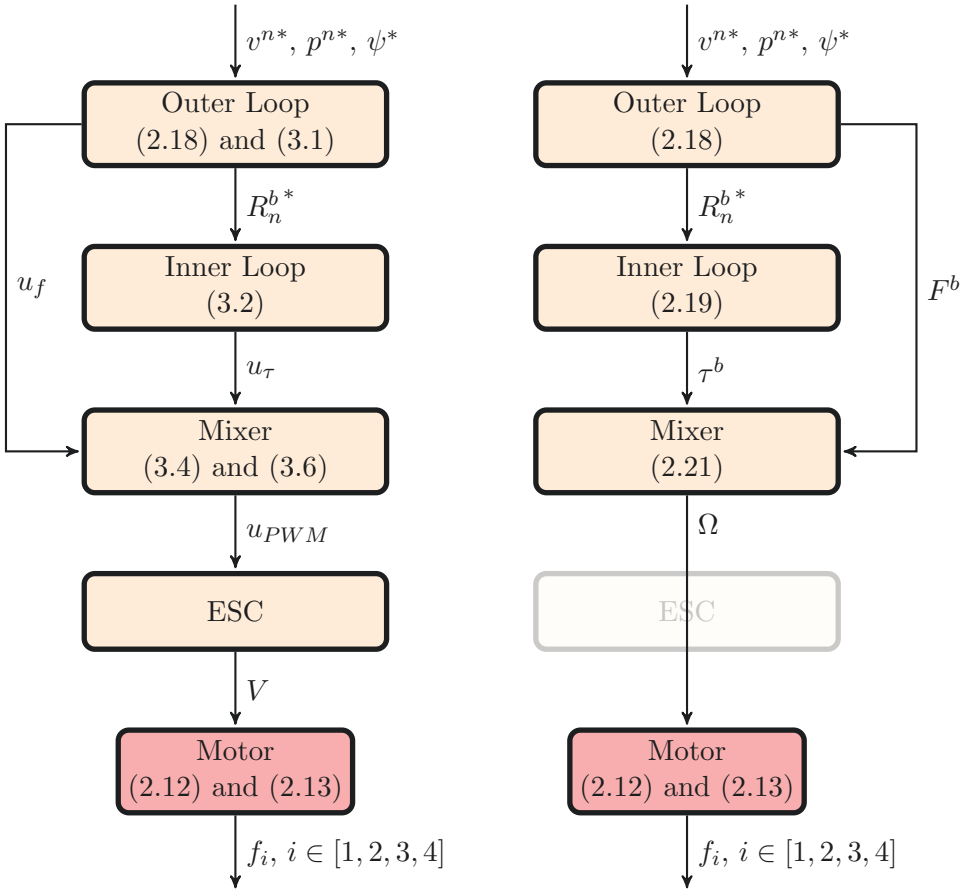
Lastly, for the ANCL quadrotor II, the propeller velocities are

$$
\begin{bmatrix} u^2_{\Omega 1} \\ u^2_{\Omega 2} \\ u^2_{\Omega 3} \\ u^2_{\Omega 4} \end{bmatrix} = \begin{bmatrix} \sin(-56.6^\circ) & \cos(-56.6^\circ) & 1 & 1 \\ \sin(123.4^\circ) & \cos(-123.4^\circ) & 1 & 1 \\ -\sin(-123.4^\circ) & -\cos(-123.4^\circ) & -1 & 1 \\ -\sin(56.6^\circ) & -\cos(56.6^\circ) & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{\tau 1} \\ u_{\tau 2} \\ u_{\tau 3} \\ u_f \end{bmatrix} \tag{3.5}
$$

The above mixing can cause the propeller control signals to saturate which can result in undesired behaviour of the quadrotor. Instead of a hard saturation, all of the propeller velocities are re-scaled so that the saturating actuator is limited to 1. Lastly, the propeller control signal is converted to a PWM signal.

$$
u_{\text{PWM}} = \frac{k u_\Omega + \min}{\max - \min} \tag{3.6}
$$

where k is a positive constant scaling factor and min and max are the minimum and maximum PWM signals. We note that in practice typical values are $k = 1000$, $\min = 1000$ and $\max = 2000$ and $u_{\text{PWM}} \in [1000, 2000]$ μs. This PWM signal is the physical output of the PX4 and is the input used by the ESCs to control the voltage



(a) Practical model (b) Idealized model

Figure 3.5: A summary of the force-torque control system

to the motor. The above process is summarized in Figure 3.5.

3.1.3 Software

In understanding how the entire software system functions it is important to examine how data flows to and from the quadrotor as well as in the quadrotor. An overview of the data flow on the ANCL quadrotor is in Figure 3.6. The backbone of the system is the micro-object request broker (uORB) module (src/modules/uORB). It is the object broker for the inter-process communication and is based on a publish-subscribe design pattern of the same name. Any number of sensors can be attached to the PX4. The most important internal sensors are described briefly below. *MPU6000* (src/drivers/mpu6000) is a six-axis gyroscope and accelerometer. It provides the raw measurements of angular velocity and acceleration to the system. *L3GD20* (src/drivers/l3gd20) is a three dimensional gyroscope. It can also provide raw angular velocity measurements to the system. *HMC5883l* (src/drivers/hmc5883) is a three dimension magnetometer. It provides the raw measurement of magnetic

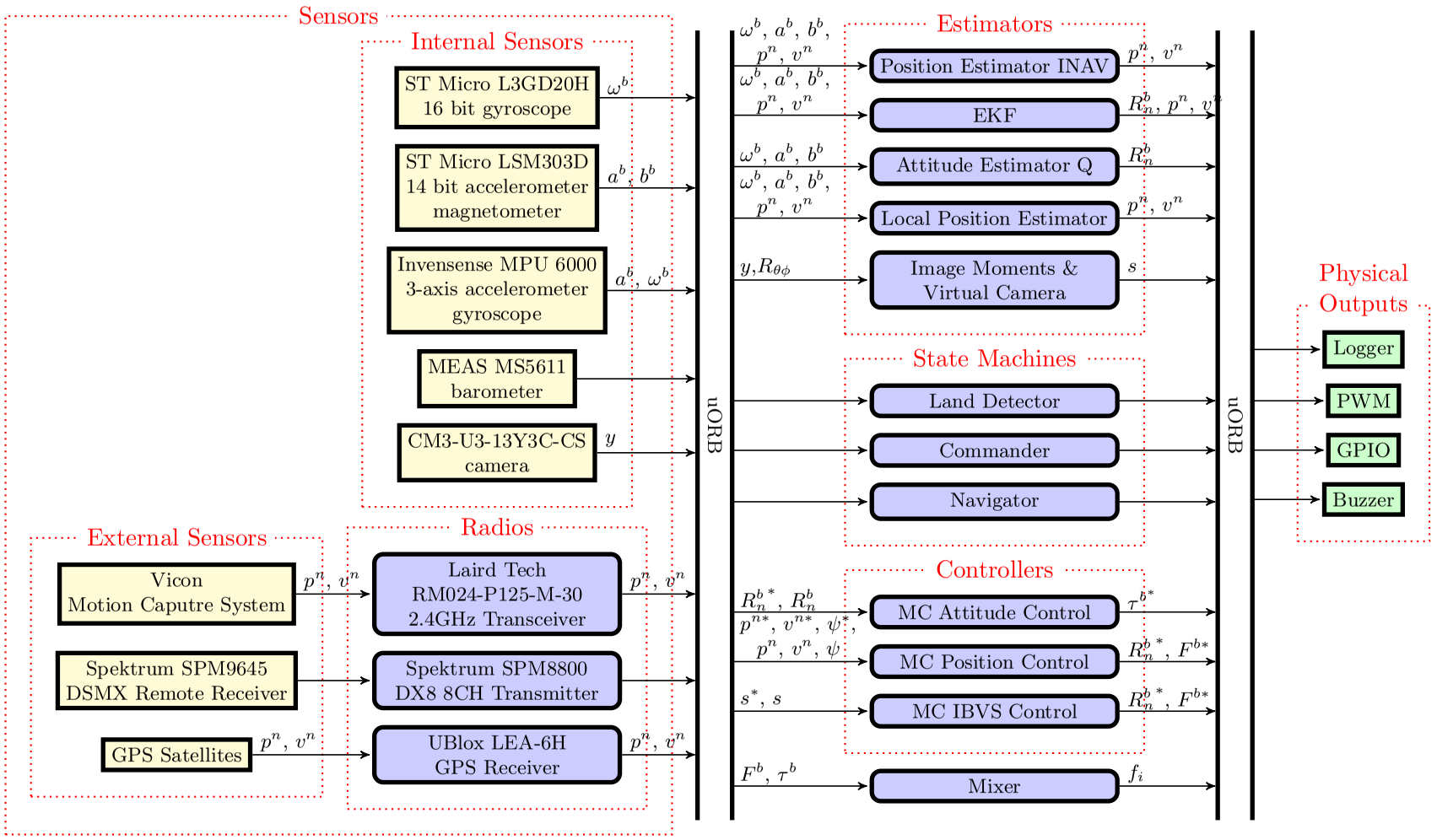


Figure 3.6: Data flow diagram of the ANCL quadrotor. uORB is the object broker for the inter-process communication based on the publish-subscribe design pattern.

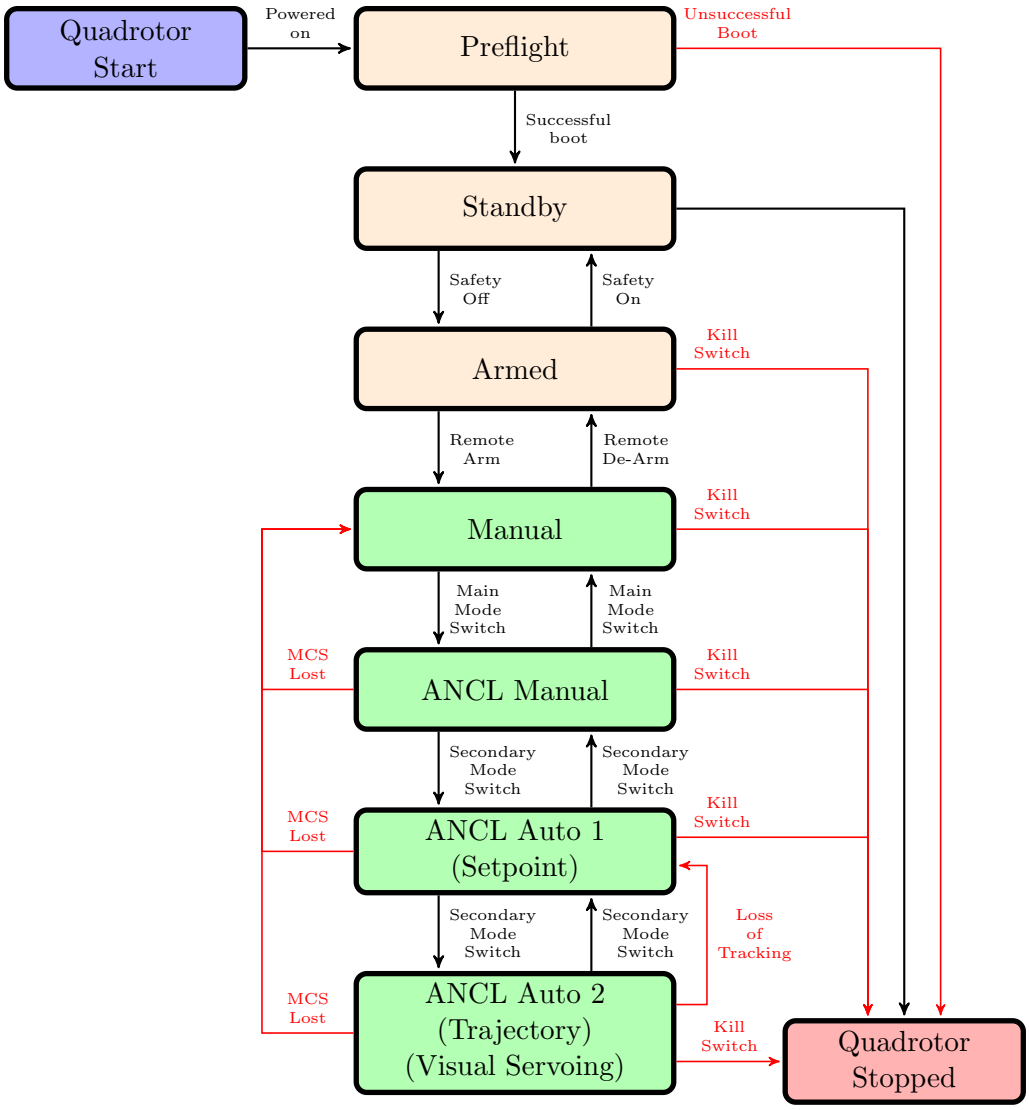


Figure 3.7: The ANCL internal software state machine for the quadrotor autopilot.

north. However, the accuracy indoors can be degraded due to iron inside the building. *MS5611* (src/drivers/ms5611) is a barometric pressure sensor. It provides the raw pressure data to the system. The platform also has three important external sensors. The *Vicon* (src/modules/mavlink) MCS provides the position and velocity of the vehicle. The Vicon cameras are connected to an offboard computer that calculates the position of the vehicle using reflective markers. The computer then calculates a low pass filtered finite difference estimate of the vehicle's linear velocity. Then the Vicon data is sent via a radio to the PX4. If the quadrotor is flying outdoors, it is also equipped a *GPS* (src/drivers/gps) module for position feedback. The last external sensor is the Spektrum *Remote Control* Transmitter (src/drivers/spektrum_rc) used for manual control of the quadrotor and for switching between control modes. There are various options for estimating its full state on the PX4. The default estimator on the PX4 is it's *EKF* module (src/modules/ekf2). This module calculates the vehicle's attitude, position and velocity. At the time we last merged the source code from the official PX4 repository it was not possible to use the *EKF2* module with a MCS[1]. Instead we use the *Quaternion Attitude Estimator* (src/modules/attitude_estimator_q) and the *Local Position Estimator* (src/modules/local_position_estimator). The attitude estimator uses quaternions to keep track of the attitude and mixes the roll and pitch estimates from the accelerometer, the heading from the the MCS, and the gyroscope measurements. The *Local Position Estimator* uses the position and velocity estimates from the MCS. It also uses the accelerometer data to extrapolate the position and velocity estimates in between data from the MCS. Onboard the PX4 the *Commander* module (src/modules/commander) maintains the main state machine. The state machine can be seen in Figure 3.7. It has been simplified for clarity. A typical flight would involve the following steps:

- The PX4 is powered on.
- The system starts all of its sensor's and does a preflight check.
- The user turns the safety switch off. When the safety is engaged the motors will not spin.
- Using the remote control, the user engages the motors. At this stage the user can fly in Manual Mode. This means that on the remote the left stick is mapped to thrust and yaw torque. The right stick is mapped to roll and pitch torque. There is no position feedback in this mode and typically the position of the quadrotor will slowly drift even with a zero roll and pitch torque.

[1]Release v1.5.5 - Jan. 25, 2017 - `https://github.com/PX4/Firmware/commit/00334ad76d03b5abc2144f1ebba7faf691448f5c`

Mode Switch	Secondary Mode Switch	Flight Mode	Thrust Stick Controls:	Roll-Pitch-Yaw Sticks Control:	Outer Loop Feedback
1	1	Manual	Thrust	Torque	Open loop
3	1	ANCL Manual	Vertical Body Velocity	Body Velocity	Position and velocity
3	2	ANCL Automatic Mode 1	Autonomous (MCS)	Autonomous (MCS)	Position and velocity
3	3	ANCL Automatic Mode 2	Autonomous (VS)	Autonomous (VS)	Image features

Table 3.2: ANCL radio switches and flight modes.

- Next the user uses the remote to switch into the ANCL Manual Mode. Now the left stick is mapped to vertical velocity and yaw speed. The right stick is mapped to horizontal velocity.
- Next the user uses the remote to switch into the ANCL Automatic Mode 1. This mode is completely autonomous. In this mode the quadrotor will fly to a setpoint and hold its position. It is typically set to the centre of the room and one metre off the ground $p^{n*} = [0, 0, -1]^T$ m.
- Next the user uses the remote to switch into the ANCL Automatic Mode 2. This mode is also completely autonomous. In this mode the quadrotor will do one of two missions. In the first it will follow a trajectory such as a circle around the lab. The second mission is typically a visual servoing (VS) control law.
- Once the mission is done the user uses the remote to switch back into the ANCL Manual Mode to land the quadrotor.
- Once the quadrotor has landed the user uses the remote to turn off the motors.
- The user now re-engages the safety.
- The PX4 is powered off.

A summary of the radio transmitter control sticks is shown in Figure 3.8 and a summary of the flight modes is shown in Table 3.2.

One of the most important groups of modules are the controllers. As seen in Section 2.1 The main control structure is split into two loops. The outer loop is implemented in the *Multicopter Position Control* module (src/modules/mc_pos_control) and the inner loop in the *Multicopter Attitude Control* module

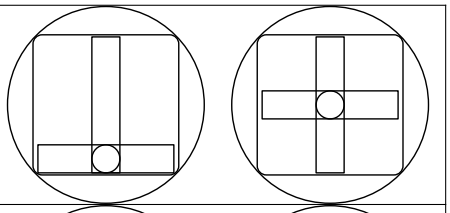
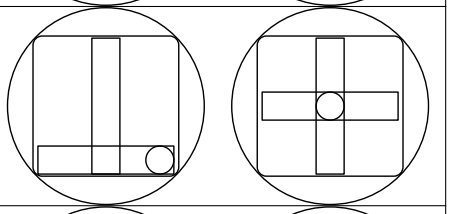
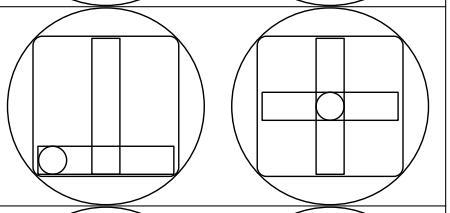
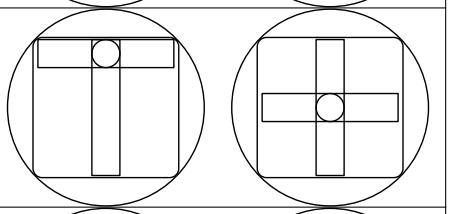
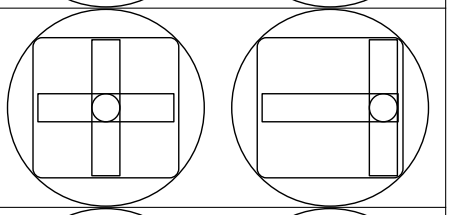
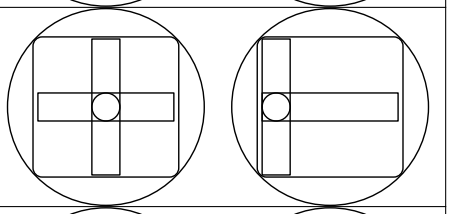
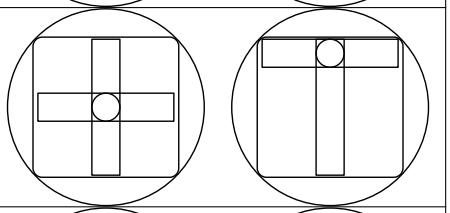
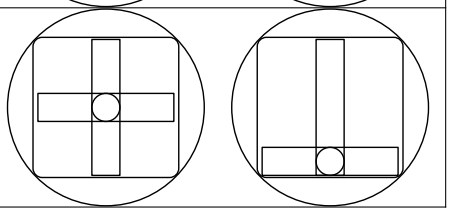
Zero Position	$\phi^* = 0$ $\theta^* = 0$ $\psi^* = 0$ $F^b = 0$	
Start Motors	$\phi^* = 0$ $\theta^* = 0$ $\psi^* = 1$ $F^b = 0$	
Stop Motors	$\phi^* = 0$ $\theta^* = 0$ $\psi^* = -1$ $F^b = 0$	
Full Throttle	$\phi^* = 0$ $\theta^* = 0$ $\psi^* = 0$ $F^b = 1$	
Roll Right	$\phi^* = 0$ $\theta^* = 1$ $\psi^* = 0$ $F^b = 0.5$	
Roll Left	$\phi^* = 0$ $\theta^* = -1$ $\psi^* = 0$ $F^b = 0.5$	
Pitch Forward	$\phi^* = -1$ $\theta^* = 0$ $\psi^* = 0$ $F^b = 0.5$	
Pitch Backwards	$\phi^* = 1$ $\theta^* = 0$ $\psi^* = 0$ $F^b = 0.5$	

Figure 3.8: The ANCL radio transmitter control stick mappings.

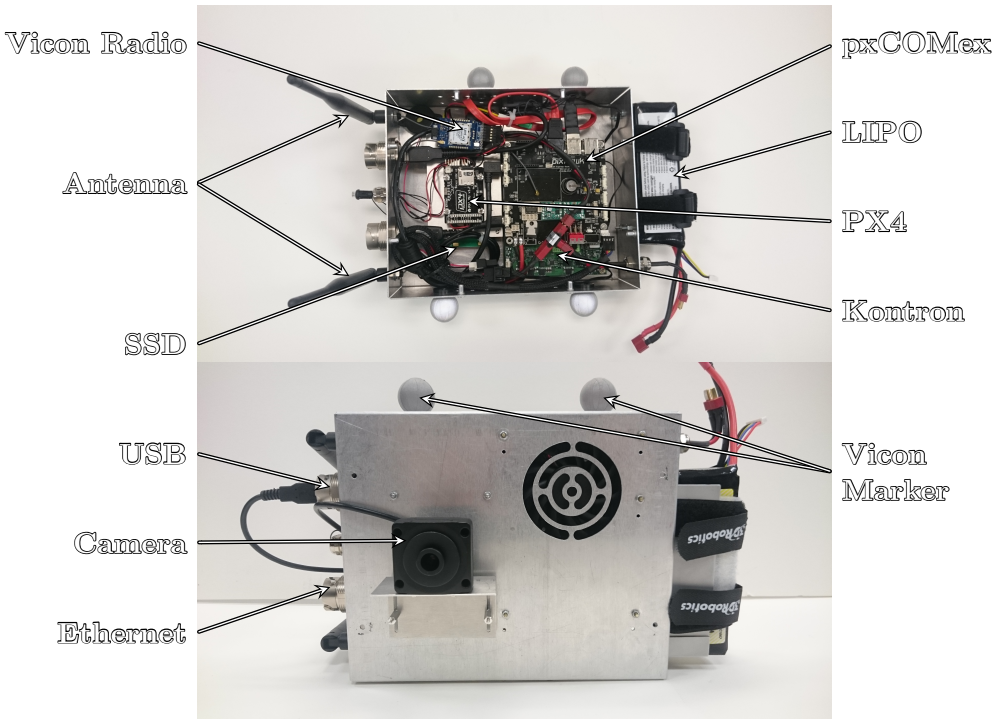


Figure 3.9: The embedded computer vision system.

(src/modules/mc_att_control). The control design for these modules is based off [101]. If we are running a VS mission the control for the image-based visual servoing (IBVS) control law is in the *MC IBVS Control* module (src/modules/mc_ibvs). The state machine and the position control module handles the switching between the different outer loop controllers.

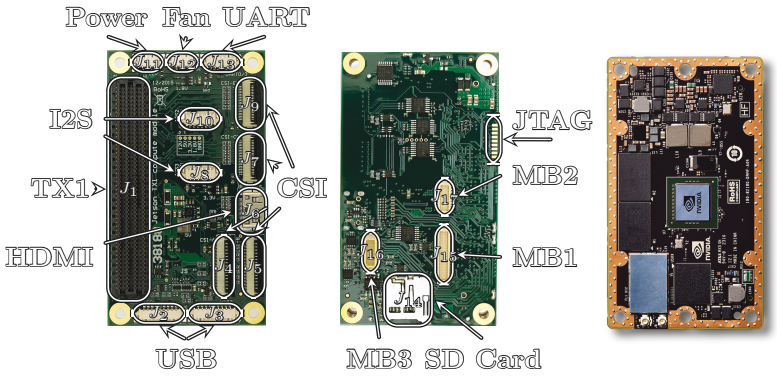
The last important group of modules are the physical output. The ESCs and in turn the motors are controlled via a PWM signal (src/drivers/px4io). Also, onboard the PX4 all of the uORB data is logged on an internal micro secure digital (SD) card (src/modules/logger).

3.2 Embedded Computer Vision System

The vision computer is a separate system from the PX4, but it is still located physically on the quadrotor. The computer vision system has gone through a number of iterations and its hardware consists of a camera and a computer system.

3.2.1 Companion Computers

The first iterations of the computer was system used a Raspberry Pi which is a low cost ARM-based computer running Linux. The second iteration used a CMUcam5 Pixy [102]. It can find hundreds of objects based on their colour up to 50 fps. It



(a) J100 Top (b) J100 Bottom (c) Jetson TX1

Figure 3.10: The Nvidia Jetson TX1 and the Auvidea J100 carrier board.

transmits image point features from a 320×200 image at a rate of about 25 Hz. Our current computer vision system is an Jetson TX1. The Jetson TX1 has a 64 bit ARM central processing unit (CPU) and a graphics processing unit (GPU) with 256 Nvidia CUDA cores. The Jetson TX1 is mounted on an Auvidea carrier board that has a serial connection to the PX4 for transferring vision data to the autopilot. The carrier board and TX1 is shown in Figure 3.10. We are in the process of phasing out the TX1 for the TX2 which provides better performance. It has a better CPU, GPU, more RAM and more storage. It is pin compatible with the TX1 which allows it to run on the same carrier board. All of the companion computer were mounted on the quadrotor and connected to the PX4 via a serial connection.

To enable faster prototyping of computer vision algorithms we developed a high performance computer vision system shown in Figure 3.9. This box includes a PX4, a camera, and a Pixhawk baseboard with an embedded computer. The embedded computer is a Kontron COM Express Module with an Intel i7-2655LE processor module, 8 GB RAM, and an Intel GMA X4500 integrated graphics card. The computer runs Ubuntu Linux and the robot operating system (ROS) [103]. The box has an solid state drive (SSD) to store camera images for post-processing. The attached camera is a Chameleon 2. Lastly, Vicon markers are mounted on the outside of the box to measure its pose. The entire vehicle system weight including battery and computer vision system is 1.6 kg.

For sensor fusion problems like visual inertial simultaneous localization and mapping (VISLAM) a source of error can be frame misalignment between $\mathcal{C}$, $\mathcal{B}$ and the inertial measurement unit (IMU). In theory the IMU and the MCS sensor are placed at the centre of mass (CoM) of the vehicle which is the origin of $\mathcal{B}$. However, in practice the exact CoM is unknown and the sensors cannot be physically placed there. Additionally, to be able to compare the estimated pose of the vehicle to the ground truth measured by our MCS, the transformation from the $\mathcal{N}$ to $\mathcal{V}$ has to be

accurately measured. We used the methods from [104] to estimate the inter-sensor calibration.

3.2.2 Cameras

Our main camera is the FLIR (previously Point Grey) Chameleon 3 USB 3 camera. It is attached to the quadrotor facing down about 5 cm below the quadrotor's CoM, i.e., $p^b_{\text{cam}} \approx [0, 0, 0.05]^T$ m. This global shutter camera can achieve framerates of up to 149 Hz when taking a photo at a resolution of 1280 $\times$ 1024. Our secondary camera is the Chameleon 2 universal synchronous bus (USB) camera. It is also a global shutter camera that can achieve framerates of up to 18 Hz when taking a photo at a resolution of 1296 $\times$ 964. Both the Raspberry Pi and the Jetson TX1 can use camera serial interface (CSI) cameras such as the Raspberry Pi Camera V2 (Sony IMX219) which can achieve up to 240 Hz when taking a photo at a resolution of 960 $\times$ 540. The main downside of this camera is that it has a rolling shutter.

3.3 Communication Network Characterization

Being able to describe the behaviour and characteristics of our communication network is crucial to developing algorithms to accurately obtain the pose of the vehicle after it has been sent from the ground station to the vehicle. The stability of network control system is now a well established branch of control engineering [105]. In the ANCL platform we use a Vicon packet that is 24 bytes. We use the Mavlink protocol to transmit the data. It adds an additional eight bytes of overhead that adds checksum and sequencing information to the packet. The checksum uses ITU X.25 and SAE AS-4 standards. See Figure 3.11 and Table 3.3 for an overview of the packet design. We designed the following experiments to identify the delay, the sources of the delay and the behaviour of the delay in our system. Apart from the delay we were also trying to measure any other problems that degrade the quality of the network such as packet loss and out of order packets.

3.3.1 Round Trip Experiment

The first experiment that we designed was a round trip experiment. In this experiment we would send a packet from our ground station PC to the PX4 using the LairdTech radios. In order to minimize any processing time on both the ground station and the PX4 all other applications were closed. The experiment was then as follows: at time t_1 on the ground station computer a packet was sent via the LairdTech radio to the same radio on the PX4. It was then sent back to the ground station where it was read in and verified at time t_2. For this experiment it was

Variable	Precision	Value	Description
T	`uint32`	t	Time in ms on the MCS clock
POS	`int16`	p^n	Position in $\mathcal{N}$ in mm
VEL	`int16`	v^n	Estimated Velocity in $\mathcal{N}$ in mm/s
ATT	`int16`	q	Attitude unit quaternion of R_n^b

(a) Vicon Packet (Payload)

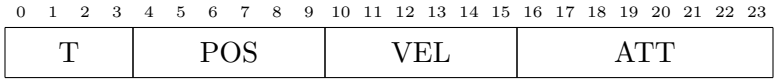
Variable	Precision	Value	Description
STX	`uint8`	0xFF	Packet start sign
LEN	`uint8`	$n = 24$	Length of payload
SEQ	`uint8`	-	Packet sequence to detect packet loss
SYS	`uint8`	42	System ID of the sending system to differentiate between different UAVs
COMP	`uint8`	20	Component ID
MSG	`uint8`	217	Message ID
PAYLOAD	-	-	Data of Message. Here it is the Vicon Packet
CK	`uint16`	-	Checksum of bytes $1 \cdots (n + 6)$

(b) Mavlink Version 1

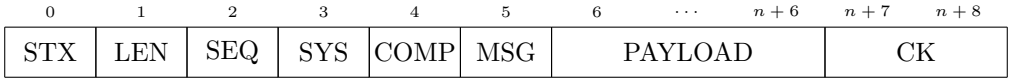
Variable	Precision	Value	Description
STX	`uint8`	0xFD	Packet start sign
LEN	`uint8`	$n = 24$	Length of payload
ICMPT	`uint8`	-	Incompatible flags (Flags must be understood)
CMPT	`uint8`	-	Compatible flags (Can be ignored if not understood)
SEQ	`uint8`	-	Packet sequence to detect packet loss
SYS	`uint8`	180	System ID of the sending system to differentiate between different UAVs
COMP	`uint8`	20	Component ID
MSG	`uint8[3]`	217	Message ID
TSYS	`uint8`	-	Target System ID (Optional)
TCMPID	`uint8`	-	Target Component ID (Optional)
PAYLOAD	-	-	Data of Message. Here it is the Vicon Packet
CK	`uint16`	-	Checksum of bytes $1 \cdots (n + 6)$
SIG	`unit8[13]`	-	Signature (Optional)

(c) Mavlink Version 2

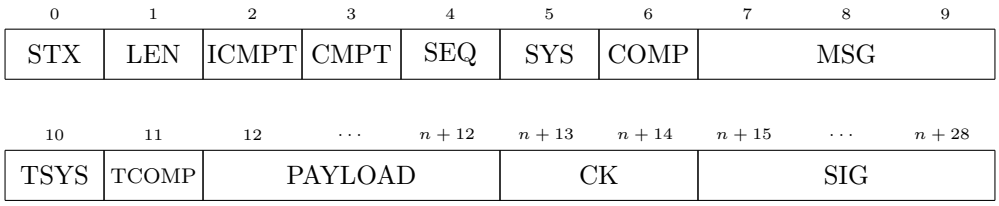
Table 3.3: Vicon packet overview.



(a) Vicon Packet (Payload)



(b) The structure of a Mavlink version 1 packet.



(c) The structure of a Mavlink version 2 packet. The signature is optional.

Figure 3.11: The structure of the Mavlink Vicon packet.

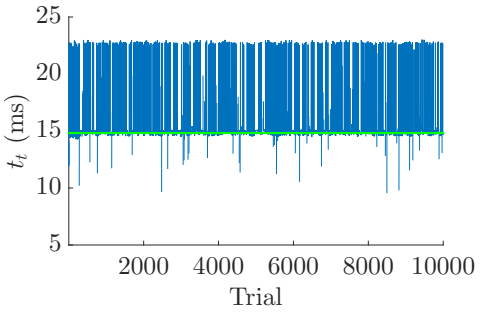


Figure 3.12: The one-way transmission time t_t in the round trip experiment where a Vicon packet was sent to the PX4 and then immediately sent back.

assumed that any processing time or buffering by the drivers and radios is a constant and included as part of the transmitting time. Secondly, it was assumed that the time for the PX4 to read, verify, and write the packet back to the radio was negligible. The last assumption is that is that the probability to lose a packet is random and is equal in both directions. Therefore the time to transmit t_t a packet is

$$t_t = \frac{t_2 - t_1}{2}$$

After 10,000 runs we obtained an average $t_t = 14.8$ ms as shown in Figure 3.12 During this experiment 1.4% of the sent packets were lost, 94.4% were sent within ± 1 ms of the average t_t, and 3.9% of the packets were received later than the average.

3.3.2 Throughput Experiment

In the previous experiment only one packet was sent at a time which measures a minimum time. In a more realistic scenario there is a constant stream of data. In

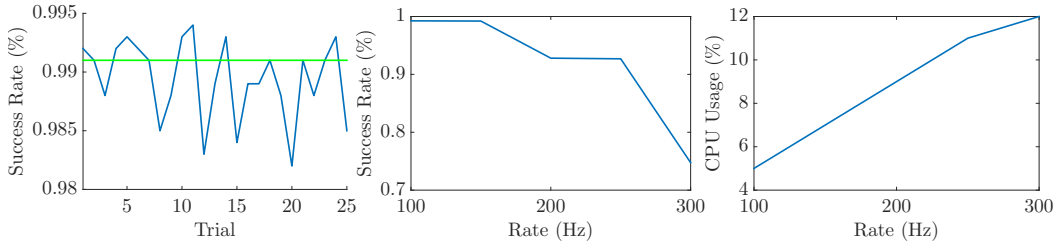


Figure 3.13: The throughput experiment where a constant stream of Vicon packets were sent to the PX4.

this experiment we would send N packets continuously from the ground station to the quadrotor, and the number of packets received at the PX4 is denoted by M. We define the percentage of packet loss as

$$\text{loss} = \frac{N - M}{N} .$$

After 25 trials sending 1000 packets per trial at a rate of 100 Hz we obtained an average success rate of greater than 99% as shown in Figure 3.13a. Next, we repeated the same experiment but sending 10,000 packets per trial and we varied the rate at which we sent the data from 100 Hz to 300 Hz. We measured both the success rate and the CPU usage as seen in Figure 3.13b and Figure 3.13c, respectively.

3.3.3 Complete Circuit Experiment

The final experiment performed was based on the complete time it takes for the MCS to calculate the pose of the vehicle and send the data to the PX4. The Bonita Cameras emit 617 nm light which is reflected by the markers on the quadrotor back to the cameras. When a marker is seen by any of the cameras, its position in the image is collected by the Vicon Tracker software where it is used to help triangulate the 3 D position of the quadrotor. To simulate the reflection of a marker we used an array of light-emitting diodes (LEDs) emitting nearly the same wavelength. We designed a simple circuit as shown in the Figure 3.14 to create a object with eight LEDs that represented two of the quadrotor's reflective markers. To use an external power supply to power the eight LEDs the circuit contains an optocoupler 4N25. The LEDs are placed in clusters to increase the surface area of the LEDs. The pairs of LEDs are placed in parallel. The input to the optocoupler is the output of one of the PX4's general purpose input/outputs (GPIOs). We could toggle on and off the array of LEDs using the PX4. On the PX4 whenever the GPIO pin is toggled high the time t_1 is recorded and when it is toggled low t_2 is recorded. At the same time when the pin is set high the LEDs are turned on, the Vicon MCS sees the LEDs and

calculates the position of the LEDs, the ground station then sends the position to PX4 where the time is recorded every time a packet is received. The LEDs are left on for 10 s and then turned off for 3 s. In this experiment we are measuring multiple times of interest: t_{on} is the time it takes for the first Vicon packet to arrive after the LEDs were turned on, t_{off} is the time the last packet was received after the LEDs were turned off, and t_{avg} is the average time between packets excluding the first packet. We also counted the total number of packets received denoted by N. From this experiment we expect to measure that t_{avg} is the same amount of time that the MCS takes in between sending data (0.01 ms). N should be the product of the total time the LED was on, the rate at which the MCS sends data and the average success rate calculated in the first experiment, i.e., $N \approx (10\text{ s})(100\text{ Hz})(99\ \%) = 990$. The abovementioned times are defined as

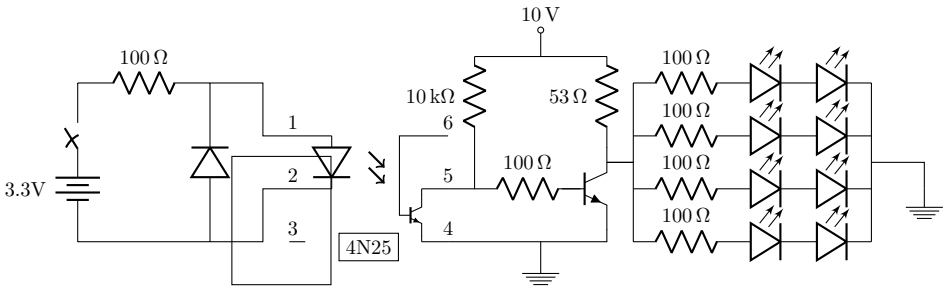
$$
\begin{aligned}
t_{\text{on}} &= t_1 - t_{\text{first}} \\
t_{\text{off}} &= t_2 - t_{\text{last}} \\
t_{\text{avg}} &= \frac{1}{N-1} \sum_{i=2}^{N} \Delta t^i
\end{aligned}
$$

where $\Delta t^i = t^i_{\text{current}} - t^i_{previous}$. In this experiment we neglected the delay of the optocoupler as the delay is in nano-seconds. The experiment was repeated 100 times.

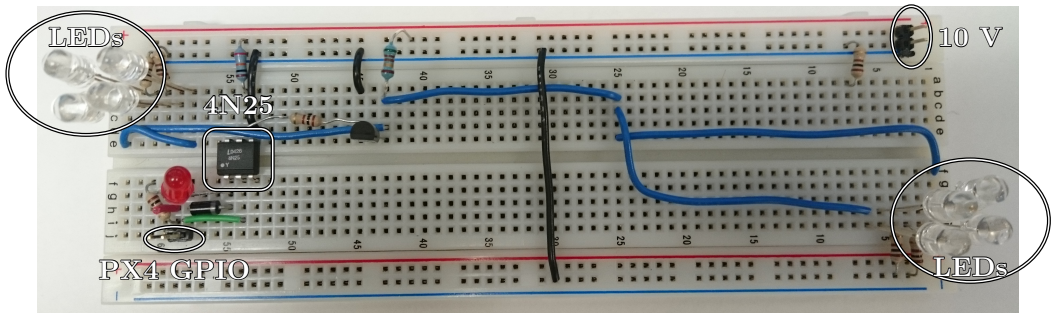
From this experiment was can see that $t_{\text{on}} > t_{\text{off}}$ and $t_{\text{on}} > t_{\text{avg}}$ as the MCS needs additional time to find an object compared to the time it takes to track an object already being tracked. In practice the quadrotor should never be lost from the MCS so the first data point is removed from the average time. Lastly we can observe that the delay is not constant. We obtained a mean of 26 ms, but the value varied from 20 ms to 33 ms. These values are in line with our first experiment where the transmission time was about 15 ms. A summary of these results is in Figure 3.15. The computation time of the Vicon MCS can be seen to be almost exactly 10 ms while tracking the LEDs, however, it needs an additional 25 ms to find the object when the LEDs are first turned on.

3.4 Summary

In this chapter we presented the ANCL quadrotor platform which made use of open source and open hardware projects. Like most autopilot platforms it is in a constant stage of change as newer technology emerges. The platform we developed will be an important tool for keeping abreast with research on computer vision applied to UAVs. The platform is used in the next two chapters for VS and visual state estimation (VSE).

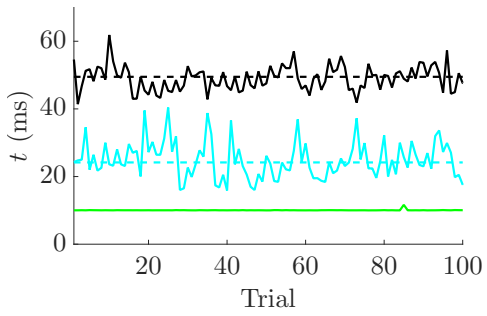


(a) Circuit Diagram

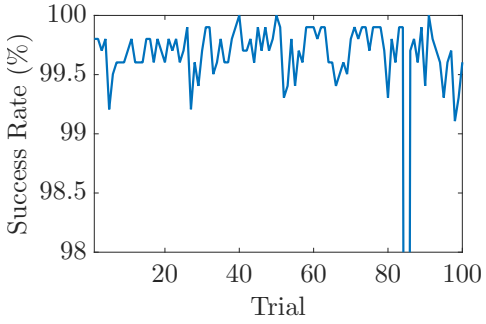


(b) Circuit

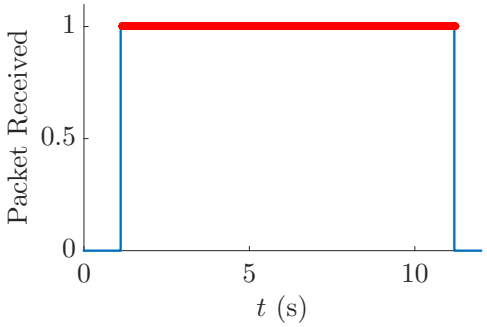
Figure 3.14: The LED array used in the complete circuit experiment.



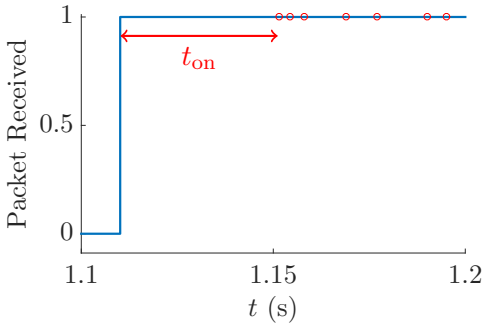
(a) The time is takes to receive a packet after the LEDs were turned on t_{on} (black), the time it takes to receive the last packet after the LEDs were turned off t_{off} (blue) and the average time in between packets t_{avg} (green).



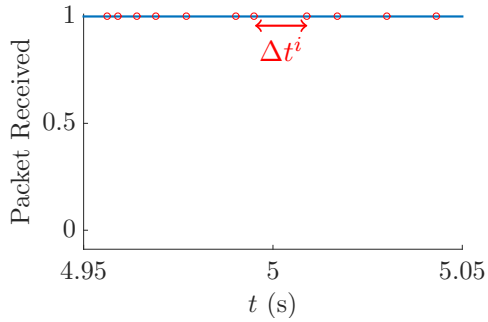
(b) The success rate of each trial. We can see one trial drops well below the average when the radios temporarily lost connection during a trial.



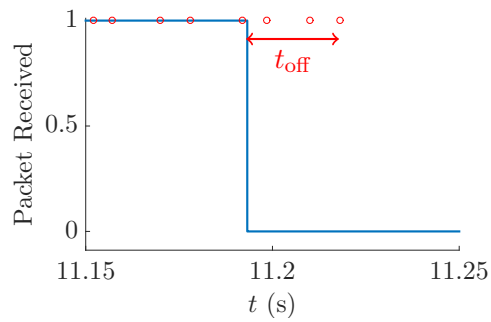
(c) Close up of the first trial in the experiment. The blue line denotes when the LEDs are turned on and every received packet is marked in red.



(d) The first trial's t_{on} measurement. The blue line denotes when the LEDs are turned on and every received packet is marked in red.



(e) An example t^i from the first trial that is used to compute t_{avg}. The blue line denotes when the LEDs are turned on and every received packet is marked in red.



(f) Example t_{off} measurements from the first trial. The blue line denotes when the LEDs are turned on and every received packet is marked in red.

Figure 3.15: The complete circuit experiment that measures the time it takes for the PX4 to receive a packet from the Vicon MCS.

Chapter 4

Dynamic Image-Based Visual Servoing

A conventional visual servoing design usually assumes the inner loop tracks its reference velocity perfectly. Analysis and design is based on a purely kinematic vehicle model. However, it is pointed out in [89] that the dynamics of a robot should be considered for high speed tasks or when the system is underactuated. We refer to a visual servo control which directly accounts for vehicle dynamics as *dynamic* visual servoing (VS). The importance of treating the dynamics of a vehicle is also underlined in [47]. Although dynamic VS has clear practical significance, there is relatively sparse literature on the topic for any type of robotic application. This is likely due in part to the difficulty in rigorously accounting for the nonlinearity of the camera's perspective projection and the vehicle's dynamics.

In Section 4.1 we present our state transformation method that was originally published in [54]. The section presents a method for nonlinear dynamic image-based visual servoing (DIBVS) using just a single feature point. Next, in Section 4.2 we present the virtual camera method. We begin by deriving image moments features and their kinematics. Next, we derive a control law using the virtual camera method and then experimentally validate it. This method was originally published in [55]. In Section 4.3 we extend our virtual camera method to be adaptive to various system parameters. This extension was originally published in [56]. Then we propose two more extensions to the virtual camera method in Section 4.4. First we extend the method for moving targets. Second, we use a homography matrix methods for non-horizontal targets. These extensions were originally publish in [57]. Lastly, in Section 4.5 we summarize our findings on DIBVS.

4.1 State Transformation-Based Approach

In this section we present our state transformation-based DIBVS approach that was originally published in [54]. This section focuses on outer loop control where the objective is to regulate lateral position above a visual target on the ground.

4.1.1 Fundamentals

We recall the image feature kinematics (2.34) and translational velocity dynamics (2.6):

$$\begin{aligned} \dot{y} &= L_v v^b + L_\omega \omega^b \\ m\dot{v}^b &= -m\omega^b \times v^b + F^b + mR_n^b g^n \end{aligned} \tag{4.1}$$

The state transformation method assumes perfect control of v_3^b and ω_3^b so that $v_3^b = \omega_3^b = 0$. As well, it only considers small η which implies $\omega \approx \dot{\eta}$. It assumes that the body frame $\mathcal{B}$ and the camera frame $\mathcal{C}$ are the same frame. Lastly, it assumes that $\lambda = \lambda_1 = \lambda_2$. Under these assumptions (4.1) becomes a two degree of freedom (DoF) system

$$\Sigma_{2D} \begin{cases} \dot{y}_1 = -\dfrac{\lambda v_1^b}{p_3^b} - \dfrac{\lambda^2 + (y_1 - y_{10})^2}{\lambda}\dot{\theta} + \dfrac{(y_1 - y_{10})(y_2 - y_{20})}{\lambda}\dot{\phi} \\ \dot{v}_1^b = -g\theta \\ \dot{y}_2 = -\dfrac{\lambda v_2}{p_3^b} + \dfrac{\lambda^2 + (y_2 - y_{20})^2}{\lambda}\dot{\phi} - \dfrac{(y_1 - y_{10})(y_2 - y_{20})}{\lambda}\dot{\theta} \\ \dot{v}_2^b = g\phi \end{cases} \tag{4.2}$$

where the inputs are θ and ϕ. This method considers the two decoupled nominal one DoF systems as a basis for the outer loop control design. This model choice is justified by both a robustness analysis given below and comes from the two loop control structure chosen.

The nominal one DoF dynamics considered are

$$\Sigma_{1D,\theta} \begin{cases} \dot{y}_1 = -\dfrac{\lambda v_1^b}{p_3^b} - \dfrac{\lambda^2 + (y_1 - y_{10})^2}{\lambda}\dot{\theta} \\ \dot{v}_1^b = -g\theta \end{cases} \tag{4.3}$$

$$\Sigma_{1D,\phi} \begin{cases} \dot{y}_2 = -\dfrac{\lambda v_2^b}{p_3^b} + \dfrac{\lambda^2 + (y_2 - y_{20})^2}{\lambda}\dot{\phi} \\ \dot{v}_2^b = g\phi \end{cases} \tag{4.4}$$

where the input to subsystem $\Sigma_{1D,\theta}$ is θ and input to $\Sigma_{1D,\phi}$ is ϕ.

A challenge with controlling the nonlinear dynamics in (4.2) to (4.4) is the ap-

pearance of the derivative of the input, i.e., $\dot{\phi}$ and $\dot{\theta}$. To account for this we make use of work in [106] which considers the problem of reducing the order of the input derivative in state space representations. This approach makes use of state transformations which depend on state and input and its derivatives. To present the conditions for the existence of the state transformation, the following definitions from [107, 108] are needed. A smooth vector field $f : \mathbb{R}^n \to \mathbb{R}^n$ is denoted $f(x) = \sum_{i=1}^{n} f_i(x)\frac{\partial}{\partial x_i}$ where $\frac{\partial}{\partial x_i}$ denotes the ith unit vector field in the x-coordinates. Given two smooth vector fields f and g the Lie bracket is denoted $[f, g](x) = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g$. Given the state space form

$$\dot{x} = f(x, u, \dot{u}), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

the goal is to attempt to construct a state transformation

$$z = z(x, u)$$

which leads to a classical state space form system

$$\dot{z} = \tilde{f}(z, u)$$

where all the input derivatives in $\tilde{f}$ have been removed.

From [106, Thm.1], the necessary and sufficient conditions for the existence of a state transformation from $\Sigma_{1D,\theta}$ into a classical state space form $\tilde{\Sigma}_{1D,\theta}$ are given by

$$\left[\frac{\partial}{\partial\theta}, \left[\frac{\partial}{\partial\theta}, \mathcal{F}\right]\right] = 0 \tag{4.5}$$

where the prolonged vector field $\mathcal{F}$ is defined as

$$\mathcal{F} = \dot{y}_1 \frac{\partial}{\partial y_1} + \dot{v}_1^b \frac{\partial}{\partial v_1^b} + \dot{\theta}\frac{\partial}{\partial\theta}$$

Similarly, the transformation from $\Sigma_{1D,\phi}$ to $\tilde{\Sigma}_{1D,\phi}$ exists if and only if

$$\left[\frac{\partial}{\partial\phi}, \left[\frac{\partial}{\partial\phi}, \mathcal{F}\right]\right] = 0 \tag{4.6}$$

where

$$\mathcal{F} = \dot{y}_2 \frac{\partial}{\partial y_2} + \dot{v}_2^b \frac{\partial}{\partial v_2^b} + \dot{\phi}\frac{\partial}{\partial\phi}$$

Straightforward calculations show both (4.5) and (4.6) hold and therefore the transformations exists. A simple choice for the transformations from $\Sigma_{1D,\theta}$ to $\tilde{\Sigma}_{1D,\theta}$

is

$$\begin{aligned} z_1 &= \arctan\left(\frac{y_1 - y_{10}}{\lambda}\right) + \theta \\ z_2 &= v_1^b \end{aligned} \tag{4.7}$$

where $u_1 = \theta$. The transformation from $\Sigma_{1D,\phi}$ to $\tilde{\Sigma}_{1D,\phi}$ is

$$\begin{aligned} z_3 &= \arctan\left(\frac{y_2 - y_{20}}{\lambda}\right) - \phi \\ z_4 &= v_2^b \end{aligned} \tag{4.8}$$

where $u_2 = \phi$. The transformed dynamics are

$$\tilde{\Sigma}_{1D,\theta} \begin{cases} \dot{z}_1 = -\dfrac{1}{p_3^b} z_2 \cos^2(z_1 - u_1) \\ \dot{z}_2 = -g u_1 \end{cases} \tag{4.9}$$

$$\tilde{\Sigma}_{1D,\phi} \begin{cases} \dot{z}_3 = -\dfrac{1}{p_3^b} z_4 \cos^2(z_3 - u_2) \\ \dot{z}_4 = g u_2 \end{cases} \tag{4.10}$$

There are three cases to consider for the two dimensional dynamics Σ_{2D}. First, the conditions to remove $\dot{\theta}$ alone are

$$\left[\tfrac{\partial}{\partial\theta}, \left[\tfrac{\partial}{\partial\theta}, \mathcal{F}\right]\right] = \left[\tfrac{\partial}{\partial\theta}, \left[\tfrac{\partial}{\partial\phi}, \mathcal{F}\right]\right] = 0 \tag{4.11}$$

where

$$\mathcal{F} = \dot{y}_1 \frac{\partial}{\partial y_1} + \dot{y}_2 \frac{\partial}{\partial y_2} + \dot{v}_1^b \frac{\partial}{\partial v_1^b} + \dot{v}_2 \frac{\partial}{\partial v_2^b} + \dot{\theta} \frac{\partial}{\partial \theta} + \dot{\phi} \frac{\partial}{\partial \phi}$$

Second, the conditions to remove $\dot{\phi}$ alone are

$$\left[\tfrac{\partial}{\partial\phi}, \left[\tfrac{\partial}{\partial\phi}, \mathcal{F}\right]\right] = \left[\tfrac{\partial}{\partial\phi}, \left[\tfrac{\partial}{\partial\theta}, \mathcal{F}\right]\right] = 0 \tag{4.12}$$

Finally, the conditions to remove both $\dot{\theta}$ and $\dot{\phi}$ are (4.11) and (4.12) and

$$\left[\left[\tfrac{\partial}{\partial\theta}, \mathcal{F}\right], \left[\tfrac{\partial}{\partial\phi}, \mathcal{F}\right]\right] = 0 \tag{4.13}$$

Although (4.11) and (4.12) hold,

$$\left[\left[\tfrac{\partial}{\partial\theta}, \mathcal{F}\right], \left[\tfrac{\partial}{\partial\phi}, \mathcal{F}\right]\right] = y_2 \frac{\partial}{\partial y_1} - y_1 \frac{\partial}{\partial y_2} \neq 0$$

and therefore both of the input derivatives can not be eliminated simultaneously

from Σ_{2D}. Thus, we choose to apply the transformations (4.7) and (4.8) which partially remove the input derivative dependence. The transformed system becomes

$$\tilde{\Sigma}_{2D}\begin{cases}\dot{z}_1 = -\dfrac{1}{p_3^b} z_2 \cos^2(z_1 - u_1) + \lambda \tan(z_1 - u_1)\tan(z_3 + u_2)\dot{u}_2 \\ \dot{z}_2 = -g u_1 \\ \dot{z}_3 = -\dfrac{1}{p_3^b} z_4 \cos^2(z_3 + u_2) - \lambda \tan(z_1 - u_1)\tan(z_3 + u_2)\dot{u}_1 \\ \dot{z}_4 = g u_2\end{cases} \tag{4.14}$$

The control design is performed in the z-coordinates and uses 1 DoF models $\tilde{\Sigma}_{1D,\theta}$ and $\tilde{\Sigma}_{1D,\phi}$. Theorem 4.1 below provides this control design and proves its exponential stability.

4.1.2 Control

We define a control law to regulate the relative lateral position of the vehicle to a stationary target located on the ground.

Theorem 4.1. *The origins of the one DoF closed-loop dynamics $\Sigma_{1D,\theta}$ and $\Sigma_{1D,\phi}$ defined in* (4.3) *and* (4.4) *with inputs*

$$\theta = -k_1 k_2 \frac{k_1 \arctan\left(\dfrac{y_1 - y_{10}}{\lambda}\right) - v_1^b}{1 + k_1^2 k_2} \quad \textit{and} \quad \phi = k_3 k_4 \frac{k_3 \arctan\left(\dfrac{y_2 - y_{20}}{\lambda}\right) - v_2^b}{1 + k_3^2 k_4} \tag{4.15}$$

are locally exponentially stable where $k_1, k_3 > 0$ and $k_2, k_4 > \dfrac{1}{g p_3^b}$.

Proof : To stabilize the dynamics $\tilde{\Sigma}_{1D,\theta}$ in (4.9) we consider $V_1 = \frac{1}{2} z_1^2$ then

$$\dot{V}_1 = z_1 \dot{z}_1 = -\frac{1}{p_3^b} z_1 z_2 \cos^2(z_1 - u_1)$$

If we treat z_2 as an input to the system and assign $z_2 = k_1 z_1$ where $k_1 > 0$ then

$$\dot{V}_1 = -\frac{k_1}{p_3^b} z_1^2 \cos^2(z_1 - u_1) \leq 0$$

Next, we define the following Lyapunov function candidate

$$V_2 = \frac{1}{2} z_1^2 + \frac{1}{2} q^2$$

where $q = z_1 - \frac{z_2}{k_1}$ then

$$\begin{aligned}\dot{V}_2 &= z_1\dot{z}_1 + q\dot{q} \\ &= -\frac{k_1}{p_3^b}z_1^2\cos^2(z_1 - u_1) + \frac{q}{k_1 p_3^b}(k_1^2 q\cos^2(z_1 - u_1) + g p_3^b u_1)\end{aligned}$$

Letting $u_1 = -k_1^2 k_2 q$ gives

$$\dot{V}_2 = -\frac{1}{p_3^b}\begin{bmatrix} z_1 & z_2 \end{bmatrix} M \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where

$$M_1 = \begin{bmatrix} g p_3^b k_1 k_2 & \cos^2(z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2) - g p_3^b k_2 \\ \cos^2(z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2) - g p_3^b k_2 & \frac{1}{k_1}\left(g p_3^b k_2 - \cos^2(z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2)\right) \end{bmatrix}$$

For $M_1 > 0$ we require

$$\det(M_1) = \left(g p_3^b k_2 - \cos^2(z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2)\right)\cos^2(z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2) > 0$$

and $g p_3^b k_1 k_2 > 0$. We can satisfy this constraint by restricting z to the region $\{(z_1, z_2) \in \mathbb{R}^2 : |z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2| < \frac{\pi}{2} - \varepsilon, 0 < \varepsilon < \frac{\pi}{2}\}$, $k_1 > 0$, and $k_2 > \frac{1}{g p_3^b}$. Then $\dot{V}_2 \leq -\frac{1}{p_3^b}\lambda_{\min}(M_1)\|z\|^2$ holds where

$$\lambda_{\min} = \left(a - \sqrt{a^2 - c}\right)/2$$

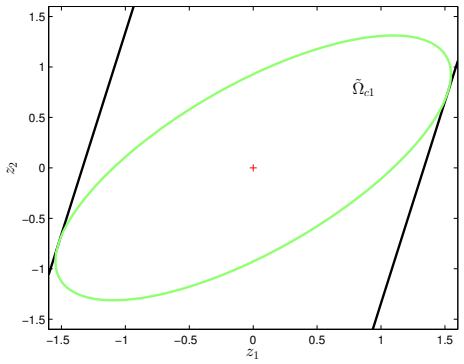
where $a = -\sin^2(\epsilon) + g k_2 p_3^b + g {k_1}^2 k_2 p_3^b$ and $c = 4\sin^2(\epsilon) k_1^2 \left(\cos^2\epsilon - 1 + g k_2 p_3^b\right)$. This ensures the equilibrium $z_1 = z_2 = 0$ is locally exponentially stable and since the change of coordinates is Lipschitz on its domain, stability is preserved for the original (y_1, v_1^b)-coordinates. An analogous proof applies to subsystem $\tilde{\Sigma}_{1D,\phi}$ with

$$V_4 = \frac{1}{2}z_3^2 + \frac{1}{2}\left(z_3 - \frac{z_4}{k_3}\right)^2$$

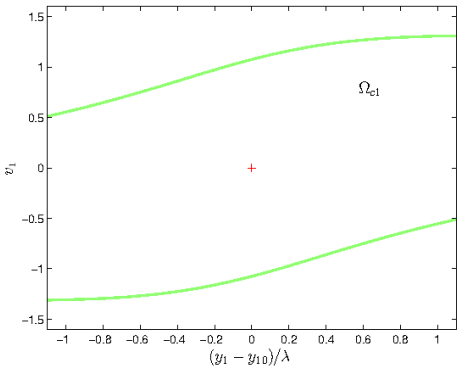
■

The choice of controller gains k_1, k_2 determine the shape of the regions of attraction. To compute an estimate of the region of attraction for (4.9) we find the intersection of the ellipse

$$V_2 = \frac{1}{2}z_1^2 + \frac{1}{2}\left(z_1 - \frac{z_2}{k_1}\right)^2 = c$$



(a) (z_1, z_2)-coordinates: $\tilde{\Omega}_{c1}$



(b) $\left(\left(\frac{y_1 - y_{10}}{\lambda}\right), v_1^c\right)$-coordinates: Ω_{c1}

Figure 4.1: Estimates of regions of attraction for controller gains $k_1 = 0.6, k_2 = 0.49$. These controller gains are used in Section 5.5.1.

with the lines

$$L_1 = \{(z_1, z_2) \in \mathbb{R}^2 : z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2 = \pi/2\}$$
$$L_2 = \{(z_1, z_2) \in \mathbb{R}^2 : z_1 + k_1^2 k_2 z_1 - k_1 k_2 z_2 = -\pi/2\}$$

The expression for c is

$$c = \frac{\pi^2}{8(k_1^4 k_2^2 + 1)}$$

An estimate of the region of attraction is $\tilde{\Omega}_{c1} = \{(z_1, z_2) \in \mathbb{R}^2 : V_2 < c\}$. This set can be expressed in the (y_1, v_1^b)-coordinates using the coordinate transformation; we denote this set Ω_{c1}. Examples of these estimated regions of attraction are given in Figure 4.1a and Figure 4.1b.

To show the stability of the two DoF dynamics Σ_{2D} with input (4.15) we make use of a result from [109, Lem.9.1] for perturbed systems. We define

$$z = [z_1, z_2, z_3, z_4]^T$$

and recall the dynamics Σ_{2D}

$$\begin{aligned}
\dot{z}_1 &= -\frac{1}{p_3^b} z_2 \cos^2(z_1 - u_1) + \underbrace{\lambda \tan(z_1 - u_1) \tan(z_3 + u_2) \dot{u}_2}_{\Delta_1} \\
\dot{z}_2 &= -g u_1 \\
\dot{z}_3 &= -\frac{1}{p_3^b} z_4 \cos^2(z_3 + u_2) - \underbrace{\lambda \tan(z_1 - u_1) \tan(z_3 + u_2) \dot{u}_1}_{\Delta_3} \\
\dot{z}_4 &= g u_2
\end{aligned} \tag{4.16}$$

where we have defined a disturbance vector

$$\Delta(z) = [\Delta_1(z), 0, \Delta_3(z), 0]^T$$

with $\Delta(0) = 0$. We remark that the dynamics (4.16) is a closed-loop system but we have not substituted the expression for inputs u_1, u_2 given in (4.15) in order to simplify presentation. We make use of the Lyapunov function candidates for the nominal systems which were given in Theorem 4.1 to define

$$\begin{aligned} V(z) &= V_2(z_1, z_2) + V_4(z_3, z_4) \\ &= \frac{1}{2}z_1^2 + \frac{1}{2}(z_1 - \frac{z_2}{k_1})^2 + \frac{1}{2}z_3^2 + \frac{1}{2}(z_3 - \frac{z_4}{k_1})^2 \end{aligned}$$

Due to the symmetry of roll and pitch dynamics we set $k_3 = k_1$ and $k_4 = k_2$. As well, from Theorem 4.1 we know

$$c_1\|z\|^2 \leq V \leq c_2\|z\|^2 \qquad \text{and} \qquad \dot{V} < c_3\|z\|^2$$

where c_1, c_2 and c_3 are positive constants. Furthermore, we have

$$\begin{aligned} \left\|\tfrac{\partial V}{\partial z}\right\|^2 &= \left(2z_1 - \frac{z_2}{k_1}\right)^2 + \left(\frac{z_1}{k_1} - \frac{z_2}{k_1^2}\right)^2 + \left(2z_3 - \frac{z_4}{k_1}\right)^2 + \left(\frac{z_3}{k_1} - \frac{z_4}{k_1^2}\right)^2 \\ &= z^T \begin{bmatrix} M_2 & 0 \\ 0 & M_2 \end{bmatrix} z \end{aligned}$$

where

$$M_2 = \begin{bmatrix} \dfrac{4k_1^2+1}{k_1^2} & -\dfrac{2k_1^2+1}{k_1^3} \\ -\dfrac{2k_1^2+1}{k_1^3} & \dfrac{k_1^2+1}{k_1^4} \end{bmatrix}$$

Hence,

$$\left\|\tfrac{\partial V}{\partial z}\right\| \leq \sqrt{\lambda_{\max}(M_2)}\|z\| \leq c_4\|z\|$$

where $c_4 > 0$. On a subset D of the origin we can calculate

$$\gamma = \big\|q(z)\big\|_\infty = \max_i \sum_{j=1}^{4} \left|\tfrac{\partial q_i}{\partial z_j}\right| = \sum_{j=1}^{4} \left|\tfrac{\partial q_1}{\partial z_j}\right|$$

which ensures $\big\|\Delta(z)\big\| \leq 2\gamma\|z\|$ on D. There exists controller gains k_1, k_2 such that $\gamma < c_3/c_4$ and this implies $z = 0$ is a locally exponentially stable equilibrium of the perturbed system (4.16). The region where the conditions for exponential stability hold can be determined numerically.

4.1.3 Simulation Results

We considered three different cases in simulation. The first simulation is the ideal case where we simulate the two DoF dynamics Σ_{2D} in (4.2) where we have a "perfect" inner loop control. That is, we ignore the rotational dynamics (2.6d) and can instantly control the roll and pitch angles. In the second case we include proportional-derivative (PD) controlled rotational dynamics and proportional-integral-derivative (PID) controlled height dynamics from (2.18).

The third simulation compares our controller against the DIBVS method in [91]. We simplify this approach to control only lateral motion which results in the image-based visual servoing (IBVS) law

$$F_3^b = -\frac{k_1^2 k_2}{m}\delta_2 \tag{4.17}$$

where

$$\delta_2 = \frac{mv}{k_1} - \delta_1 \text{ and } \delta_1 = s - R^T s^*$$

where $k_1, k_2 > 0$ are controller gains, and s is the image feature in spherical coordinates, and $s^* = [0, 0, 1]^T$ is the desired image feature. The spherical coordinates are calculated from the 2 D image coordinates $y = [y_1, y_2, f]^T$ using $s = y/\|y\|$. To improve fairness of the comparison we replaced the visual height controller with the PID controller used in our experiments. In this simulation the complete dynamics (2.6) is used. The inner loop is controlled by a well tuned PD controller. We use the controller gains $k_{p\theta} = k_{p\phi} = 16$, $k_{d\theta} = k_{d\phi} = 6.4$, $k_{pz} = 16$, $k_{iz} = 0.1$ and $k_{dz} = 6.4$. The spherical coordinates are calculated assuming no calibration error.

In all three simulation cases we use the same controller gains $k_1 = 0.6$ and $k_2 = 0.49$. The simulations take the same value for the scaled focal length as $\lambda = f/\rho \approx 2.8$ mm/2.75 μm ≈ 1020 pixels. The camera takes images at a rate of 20 Hz with a resolution of 640×480 pixels. This is in-line with the capabilities of the Pixy. We initialize the vehicle's height to 1 m above the ground (i.e., $p_3^n(0) = -1$ m) and take a desired height setpoint of 1.5 m (i.e., $p_3^{n*} = -1.5$ m). A summary of the simulation parameters is in Table 4.1.

Figure 4.2 and Figure 4.3 contain the trajectories of y, θ, ϕ, v_1^b, and v_2^b for the ideal and non-ideal cases, respectively. The performance of the non-ideal simulation is degraded but still convergent. The speed of response in y is similar in both cases, but there is more overshoot in the non-ideal case. The θ, ϕ trajectories in the non-ideal case are oscillatory. Other choice of inner loop gains can limit the oscillations but would require too much control effort to be practically useful. In fact, in flight tests the range of stable inner loop gains is quite limited and finding stable gains was a significant challenge. In Figure 4.4 the system states using controller (4.17)

Parameter	Value
k_1	0.6
k_2	0.49
g	9.81 m/s^2
$p_3^n(0)$	-1.0 m
p_3^{n*}	-1.5 m
$v^n(0)$	$(0, 0)$ m/s
Image Size	640×480 pixels
$y(0)$	$(150, -150)$ pixels

Table 4.1: Controller gains and simulation parameters for state transformation experiment.

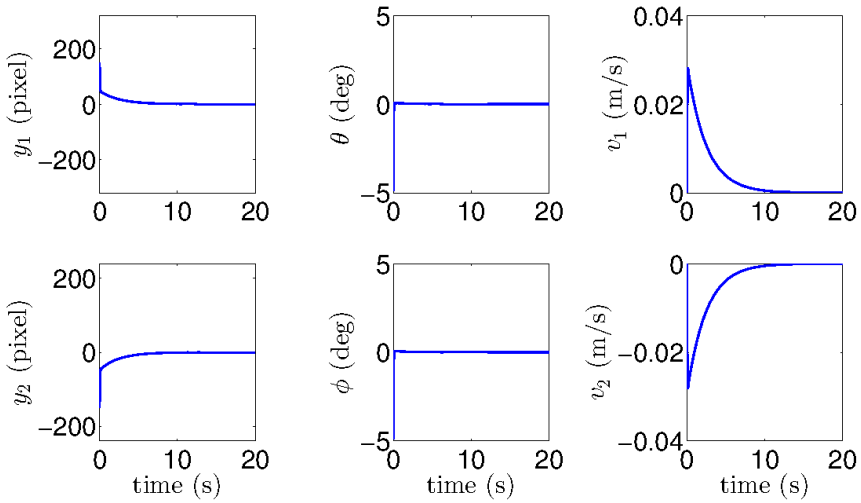


Figure 4.2: Ideal simulation results of the two DoF dynamics Σ_{2D} with perfect inner loop control of roll and pitch angles.

are plotted in solid blue line and those corresponding to the proposed controller are plotted with a dashed line. From the figure we can see that the performance of the two controllers has been tuned to be similar. The trajectory of y in the camera image is shown in Figures 4.5a to 4.5c. The initial image feature location is shown in red, and the goal location is shown in green. The trajectories of the image feature in the image plane are shown in Figure 4.5c. The trajectories for the proposed controller and (4.17) are almost identical.

4.1.4 Experimental Results

In this section we present the experimental performance of the proposed control on the Applied Nonlinear Control Lab (ANCL) quadrotor unmanned aerial vehicle (UAV) platform using Quadrotor I. This demonstrates the method's robustness to various model uncertainties and the feasibility of implementation on embedded hard-

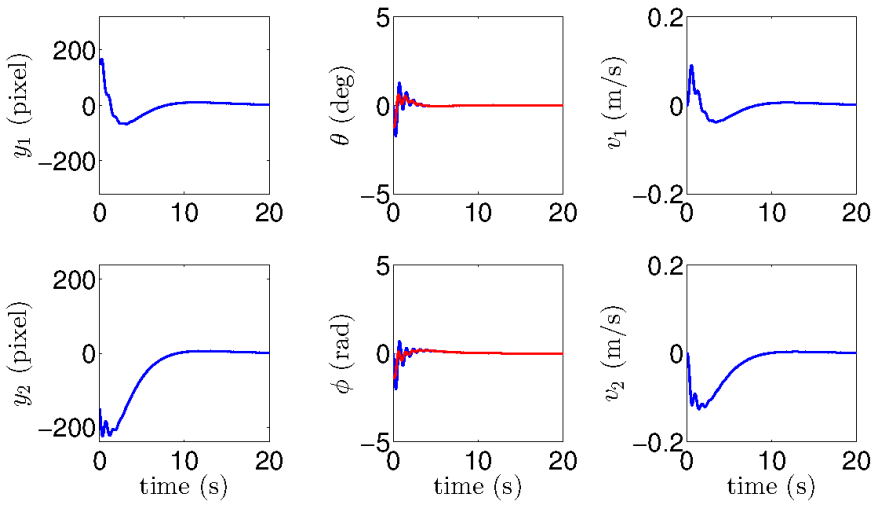


Figure 4.3: System trajectories for the non-ideal case in simulation. The inner loop roll-pitch tracking is achieved with a PD controller. The simulation includes the disturbances entering the image feature kinematics which where neglected in the control design. The system state is in blue and the reference angles for the inner loop are in red.

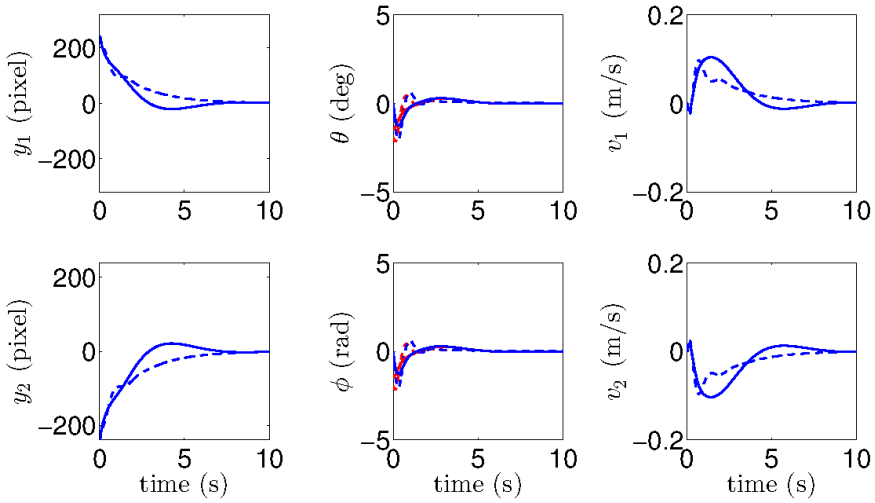
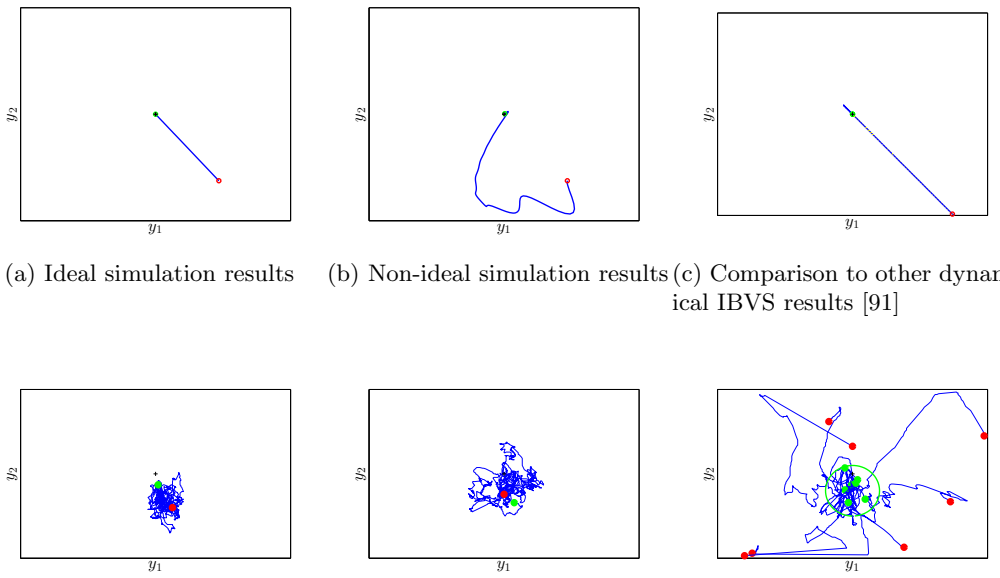


Figure 4.4: Comparison of the proposed approach with that in [91]. The system states for the proposed control are in dashed blue, and the results for the control in [91] are in solid blue. The reference angles for the inner PD loop are in red.



(a) Ideal simulation results

(b) Non-ideal simulation results

(c) Comparison to other dynamical IBVS results [91]

(d) Experimental results

(e) Other dynamic IBVS Experimental results [91]

(f) Experimental image feature trajectories with multiple initial conditions.

Figure 4.5: Image feature trajectories for the experimental and simulation results. The initial image feature location is shown in red, and the goal location is in green.

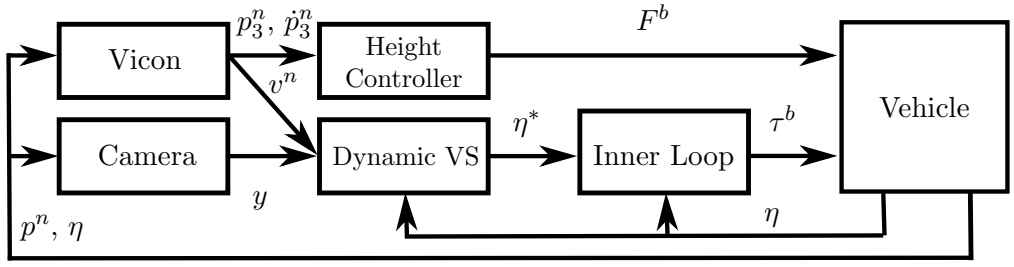


Figure 4.6: Block diagram of the proposed control structure.

ware where resources are limited. Additionally, we implemented the controller (4.17) from [91] for comparison. In practice our approach requires the implementation of an inner attitude control. We choose a simple PID control structure for roll and pitch as is often used in practice. For safety, tuning of the inner loop gains required an outer loop position control due to the coupled nature of the rotational and translational dynamics. The Vicon system initially provided position feedback in three DoF to facilitate the tuning stage. Once a satisfactory inner-loop performance was obtained, we implemented the DIBVS control one axis at time. The control gains were tuned online to obtain $k_1 = 0.6$ and $k_2 = 0.49$. The lateral velocity estimates used in the IBVS control law were obtained from the Vicon system. Yaw was regulated using proportional control and the onboard magnetometer. The use of a magnetometer to control yaw indoors was possible even with the existence of disturbance magnetic fields. This was because stabilization only required a constant reference value; not necessarily an accurate measurement of the earth's magnetic field. The height was controlled using PID and the Vicon estimate of altitude and vertical velocity. The experiments were performed in two phases. An initial phase consisted of using the Vicon to control all positional DoFs. This allowed us to avoid ground effects and ensure the image feature was in the camera's field of view. The second phase involved switching to the proposed control. A block diagram of the control structure used in the experiments is shown in Figure 4.6.

We demonstrated the robustness of the controller to a wide range of initial conditions in $p_1^n, p_2^n, p_3^n, \psi$. The p_1^n and p_2^n position were varied up to 90 cm in all directions from the desired visual feature. The quadrotor was always initialized at about 25 cm below the desired height. The initial yaw was varied in a range of 60 degrees. The height and yaw controllers would then regulate the height to $p_3^{n*} = -1.5$ m and yaw to $\psi^* = 0$. The experiments show that disturbances due to nonzero v_3 and $\dot{\psi}$ do not affect closed-loop performance. As seen in Figure 4.5f, the controller successfully stabilizes the wide range of initial conditions.

We evaluate the steady-state performance of the controllers in Figure 4.7 and Figure 4.5d. Clearly, the proposed control demonstrates good hover performance with y_1, y_2 having maximum amplitudes of 71 and 102 pixels, respectively. The Vicon data shows that this corresponds to variation in displacements of 0.18 m and 0.32 m in the n_1 and n_2 axes, respectively. Roll and pitch angles have relatively small variation and exhibit non-zero average values due to non-ideal weight distribution in the vehicle. Such a weight imbalance means zero roll and pitch create large lateral velocity and cause the image feature to rapidly leave the camera's field of view. The experimental results demonstrate robustness to unmodelled dynamics neglected in the design.

To further evaluate the robustness of the proposed controller, the visual target

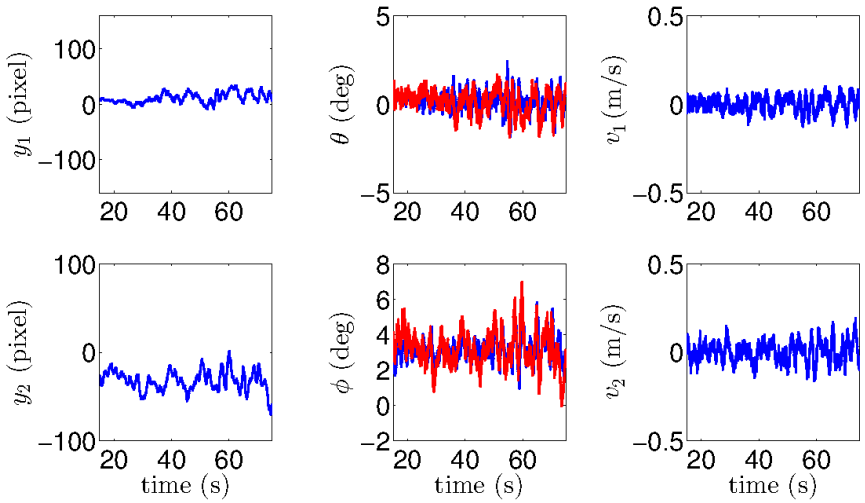


Figure 4.7: Experimental results of the proposed algorithm. System states are shown in blue and the reference angles for the inner PD loop are in red.

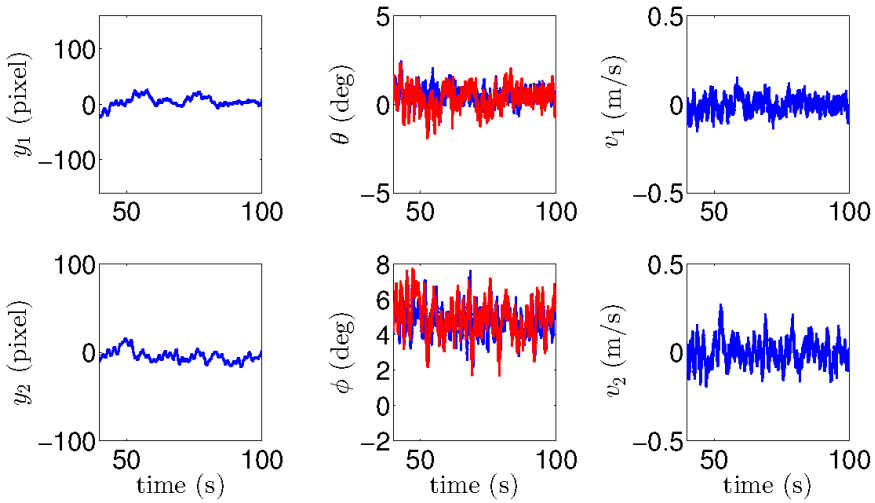
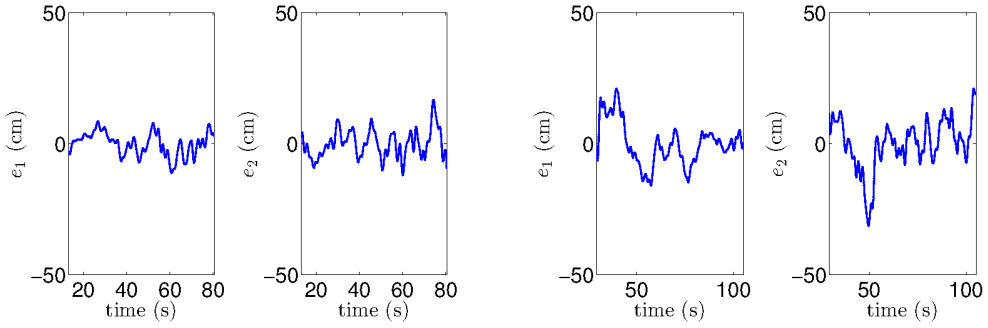


Figure 4.8: Experimental results of the control in [91]. System states are shown in blue and the reference angles for the inner PD loop are in red.



(a) Proposed Controller (b) Controller from [91]

Figure 4.9: Experimental results of the error in lateral position for the proposed control and that in [91].

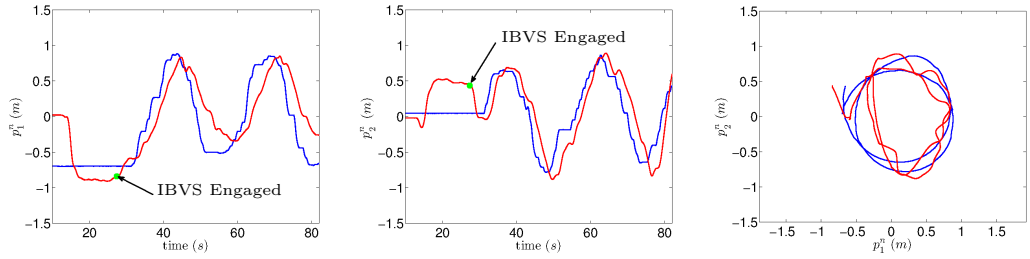


Figure 4.10: Experimental results for a moving target. The quadrotor trajectory is red and target trajectory is blue.

was attached to a remote control car that was driven around the lab. Vicon markers were placed on the car to establish ground truth data. As seen in Figure 4.10 the tracking error is similar in magnitude to the static target experiment.

As in Section 4.1.3 we implemented the DIBVS control in [91]. As it is difficult to compare controller performance on different experimental platforms, implementing this control on our platform allows for a fair comparison with the proposed method. The experimental results of (4.17) are shown in Figure 4.8. The performance is similar to the proposed controller. Figure 4.9 shows that in both experiments the vehicle remained within a radius of about 30 cm from the origin. The lateral errors are defined as $e_1 = p_1^{n*} - p_1^n$ and $e_2 = p_2^{n*} - p_2^n$. For the proposed control the standard deviation of the lateral errors were 4 cm and 5 cm for the n_1 and n_2 axis, respectively. For (4.17) the standard deviation of the lateral errors were 8 cm and 9 cm for the n_1 and n_2 axis, respectively. Other experimental quadrotor results such as [64] show similar results with a non-vision-based position controller and without compensation for advanced aerodynamic effects such as blade flapping and drag force. Their position control which compensates for these effects improved hovering performance from 40 cm to 10 cm radial error. On the other hand, our controller does not rely on position estimates and hovers in a radius of about 30 cm. Another

experimentally validated non-vision-based position controller is presented in [98]. This control provides hover performance in a circle of about 20 cm radius. Few experimental results for IBVS for UAVs have been published. Even fewer results exist for DIBVS laws. An exception is the work in [110] which presents a comparison of a number of IBVS algorithms including that in [91]. Some of these controllers are DIBVS and have hover performance similar to the proposed controller. However, they have increased complexity due to the spherical projection used.

4.1.5 Summary

In this section we investigated the control of a quadrotor using nonlinear DIBVS to regulate the lateral position of the vehicle relative to a static visual feature point on the ground. The control relies on a single visual feature point which can be robustly obtained and can be defined generically depending on the application. The convergence of the proposed control law is proven to be locally exponentially stable and an estimate of the region of attraction is provided. An indoor quadrotor platform was implemented and used to validate the control successfully. The results demonstrate robustness to various unmodelled effects including non-ideal inner loop performance and variation in the uncontrolled DoF (i.e., yaw and altitude) which are modelled as a disturbance. The proposed control is compared to a state-of-the-art DIBVS approach and benefits from reduced complexity with a similar level of performance.

4.2 Virtual Camera-Based Approach

4.2.1 Image Moments

Image moments are important visual features for VS as they are simple to calculate and can provide decoupled image feature kinematics. Let $\mathcal{O}(t)$ be a dense object with a continuous surface in the image $\pi(t)$ and is defined by a set of closed contours $\mathcal{C}(t)$ as shown in Figure 4.11. For the sake of clarity of presentation this section breaks from the standard notation defined in this thesis and instead of denoting the position of the target in frame $\mathcal{C}$ as $p^c = [p_1^c, p_2^c, p_3^c]^T$ we denote it $P = [X, Y, Z]^T$. Instead of using $y = [y_1, y_2]^T$ we denote a point in the image plane we use $p = [x, y]^T$. Instead of using $v^c = [v_1^c, v_2^c, v_3^c]^T$ we denote the velocity of the target in frame $\mathcal{C}$ as $v = [v_x, v_y, v_z]^T$. Instead of using $\omega^c = [\omega_1^c, \omega_2^c, \omega_3^c]^T$ we denote the angular velocity of the camera in $\mathcal{C}$ as $\omega = [\omega_x, \omega_y, \omega_z]^T$.

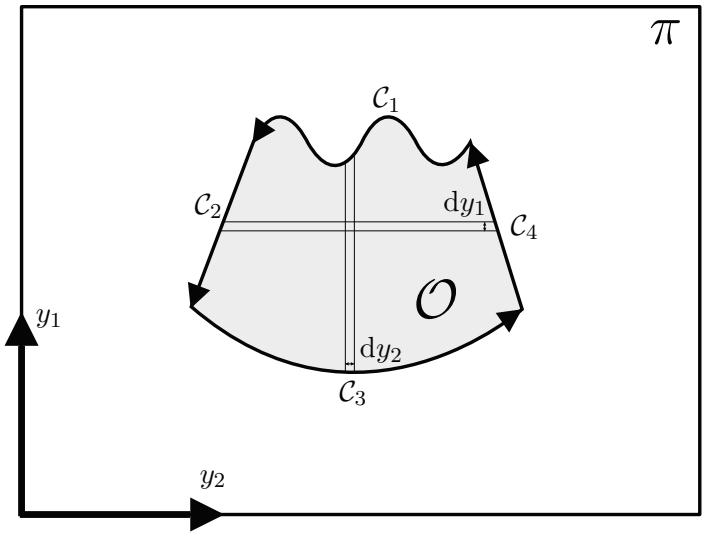


Figure 4.11: A dense object $\mathcal{O}$ defined in the binary image π by a closed set of contours $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$.

Continuous Image

We begin by examining a continuous image of the object [111]. The moments m_{ij} of the image are

$$m_{ij}(t) = \int \int_{\mathcal{O}} f(x, y)\, \mathrm{d}x\, \mathrm{d}y \tag{4.18}$$

where $f(x, y) = x^i y^j$. Centred moments are of particular interest to visual servoing as they are invariant to 2 D translational motion. The centred moments are

$$\mu_{ij}(t) = \int \int_{\mathcal{O}} f_\mu(x, y)\, \mathrm{d}x\, \mathrm{d}y \tag{4.19}$$

where $f_\mu = (x - x_g)^i (y - y_g)^j$, the geometric centre of the object is $c_g = [x_g, y_g]^T$ and

$$x_g = m_{10}/m_{00} \tag{4.20}$$

$$y_g = m_{01}/m_{00} \tag{4.21}$$

Next the image feature kinematics with respect to the camera are calculated by differentiating the image moments with respect to time.

$$\dot{m}_{ij} = \int \int_{\mathcal{O}} \left(\tfrac{\partial f}{\partial x}\dot{x} + \tfrac{\partial f}{\partial y}\dot{y} + f(x, y) \left(\tfrac{\partial \dot{x}}{\partial x} + \tfrac{\partial \dot{y}}{\partial y} \right) \right) \mathrm{d}x\, \mathrm{d}y \tag{4.22}$$

Similarly, for the centred image moments

$$\dot{\mu}_{ij} = \int\int_{\mathcal{O}} \left(\frac{\partial f_\mu}{\partial x}(\dot{x} - \dot{x}_g) + \frac{\partial f_\mu}{\partial y}(\dot{y} - \dot{y}_g) + f_\mu(x,y)\left(\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y}\right)\right) \mathrm{d}x\,\mathrm{d}y \tag{4.23}$$

To be able to do VS the interaction matrix or Jacobian L_{ij} for each moment needs to be calculated.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = L_{ij} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.24}$$

To begin, the depth of the object Z is expressed as a continuous function of the image coordinates.

$$\frac{1}{Z} = \sum_{p\geq 0, q\geq 0} A_{pq} x^p y^q \tag{4.25}$$

Next the single point image kinematics (2.34) are combined with (4.22), (4.24) and (4.25).

$$L_{ij} = \begin{bmatrix} -\sum_{p,q}(i+p)A_{pq}m_{i+p-1,j+q} \\ -\sum_{p,q}(j+q)A_{pq}m_{i+p,j+q-1} \\ \sum_{p,q}(i+j+p+q+2)A_{pq}m_{i+p,j+q} \\ (i+j+3)m_{i,j+1} + jm_{i,j-1} \\ (i+j+3)m_{i+1,j} - im_{i-1,j} \\ im_{i-1,j} - jm_{i+1,j-1} \end{bmatrix}^T \tag{4.26}$$

However, the translational terms of the interaction matrix for order $i+j$ require image moments at a minimum of order $i+j+p+q+2$. Furthermore, they require all of the terms A_{pq} and therefore are not a good candidate for VS. However, the kinematics can be simplified by assuming that the object is planar and thus the points are related by the following constraints

$$\frac{1}{Z} = Ax + By + C \tag{4.27}$$

The combination of (4.27), (2.34), and (4.24) is

$$\dot{x} = -(Ax+By+C)v_x + x(Ax+By+C)v_z + xy\omega_x - (1+x^2)\omega_y + y\omega_z \tag{4.28}$$

$$\dot{y} = -(Ax+By+C)v_y + y(Ax+By+C)v_z + (1+y^2)\omega_x - xy\omega_y - x\omega_z \tag{4.29}$$

Thus

$$\begin{aligned}
\frac{\partial f}{\partial x} &= ix^{i-1}y^{j} \\
\frac{\partial f}{\partial y} &= jx^{i}y^{j-1} \\
\frac{\partial \dot{x}}{\partial x} &= -Av_x + (2Ax + By + C)v_z + y\omega_x - 2x\omega_y \\
\frac{\partial \dot{y}}{\partial y} &= -Bv_y + (Ax + 2By + C)v_z + 2y\omega_x - x\omega_y
\end{aligned} \tag{4.30}$$

The interaction matrix can then be calculated by substituting (4.30) into (4.22).

$$L_{ij} = \begin{bmatrix} -i(Am_{ij} + Bm_{i-1,j+1} + Cm_{i-1,j}) - Am_{ij} \\ -j(Am_{i+1,j-1} + Bm_{ij} + Cm_{i,j-1}) - Bm_{ij} \\ (i+j+3)(Am_{i+1,j} + Bm_{i,j+1} + Cm_{ij}) - Cm_{ij} \\ (i+j+3)m_{i,j+1} + jm_{i,j-1} \\ -(i+j+3)m_{i+1,j} - im_{i-1,j} \\ im_{i-1,j+1} - jm_{i+1,j-1} \end{bmatrix}^T \tag{4.31}$$

Now the translational terms of the interaction matrix for order $i+j$ only requires image moments of order $i+j+1$. Similarly, the interaction matrix for the centred image moments is

$$L_{ij} = \begin{bmatrix} -(i+1)A\mu_{ij} - iB\mu_{i-1,j+1} \\ -jA\mu_{i+1,j-1} - (j+1)B\mu_{ij} \\ -A\mu_{wy} + B\mu_{wx} + (i+j+2)C\mu_{ij} \\ (i+j+3)\mu_{i,j+1} + ix_g\mu_{i-1,j+1} + (i+2j+3)y_g\mu_{ij} - 4i\dfrac{\mu_{11}}{m_{00}}\mu_{i-1,j} - 4j\dfrac{\mu_{02}}{m_{00}}\mu_{i,j-1} \\ -(i+j+3)\mu_{i+1,j} - (2i+j+3)x_g\mu_{ij} - jy_g\mu_{i+1,j-1} + 4i\dfrac{\mu_{20}}{m_{00}}\mu_{i-1,j} + 4j\dfrac{\mu_{11}}{m_{00}}\mu_{i,j-1} \\ i\mu_{i-1,j+1} - j\mu_{i+1,j-1} \end{bmatrix}^T \tag{4.32}$$

where $\alpha = i + j + 3$.

The next step is to choose what features will be good for VS. For simplicity we assume the object is parallel to image plane (i.e., $A = B = 0$ and $C = 1/Z$). Later in this thesis we will use this assumption as the basis for the virtual camera approach for IBVS. Next, we analyze a visual feature relating to the depth of the image. We choose the area moment. Let

$$s_z = a = m_{00} \tag{4.33}$$

The interaction matrix of the area feature is

$$\begin{aligned} L_{s_z} &= L_{00} \\ &= \begin{bmatrix} 0 & 0 & 2Cm_{00} & 3m_{01} & -3m_{01} & 0 \end{bmatrix} \\ &= a \begin{bmatrix} 0 & 0 & 2C & 3y_g & -3x_g & 0 \end{bmatrix} \end{aligned} \tag{4.34}$$

We can see that $\dot{s}_z$ depends linearly on a, but it is not decoupled. Normalizing the area is a better choice. Next we choose

$$s_z = Z^* \sqrt{a^*/a} \tag{4.35}$$

where a^* and Z^* are the desired area and depth of the object, respectively. Using the relationship $Z^*\sqrt{a^*} = Z\sqrt{a}$ we can solve for the interaction matrix of the normalized area feature.

$$L_{s_z} = \begin{bmatrix} 0 & 0 & -1 & \frac{-3s_z y_g}{2} & \frac{3s_z x_g}{2} & 0 \end{bmatrix} \tag{4.36}$$

Now the interaction matrix is decoupled. Next, we choose features for the lateral motion in the X and Y directions. Let $s_x = x_g = m_{10}/m_{00}$ and $s_y = y_g = m_{01}/m_{00}$. Then the interaction matrix for this feature is

$$\begin{bmatrix} L_{s_x} \\ L_{s_y} \end{bmatrix} = \begin{bmatrix} -C & 0 & Cx_g & (x_g y_g + \frac{4\mu_{11}}{m_{00}}) & -(1 + x_g^2 + \frac{4\mu_{20}}{m_{00}}) & y_g \\ 0 & -C & Cy_g & (1 + y_g^2 + \frac{4\mu_{20}}{m_{00}}) & -(x_g y_g + \frac{4\mu_{11}}{m_{00}}) & -x_g \end{bmatrix} \tag{4.37}$$

We can see that the $\dot{s}_x$ and $\dot{s}_y$ have the terms $-Cv_x$ and $-Cv_y$, respectively. and thus they are still coupled with v_z. We note that the coupling with ω_x and ω_y will not be a problem with the virtual camera approach because $\omega_x = \omega_y = 0$. Fortunately, we can use the same area normalization to correct this problem. Let

$$\begin{aligned} s_x &= x_g Z^* \sqrt{a^*/a} \\ s_y &= y_g Z^* \sqrt{a^*/a} \end{aligned} \tag{4.38}$$

then

$$\begin{bmatrix} L_{s_x} \\ L_{s_y} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & s_z(\frac{-x_g y_g}{2} + \frac{4\mu_{11}}{m_{00}}) & -s_z(1 - \frac{x_g^2}{2} + \frac{4\mu_{20}}{m_{00}}) & s_y \\ 0 & -1 & 0 & s_z(1 - \frac{y_g^2}{2} + \frac{4\mu_{02}}{m_{00}}) & -s_z(-\frac{x_g y_g}{2} + \frac{4\mu_{11}}{m_{00}}) & -s_x \end{bmatrix} \tag{4.39}$$

The image feature kinematics are now independent of v_z and depend linearly on v_x and v_y.

Now we move on to the orientation of the object θ. It can be calculated in various ways. We define the orientation by using the principal axis theorem. Second order

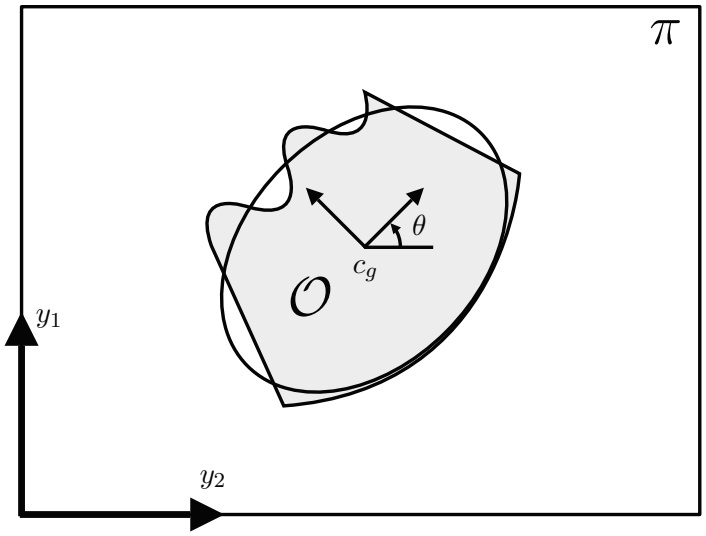


Figure 4.12: The orientation θ of the object $\mathcal{O}$ is calculated by approximating the object as an ellipse and measuring the orientation of the ellipse with respect to the image.

centred image moments are used to approximate an ellipse about the geometric centre of the object c_g and then we measure the orientation of the ellipse with respect to the image as seen in Figure 4.12.

$$\theta = 0.5 \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{4.40}$$

Then the interaction matrix is

$$L_{s_\psi} = \begin{bmatrix} 0 & 0 & 0 & \alpha & \beta & -1 \end{bmatrix} \tag{4.41}$$

where

$$\alpha = \frac{-(2\mu_{11}^2 + \mu_{02}(\mu_{02} - \mu_{20}))x_g + \mu_{11}(\mu_{20} + \mu_{02})y_g + 5(\mu_{12}(\mu_{20} - \mu_{02}) + \mu_{11}(\mu_{03} - \mu_{21}))}{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}$$
$$\beta = \frac{\mu_{11}(\mu_{20} + \mu_{02})x_g + -(2\mu_{11}^2 + \mu_{20}(\mu_{20} - \mu_{02}))y_g + 5(\mu_{21}(\mu_{02} - \mu_{20}) + \mu_{11}(\mu_{30} - \mu_{12}))}{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}$$

Discrete Image

Figure 4.13 shows the same dense object as before, but now in a 10×10 pixel image. Each pixel is a binary 1 or 0. With a perfect tracker all of the highlighted grey pixels will be found and hence be a 1. All the white pixels will be a 0. In this case we can see that object is fairly well represented in the image and there are

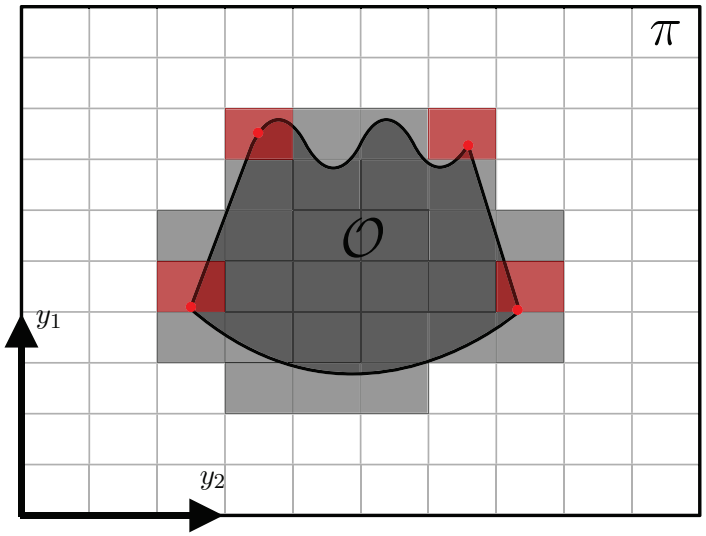


Figure 4.13: Discretization of a dense object $\mathcal{O}$ defined in the binary image π.

only a few minor discretization errors. The discretization errors become negligible in high resolution images. However, in tracking it is extremely difficult to correctly find one hundred percent of the pixels relating to the object. Even if the object is a single colour tracking often misses part of the object due to highlights from nearby lights and will often mislabel pixels that are not part of the image. On the other hand, trackers are good at following blobs. This allows us to instead represent the object by n points which are in turn blobs. This is a very practical solution to a currently difficult computer vision (CV) problem. It does require that either artificial markers are placed on the object being tracked, or visually distinct parts of the object can also be chosen. Another example to highlight the difference between the two methods is tracking a face. There are two main methods. The first option would be track by skin colour. This will give us the rough shape of a face, but it will exclude hair and eyes. It will also have problems with changes in lighting. The second option is to track the eyes. Then from the position of the eyes, the entire position of the face can be extrapolated. The drawback of tracking eyes is if one eye is closed or occluded, the face tracking fails. The choice of tracking method does not affect the subsequent image moments other then how the area feature is calculated.

To analyze the discrete image of the object we begin by defining the discrete image moment [112]. Given a set of n image points $\{p_1, \cdots, p_n\}$ where $p_i = \{x_i, y_i\}$ the discrete image moments are

$$m_{ij} = \sum_{k=1}^{n} x_k^i y_k^j \tag{4.42}$$

and for the discrete image the centred moments are

$$\mu_{ij} = \sum_{k=1}^{n} (x_k - x_g)^i (y_k - y_g)^j \tag{4.43}$$

Similar to the continuous case, the image feature kinematics for the image moments are

$$\dot{m}_{ij} = \sum_{k=1}^{n} \left(i x_k^{i-1} y_k^j \dot{x}_k + j x_k^i y_k^{j-1} \dot{y}_k \right) \tag{4.44}$$

and for the centred image moments they are

$$\dot{\mu}_{ij} = \sum_{k=1}^{n} \left(i x_k^{i-1} y_k^j (\dot{x}_k - \dot{x}_g) + j x_k^i y_k^{j-1} (\dot{y}_k - \dot{y}_g) \right) \tag{4.45}$$

Thus the interaction matrix for the image moments is

$$L_{ij} = \begin{bmatrix} -i(Am_{ij} + Bm_{i-1,j+1} + Cm_{i-1,j}) \\ -j(Am_{i+1,j-1} + Bm_{ij} + Cm_{i,j-1}) \\ (i+j)(Am_{i+1,j} + Bm_{i,j+1} + Cm_{ij}) \\ (i+j)m_{i,j+1} + jm_{i,j-1} \\ -(i+j)m_{i+1,j} - im_{i-1,j} \\ im_{i-1,j+1} - jm_{i+1,j-1} \end{bmatrix}^T \tag{4.46}$$

and for the centred image moments it is

$$L_{ij} = \begin{bmatrix} -iA\mu_{ij} - iB\mu_{i-1,j+1} \\ -jA\mu_{i+1,j-1} - jB\mu_{ij} \\ -A\mu_{wy} + B\mu_{wx} + (i+j)C\mu_{ij} \\ (i+j)\mu_{i,j+1} + ix_g\mu_{i-1,j+1} + (i+2j)y_g\mu_{ij} - i\dfrac{\mu_{11}}{m_{00}}\mu_{i-1,j} - j\dfrac{\mu_{02}}{m_{00}}\mu_{i,j-1} \\ -(i+j)\mu_{i+1,j} - (2i+j)x_g\mu_{ij} - jy_g\mu_{i+1,j-1} + i\dfrac{\mu_{20}}{m_{00}}\mu_{i-1,j} + j\dfrac{\mu_{11}}{m_{00}}\mu_{i,j-1} \\ i\mu_{i-1,j+1} - j\mu_{i+1,j-1} \end{bmatrix}^T \tag{4.47}$$

If we want to use the same images features used for continuous images, we must first reexamine the area feature. Recalling in the continuous case the area is $a = m_{00}$. If the entire object is being tracked in the image then the area feature can remain the same. The area is simply the number of pixels/points and $\dot{a} = \dot{m}_{00} = \dot{n}$. However, if the object is represented by a fixed set of n points that are being tracked on the object then m_{00} is a poor choice for the area feature as n is constant and $\dot{a} = 0$.

Instead, second order centred image moments can be used

$$a = \mu_{02} + \mu_{20} \tag{4.48}$$

From here we can re-use the same moments image features as in (4.38), (4.38) and (4.40) and following the same method used for continuous images and assuming the object is parallel to the image plane we calculate the interaction matrix to be

$$\begin{bmatrix} \dot{s}_x \\ \dot{s}_y \\ \dot{s}_z \\ \dot{s}_\psi \end{bmatrix} = \begin{bmatrix} L_{s_x} \\ L_{s_y} \\ L_{s_z} \\ L_{s_\psi} \end{bmatrix} \begin{bmatrix} v^c \\ \omega^c \end{bmatrix} \tag{4.49}$$

where

$$\begin{bmatrix} L_{s_x} \\ L_{s_y} \\ L_{s_z} \\ L_{s_\psi} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & M_1 & M_2 & s_y \\ 0 & -1 & 0 & M_3 & M_4 & -s_x \\ 0 & 0 & -1 & M_5 & M_6 & 0 \\ 0 & 0 & 0 & M_7 & M_8 & -1 \end{bmatrix} \tag{4.50}$$

where

$$M_1 = \frac{-s_z(x_g(y_g\mu_{02} + x_g\mu_{11} + \mu_{21} + \mu_{03})m_{00} - a\mu_{11})}{am_{00}}$$
$$M_2 = \frac{-s_z(am_{00} - x_g(x_g\mu_{20} + y_g\mu_{11} + \mu_{12} + \mu_{30})m_{00} + a\mu_{20})}{am_{00}}$$
$$M_3 = \frac{s_z(am_{00} - y_g(y_g\mu_{02} + x_g\mu_{11} + \mu_{21} + \mu_{03})m_{00} + a\mu_{02})}{am_{00}}$$
$$M_4 = \frac{-s_z(y_g(x_g\mu_{20} + y_g\mu_{11} + \mu_{12} + \mu_{30})m_{00} + a\mu_{11})}{am_{00}}$$
$$M_5 = \frac{-s_z(s_y + y_g\mu_{02} + x_g\mu_{11} + \mu_{21} + \mu_{03})}{a}$$
$$M_6 = \frac{s_z(s_x + x_g\mu_{20} + y_g\mu_{11} + \mu_{12} + \mu_{30})}{a}$$
$$M_7 = \frac{-(2\mu_{11}^2 + \mu_{02}(\mu_{02} - \mu_{20}))x_g + \mu_{11}s_y + 5(\mu_{12}(\mu_{20} - \mu_{02}) + \mu_{11}(\mu_{03} - \mu_{21}))}{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}$$
$$M_8 = \frac{\mu_{11}s_x + -(2\mu_{11}^2 + \mu_{20}(\mu_{20} - \mu_{02}))y_g + 5(\mu_{21}(\mu_{02} - \mu_{20}) + \mu_{11}(\mu_{30} - \mu_{12}))}{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}$$

Moving Target

To extend our work to moving targets we define the target in the navigation frame $\mathcal{N}$. We assume there are $n > 1$ target point correspondences in the images. These points

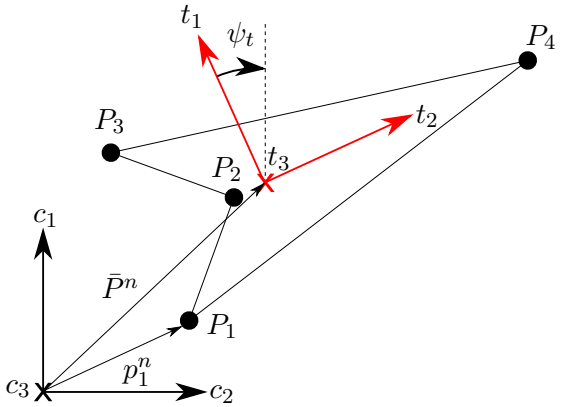


Figure 4.14: Target composed of $n = 4$ points denoted $p_i, 1 \leq i \leq 4$. The position of the geometric centre is denoted $\bar{p}^n = \sum_{i=1}^{n} p_i^n$, where p_k^i is p_i expressed in $\mathcal{N}$. The angle between target frame $\mathcal{T}$ and $\mathcal{N}$ is ψ_t.

make up a target which remains horizontal and can rotate about its geometric centre. Figure 4.14 shows a target composed of four (i.e., $n = 4$) points. A frame $\mathcal{T}$ is attached to the target. The origin of $\mathcal{T}$ is at the geometric centre of the target. We denote this geometric centre by vector $\bar{P}$. The basis of $\mathcal{T}$ is $\{t_1, t_2, t_3\}$ which is defined so that the $t_1 - t_2$ plane is parallel to the $n_1 - n_2$ plane, and $t_3 = n_3$. The yaw between $\mathcal{N}$ and $\mathcal{T}$ is denoted ψ_t and its positive direction is defined in Figure 4.14.

Summary

The image moments we have defined simplify the image kinematics. Returning to the notation used throughout this thesis the image moment feature $s = [s_1, s_2, s_3, s_4]^T$ is a function of the image coordinates $\{y_1, \cdots, y_n\}$,where $y_i = [y_{1i}, y_{2i}]^T$ and $1 \leq i \leq n$ in the camera $\mathcal{C}$ and is defined as

$$s_1 = s_3 y_{1g} \tag{4.51a}$$

$$s_2 = s_3 y_{2g} \tag{4.51b}$$

$$s_3 = \sqrt{\frac{\mu_{20}^* + \mu_{02}^*}{\mu_{20} + \mu_{02}}} \tag{4.51c}$$

$$s_4 = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{4.51d}$$

where

$$y_{1g} = \frac{1}{n}\sum_{k=1}^{n} y_{1k}$$
$$y_{2g} = \frac{1}{n}\sum_{k=1}^{n} y_{2k}$$
$$\mu_{ij} = \sum_{k=1}^{n} \left(y_{1k} - y_{1g}\right)^i \left(y_{2k} - y_{2g}\right)^j$$

and μ_{ij}^* is the desired value of μ_{ij} which corresponds to when the vehicle is in its desired pose. Then the image kinematics is

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \end{bmatrix} = \frac{-1}{{p_3}^*} \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{v}^v + \begin{bmatrix} s_2 \\ -s_1 \\ 0 \end{bmatrix} \dot{\psi} \tag{4.52a}$$

$$\dot{s}_4 = \dot{\tilde{\psi}} \tag{4.52b}$$

where p_3^* denotes desired depth and

$$\tilde{v} = R_{\tilde{\psi}}^T (v^n - \bar{v}_t^n)$$
$$\tilde{\psi} = \psi - \psi_t$$
$$R_\psi = \begin{bmatrix} \mathrm{c}_\psi & -\mathrm{s}_\psi & 0 \\ \mathrm{s}_\psi & \mathrm{c}_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and assuming that $\theta = \phi = \dot{\theta} = \dot{\phi} = 0$ and that the pixels are square $\lambda_1 = \lambda_2$. The image features s_1, s_2, s_3 are used to control the translational motion of the vehicle, and s_4 controls the yaw motion. Without loss of generality, we regulate the image feature s to a desired value $s^* = [0, 0, 1, 0]^T$. This ensures the vehicle's lateral position tracks the target's geometric centre, its yaw is the same as the target, and its height above the target is p_3^{v*}. Nonzero s_1^* and s_2^* values can be chosen, but are less practically relevant since we require the target to remain in the camera's limited field of view. The values of s_3^* and s_4^* can be chosen to regulate specific desired heights and yaw. We remark in practice that the choice of $s_1^* = s_2^* = 0$ does not imply that the camera is directly above the geometric centre of the target. An offset can exist due to misalignment between the body and camera frames; and the centre of mass and geometric centre.

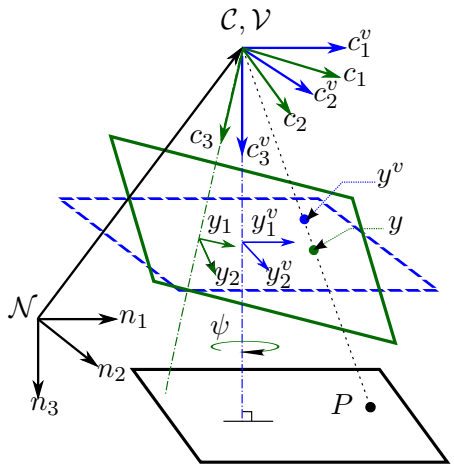


Figure 4.15: Pinhole camera geometry model and frame definition.

4.2.2 Fundamentals

Virtual Camera

In order to get rid of the roll and pitch dependency in (4.52) we introduce the notion of a virtual camera [46, 47]. The virtual camera has the same intrinsic parameters as the real camera and the same optical centre, but it is oriented such that its image plane is parallel to the n_1-n_2 ground plane. That is, the origin of the virtual camera frame, denoted $\mathcal{V}$, is fixed to the origin of $\mathcal{C}$ and moves with the vehicle. We denote the basis of $\mathcal{V}$ as $[c_1^v, c_2^v, c_3^v]$. The virtual and real cameras are shown in Figure 4.15.

A point expressed in the virtual camera frame is

$$p^v = [p_1^v, p_2^v, p_3^v]^T = R_{\theta\phi} p^c$$

where

$$R_{\theta\phi} = \begin{bmatrix} \mathrm{c}_\theta & \mathrm{s}_\phi \mathrm{s}_\theta & \mathrm{c}_\phi \mathrm{s}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix}$$

Its kinematics is

$$\dot{p}^v = -\dot{\psi}\, \mathrm{sk}\,(n_3) p^v - v^v \tag{4.53}$$

where $v^v = [v_1^v, v_2^v, v_3^v]^T = R_{\theta\phi} v^c$ is the velocity of the camera expressed in $\mathcal{V}$ [47]. The corresponding projected image point y^v in the virtual camera image plane is

$$y^v = [y_1^v, y_2^v]^T = \left[\lambda_1 \frac{p_1^v}{p_3^v}, \lambda_2 \frac{p_2^v}{p_3^v} \right]^T \tag{4.54}$$

The conversion from the real camera image coordinate to virtual camera image

coordinate is

$$y^v = \frac{1}{\beta} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \end{bmatrix} R_{\theta\phi} \begin{bmatrix} \lambda_2 y_1 \\ \lambda_1 y_2 \\ \lambda_1 \lambda_2 \end{bmatrix} \tag{4.55}$$

where

$$\beta = n_3^T R_{\theta\phi} \begin{bmatrix} \lambda_2 y_1 \\ \lambda_1 y_2 \\ \lambda_1 \lambda_2 \end{bmatrix} = \lambda_1 \lambda_2 \mathrm{c}_\phi \mathrm{c}_\theta - y_1 \lambda_2 \mathrm{s}_\theta + y_2 \lambda_1 \mathrm{s}_\phi \mathrm{c}_\theta$$

Based on (4.53) and (4.54) we can solve for the image kinematics in the virtual camera:

$$\dot{y}^v = \frac{1}{p_3^v} \begin{bmatrix} -\lambda_1 & 0 & y_1^v \\ 0 & -\lambda_2 & y_2^v \end{bmatrix} v^v + \frac{1}{\lambda_1 \lambda_2} \begin{bmatrix} \lambda_1^2 y_2^v \\ -\lambda_2^2 y_1^v \end{bmatrix} \dot{\psi}$$

These kinematics have the important property that they are independent of the roll and pitch rates unlike those of the real camera in (2.34). In fact, the virtual camera can be seen as defining new state coordinates in which dependence on roll and pitch rates is eliminated [113]. The new coordinates are similar in nature to those used in our state transformation [54] where they were also used to eliminate roll and pitch rates.

Image Moments in the Virtual Camera

The features defined in (4.51) are selected to control the motion of the vehicle. The assumption that $\phi = \theta = \dot{\phi} = \dot{\theta} = 0$ is no longer needed as they are defined to be identically zero in the virtual camera.

4.2.3 Control

We define a control law to regulate the relative position and yaw of the vehicle to a stationary target located on the ground. To derive the control we make two assumptions that simplify the presentation, but are not necessary. The first is a small angle approximation i.e., we assume that the roll and pitch angles are small. Hence, thrust in the virtual camera frame is

$$\begin{aligned} F^v &= F_3^v [-\mathrm{s}_\theta \mathrm{c}_\phi, \mathrm{s}_\phi, -\mathrm{c}_\theta \mathrm{c}_\phi]^T \\ &\approx F_3^v [-\theta, \phi, -1]^T \end{aligned} \tag{4.56}$$

This is a very practical assumption as the vehicle has to have small roll and pitch angles to keep the target in the camera's field of view. The second assumption is that the inertia matrix is diagonal $J = \mathrm{diag}(J_1, J_2, J_3)$ i.e., the quadrotor is symmetrical. To derive a control law we write the dynamics (2.6) with the image moment

kinematics (4.52) as

$$\dot{s}_1 = -k_2 v_1^v + s_2 \dot{\psi} \tag{4.57a}$$
$$\dot{s}_2 = -k_2 v_2^v - s_1 \dot{\psi} \tag{4.57b}$$
$$\dot{s}_3 = -k_1 v_3^v \tag{4.57c}$$
$$\dot{v}_1^v = v_2^v \dot{\psi} + u_\theta \tag{4.57d}$$
$$\dot{v}_2^v = -v_1^v \dot{\psi} + u_\phi \tag{4.57e}$$
$$\dot{v}_3^v = u_F \tag{4.57f}$$
$$\dot{s}_4 = -\dot{\psi} \tag{4.57g}$$
$$\dot{\psi} = u_\psi \tag{4.57h}$$

where $k_1 = 1/p_3^{v*}$, $k_2 = \lambda k_1$, and the inputs are $u_F = -F_3^v/m + g$, $u_\theta = -F_3^v \theta/m$, $u_\phi = F_3^v \phi/m$ and $u_\psi = \int_0^t \tau_3^b / J_3$.

Theorem 4.2. *The subsystem* (4.57a) *to* (4.57f) *is globally asymptotically stable (GAS) and the subsystem* (4.57g) *and* (4.57h) *is globally exponentially stable (GES) with the input*

$$u_F = -k_{h2}\left(\frac{1}{k_{h1}} v_3^v - s_3 + 1\right) \tag{4.58a}$$
$$u_\theta = -k_{\ell 2}(v_1^v/k_{\ell 1} - s_1) \tag{4.58b}$$
$$u_\phi = -k_{\ell 2}(v_2^v/k_{\ell 1} - s_2) \tag{4.58c}$$
$$u_\psi = k_\psi / J_3 s_4 \tag{4.58d}$$

where k_{h1}, k_{h2}, $k_{\ell 1}$, $k_{\ell 2}$ and k_ψ are positive constant gains such that $p_3^{v*} k_{h2} > k_{h1}^2$ and $p_3^{v*} k_{\ell 2} > k_{\ell 1}^2$.

Proof. First we consider the translational motion subsystem (4.57a) to (4.57f) and define error signals

$$\delta_1 = \frac{1}{\lambda_1} s_1$$
$$\delta_2 = \frac{1}{\lambda_2} s_2$$
$$\delta_3 = s_3 - 1$$
$$\delta_4 = v_1^v/k_{\ell 1} - \delta_1$$
$$\delta_5 = v_2^v/k_{\ell 1} - \delta_2$$
$$\delta_6 = v_3^v/k_{h1} - \delta_3$$

We consider the Lyapunov function candidate

$$V_1 = \frac{1}{2}\delta_3^2 + \frac{1}{2}\delta_6^2$$

so that

$$\begin{aligned}\dot{V}_1 = &- k_{h1}k_1\left(\delta_3^2 + \delta_3\delta_6\right) + \delta_6 u_F/k_{h1} \\ &+ k_{h1}k_1\delta_6(\delta_3 + \delta_6)\end{aligned}$$

Let $u_F = -k_{h2}\delta_6$ then

$$\dot{V}_1 = -k_{h1}k_1\delta_3^2 - (1/k_{h1})\left(k_{h2} - k_{h1}^2k_1\right)\delta_6^2$$

If $k_{h2} > k_{h1}^2k_1$ then $\dot{V}_1$ is negative definite. Next we consider the Lyapunov function candidate

$$V_2 = \frac{1}{2}\delta_1^2 + \frac{k_2}{2}\delta_2^2 + \frac{1}{2}\delta_4^2 + \frac{k_2}{2}\delta_5^2$$

then

$$\dot{V}_2 = (u_\theta\delta_4 + u_\phi\delta_5)/k_{\ell 1} + k_{\ell 1}k_1\left(-\delta_1^2 - \delta_2^2 + \delta_4^2 + \delta_5^2\right)$$

Let $u_\theta = -k_{\ell 2}\delta_4$ and $u_\phi = -k_{\ell 2}\delta_5$ then

$$\dot{V}_2 = -k_{\ell 1}k_1(\delta_1^2 + \delta_2^2) - (k_{\ell 2} - k_1k_{\ell 1}^2)(\delta_4^2 + \delta_5^2)/k_{\ell 1}$$

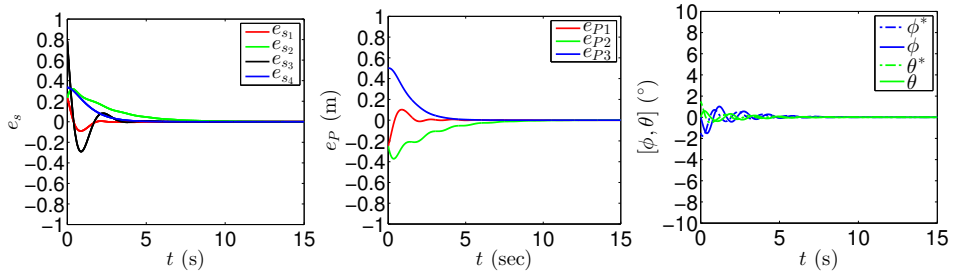
If $k_{\ell 2} > k_{\ell 1}^2k_1$ then $\dot{V}_2$ is negative definite. Lastly, consider the Lyapunov function candidate $V = V_1 + V_2$ and then $\dot{V}$ is negative definite and the subsystem (4.57a) to (4.57f) is GAS. For the yaw subsystem (4.57g) and (4.57h) we substitute the control law (4.58d) into the dynamics (4.52b) and obtain the closed loop dynamics

$$\dot{s}_4 = -k_\psi s_4$$

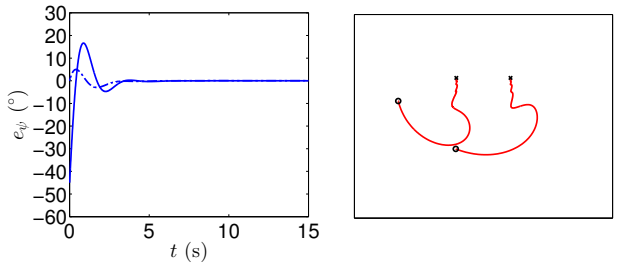
which is GES. □

4.2.4 Simulation Results

To validate the proposed control we simulated it in Simulink. The simulation includes the rigid body dynamics and kinematics from (2.6), the pinhole camera from (2.33), and the control laws (4.58). We use the quadrotor and camera parameters in Table 3.1. We used the PID inner loop control in (2.19). We remark the control law for torque (2.19) combined with u_F determine the propeller velocities based on the simplified model given in Section 2.1. To calculate the desired yaw $\hat{\psi}^*$ used in the inner loop we integrate the control from the outer loop $\hat{\psi}^* = \int \dot{\psi}^*(\tau)d\tau = \int u_\psi(\tau)d\tau$



(a) Error in image features s (b) Error in position P^n (c) Roll ϕ and pitch θ



(d) Error in yaw e_ψ (e) Image feature trajectory

Figure 4.16: Simulation results of dynamic IBVS control law.

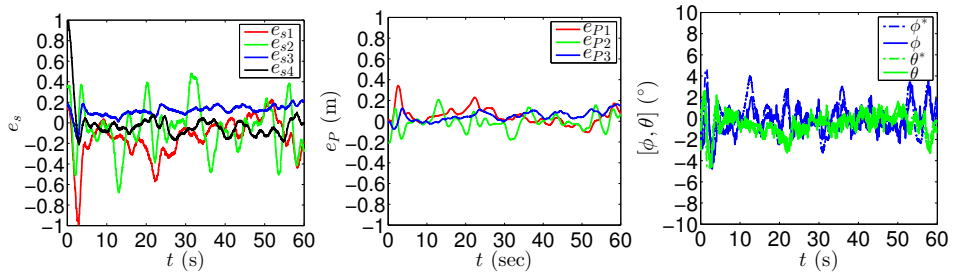
In the simulations the initial quadrotor position is $p^n(0) = [0.25, 0.25, -1]$ m and the desired position is $p^{n*} = [0, 0, -1.5]$ m. The initial yaw is $\psi(0) = 45°$ and desired yaw is $\psi^* = 0°$. Two visual markers were located on the ground 36 cm apart at $[-0.18, 0, 0]$ m and $[0.18, 0, 0]$ m. The controller gains are $k_{h1} = 4.4$, $k_{h2} = 1.0$, $k_{\ell 1} = 1.2$ and $k_{\ell 2} = 0.4$. Lastly, the thrust coefficient was taken to be $k_f = 40$ N. A summary of these parameters is in Table 4.2. The simulation results are shown in Figure 4.16. Figure 4.16a shows the error of the image features e_s rapidly converging to zero where $e_{si} = s_i^* - s_i, i = 1, 2, 3, 4$. Similarly in Figure 4.16b the error in position of the quadrotor converges to zero where $e_{pi} = p_i^{n*} - p_i^n, i = 1, 2, 3$. In Figure 4.16c, the roll ϕ and pitch θ angles (solid lines) converge to their zero reference values (dashed lines) and in Figure 4.16d the error in the yaw e_ψ (solid line) converges to zero as does the error in the desired yaw (dashed line) defined by $e_\psi^* = \psi^* - \hat{\psi}^*$. Finally, in Figure 4.16e the trajectories of the image point features are shown in the camera image. The image features at time $t = 0$ s are denoted by a circle and at time $t = 30$ s by an x.

4.2.5 Experimental Results

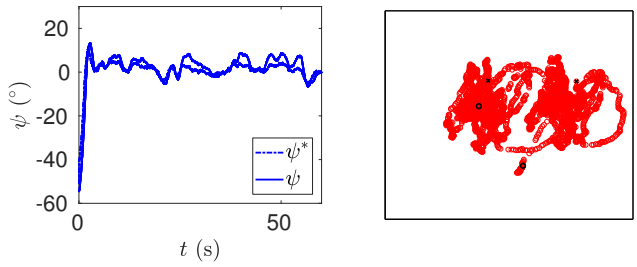
The experiments performed on the ANCL platform were similar to the simulations. We used the same controller gains as in Table 4.2. In the experiment two visual markers were placed on the ground 36 cm apart. The quadrotor was placed between

Parameter	Value
$[k_{h1}, k_{h2}]$	$[4.4, 1.0]$
$[k_{\ell 1}, k_{\ell 2}]$	$[1.2, 0.4]$
k_ψ	0.60
$[k_{p\phi}, k_{i\phi}, k_{d\phi}]$,	$[10, 1, 3]$
$[k_{p\theta}, k_{i\theta}, k_{d\theta}]$,	$[10, 1, 3]$
$[k_{p\psi}, k_{i\psi}, k_{d\psi}]$,	$[6, 1, 3]$
k_f	40 N
$p^n(0)$	$[0.25, 0.25, -1.0]$ m
p^{n*}	$[0, 0, -1.5]$ m
$\psi(0)$	45°
ψ^*	0°

Table 4.2: Simulation and experimental parameters for virtual camera experiment



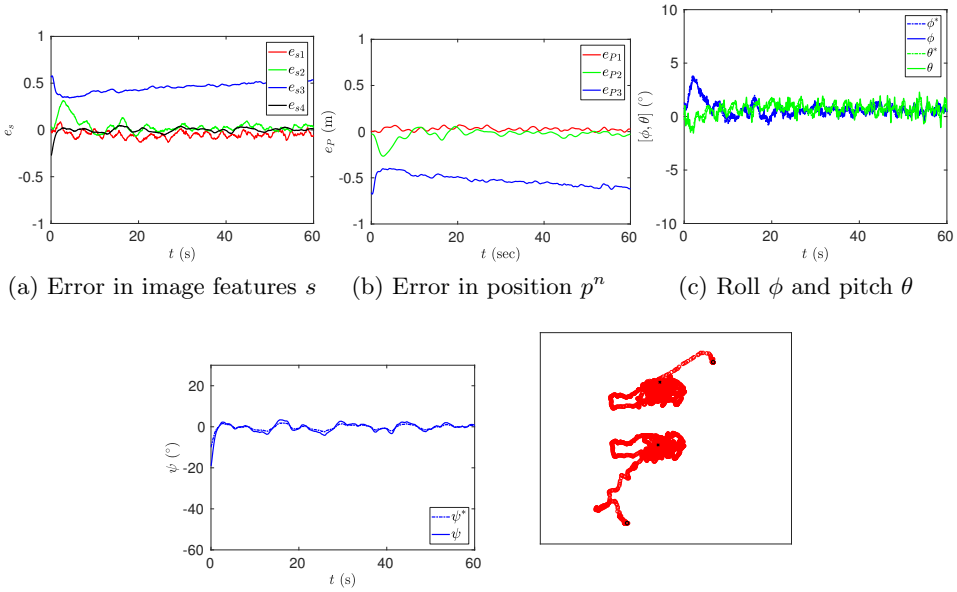
(a) Error in image features s (b) Error in position P^n (c) Roll ϕ and pitch θ



(d) Yaw ψ

(e) Image feature trajectory

Figure 4.17: Experimental results of dynamic IBVS control law using ANCL's Quadrotor I.



(a) Error in image features s (b) Error in position p^n (c) Roll ϕ and pitch θ

(d) Yaw ψ (e) Image feature trajectory

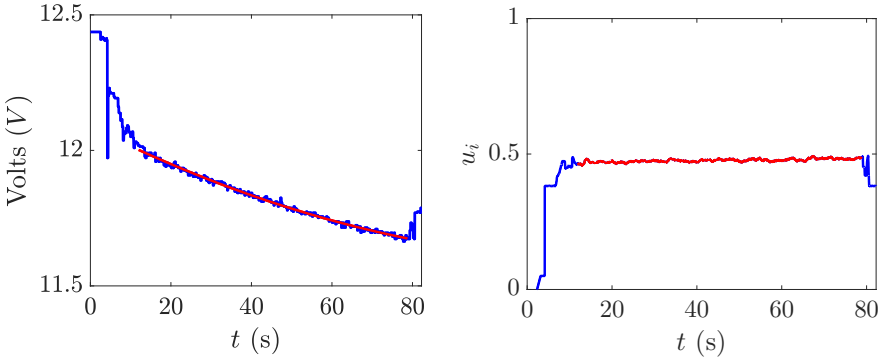
Figure 4.18: Experimental results of dynamic IBVS control law using ANCL's Quadrotor II.

the two markers and manually flown to about $[0.1, 0.1, -1.0]$ m to ensure that the markers were in the camera's field of view and to be above any ground effects. The initial yaw angle was about 50°. At this time the DIBVS controller was turned on and run for about 60 s. The experimental results are shown in Figure 4.17. Figure 4.17a shows the components of error in the image features e_s rapidly converging to a bounded trajectory near 0. We remark that the oscillations in s_1 and s_2 are probably caused by tracking error in the inner loop control and the performance is inline with our visual servoing experiments on the same platform in [54]. The error s_4 slowly converges to zero. Over the period of 60 s the height of the vehicle dropped about 10 cm. This is to be expected as the battery voltage drops and for the same thrust control signal less thrust will be generated. In Figure 4.17b error in vehicle position $e_{Pi} = p_i^{n*} - p_i^n, i = 1, 2, 3$ converges to a bounded trajectory near zero. Table 4.3 shows the mean and standard deviation of the error signals. The magnitude of errors are inline with our past work [54]. Relative to that work here we control all four DoF's (3 D position and yaw). Figure 4.17c shows the roll and pitch angles. In Figure 4.17d the error in the yaw angle ψ converges to zero. Finally, in Figure 4.17e the trajectories of the image point features are shown in the image plane in pixels. The image features at time $t = 0$ s are denoted by a circle and at time $t = 60$ s by an cross.

As seen in Figure 4.17a and Figure 4.17b, the height of the vehicle drops about

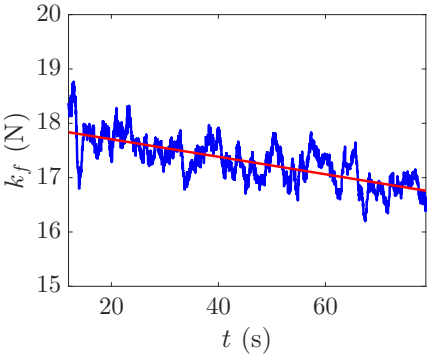
	Original Quadrotor Platform (Quadrotor I)			Latest Quadrotor Platform (Quadrotor II)			
Variable	**Mean Error**	**Mean Absolute Error**	**Standard Deviation of Error**	**Mean Error**	**Mean Absolute Error**	**Standard Deviation of Error**	**Units**
e_{s_1}	0.1185	0.1534	0.1537	0.0541	0.0546	0.0301	-
e_{s_2}	0.0624	0.1674	0.2162	-0.0190	0.0303	0.0340	-
e_{s_3}	0.1249	0.1249	0.0395	0.5290	0.5290	0.0370	-
e_{s_4}	0.0474	0.0626	0.0612	0.0001	0.0185	0.0222	-
e_{p1}	3.78	6.04	6.81	2.87	2.91	1.73	cm
e_{p2}	-0.16	5.31	7.13	-1.52	2.46	2.75	cm
e_{p3}	-5.42	5.64	4.20	-54.45	54.45	4.26	cm
e_{ϕ}	-1.3152	1.3845	1.0448	-0.0035	0.2511	0.3178	$^\circ$
e_{θ}	-3.4388	3.4388	0.8471	0.0087	0.3150	0.4021	$^\circ$
e_{ψ}	1.6302	2.1472	3.69	0.0028	0.6399	0.7670	$^\circ$

Table 4.3: Mean and standard deviation of error for the virtual camera experimental data using the original ANCL quadrotor (2015) and the latest ANCL quadrotor (2017).

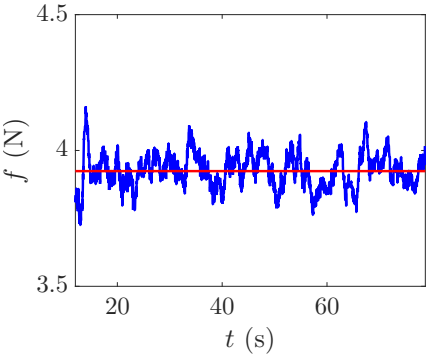


(a) Drop in battery voltage

(b) Control signal u_i



(c) Estimated thrust coefficient (red) and data (blue)



(d) Estimated force f_i (blue) and hover force $mg/4$ (red)

Figure 4.19: Time varying thrust constant.

10 cm over a period of about one minute. This is expected as the integral term in the control law (4.58a) was disabled (i.e., $k_{hi} = 0$). As the battery voltage drops, less thrust is generated for the same control signal. Although the control signal increases slightly due to the increased error, it is insufficient to compensate for the decrease in thrust coefficient. Figure 4.19a and Figure 4.19b shows battery voltage and thrust input u_i, respectively. To estimate the thrust constant in force model (2.12) we assume that each propeller generates the same thrust and denote thrust coefficient $k_f = k_{fi}, i = 1, 2, 3, 4$. We also assume the quadrotor is hovering. In this case the thrust coefficient is given by $k_f = 0.25mg/u_i^2$. We model the dependence of k_f on time as linear:

$$k_f(t) = k_{f1}t + k_{f0} \tag{4.59}$$

A least squares fit for the thrust coefficients in model (4.59) is performed. This model is shown in Figure 4.19c. Figure 4.19d shows estimated force f_i compared to the assumed hover force $mg/4$. In simulation we used (4.59) to show the integral term in (4.58a) can compensate for a decrease in thrust coefficient.

After the quadrotor platform was updated in 2017 the flight test was re-run and

the results are shown in Figure 4.18[1]. The mean of the error was similar in both of the experiments, but with the new platform the standard deviation is about an order of magnitude less. Table 4.3 shows the mean and standard deviation of the error signals.

4.2.6 Summary

In this section we investigated the control of a quadrotor using a nonlinear DIBVS to regulate the position and yaw of the vehicle relative to visual features on the ground. The control relies on two or more visual feature point which can be robustly obtained and can be defined generically depending on the application. The convergence of the translational kinematics and dynamics with the proposed control law is proven to be GAS and the yaw motion is proven to be GES. An indoor quadrotor platform was used to validate the control successfully. The results demonstrate a robustness to various unmodelled effects including non-ideal inner loop performance, mass imbalances, change in battery voltage, and relative pose between $\mathcal{V}$ and $\mathcal{B}$.

4.3 Adaptive Virtual Camera-Based Approach

In this section we present out adaptive virtual camera-based DIBVS approach that was originally publish in [56].

4.3.1 Uncertainty Modelling

From our previous experimental results in [54] we observed that lateral image feature error converged to a non-zero constant. As confirmed by our simulation and experimental results we conclude bias errors in Euler angle estimates from the attitude and heading reference system (AHRS) are the cause of this image feature error. To account for this error we model the actual roll angle $\phi = \phi_m - \phi_e$ as the difference between the measured roll ϕ_m and a constant bias error ϕ_e. Similarly the actual pitch angle $\theta = \theta_m - \theta_e$ is the difference between the measured pitch θ_m and a constant bias error θ_e. The errors ϕ_e and θ_e will be treated as additive input disturbances to the outer loop and compensated with the adaptive control. In simulation we have observed that even a small value of ϕ_e and θ_e can lead to large image error. Although we treat ϕ_e and θ_e as input disturbances in the force model, we assume the small biases have negligible influence on the kinematics of the image moments (4.52a). This assumption is practical and made to simplify the derivation of the control law. The justification for this assumption is that the difference between image coordinates in the ideal virtual camera (with zero roll and pitch) and

[1] A video of this experiment is available at `https://youtu.be/IUi_vm2Y8ak`

the real virtual camera (tilted due to small bias error) is negligible when height is large relative to lateral motion. To show small bias leads to negligible difference in image coordinates, we provide the following argument. First, we define the ideal virtual camera frame with zero roll and pitch as $\mathcal{V}^*$. Any vector X can be expressed in the real virtual camera frame $\mathcal{V}$ and $\mathcal{V}^*$ as

$$X^v = R_{\theta\phi} R_n^b X^n$$
$$X^{v*} = R_3^T X^n$$

From above two equations we obtain

$$X^v = R_2(\theta_m) R_1(\phi_m) R_1^T(\phi_m - \phi_e) R_2^T(\theta_m - \theta_e) R_3^T(\psi) X^n$$

For small ϕ_e and θ_e we have the approximation

$$X^v \approx R_1^T(-\phi_e) R_2^T(-\theta_e) X^{v*} \tag{4.60}$$

With the approximations $s_\xi \approx \xi, c_\xi \approx 1$ and (4.60), the projected image point coordinates of any 3 D point P in the virtual cameras are

$$p^v = \lambda \begin{bmatrix} \frac{p_1^{v*} - \theta_e p_3^{v*}}{p_3^{v*} + \theta_e p_1^{v*} - \phi_e p_2^{v*}} \\ \frac{p_2^{v*} + \phi_e p_3^{v*}}{p_3^{v*} + \theta_e p_1^{v*} - \phi_e p_2^{v*}} \end{bmatrix} \tag{4.61}$$

$$p^{v*} = \lambda \begin{bmatrix} \frac{p_1^{v*}}{p_3^{v*}} \\ \frac{p_2^{v*}}{p_3^{v*}} \end{bmatrix} \tag{4.62}$$

where $[p_1^{v*}, p_2^{v*}, p_3^{v*}]$ are the coordinates of P in frame $\mathcal{V}^*$. The difference of coordinates in two camera image planes is

$$p^v - p^{v*} = \lambda \begin{bmatrix} \frac{-\theta_e (p_3^{v*})^2 - \theta_e (p_1^{v*})^2 + \phi_e p_1^{v*} p_2^{v*}}{(p_3^{v*})^2 + \theta_e p_1^{v*} p_3^{v*} - \phi_e p_2^{v*} p_3^{v*}} \\ \frac{-\phi_e (p_3^{v*})^2 - \theta_e p_1^{v*} p_2^{v*} + \phi_e (p_2^{v*})^2}{(p_3^{v*})^2 + \theta_e p_1^{v*} p_3^{v*} - \phi_e p_2^{v*} p_3^{v*}} \end{bmatrix} \tag{4.63}$$

The magnitudes of ϕ_e and θ_e are far less than the magnitudes of $p_1^{v*}, p_2^{v*}, p_3^{v*}$, and the magnitudes of p_1^{v*} and p_2^{v*} are less than the magnitude of p_3^{v*} in most visual servoing applications. Hence, we have the approximate bound

$$\left\| p^v - p^{v*} \right\| \leq \sqrt{2}\lambda \left(2|\theta_e| + |\phi_e| \right) \tag{4.64}$$

Therefore, we can conclude $\|p^v - p^{v*}\|$ is negligible for small bias error ϕ_e and θ_e.

During the system identification of the UAV model we observed the thrust constant k_f slowly decreases with time as the battery voltage drops during a flight.

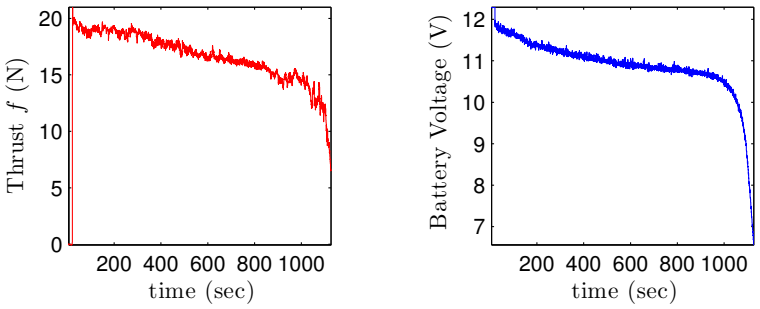


Figure 4.20: Time evolution of thrust. PWM pulse width is set to 1.4 ms.

This is shown in Figure 4.20 for one Turnigy 1100 KV brushless motor attached to an 11×4.5" propeller. The motor is powered by two 3 cell 2600 mAh LiPo batteries. To compensate for its variation we treat k_f as an unknown constant parameter. The control objective in this section is to regulate the image feature s to its desired value s^* while adapting to uncertainty in mass m, thrust constant k_f, desired depth p_3^{v*}, and the constant biases ϕ_e and θ_e that appear in force model F^v.

4.3.2 Adaptive Control

In our previous work [54] we observed that reference roll-pitch angles have to be limited to a small range to avoid the target leaving the camera's field of view (FoV). The camera used in our experiments has a 75° horizontal FoV and 47° vertical FoV. Given the small range of roll and pitch we make a small angle approximation so that the force model becomes

$$F^v = k_f u \begin{bmatrix} -s_{(\theta_m-\theta_e)}c_{(\phi_m-\phi_e)} \\ s_{(\phi_m-\phi_e)} \\ -c_{(\theta_m-\theta_e)}c_{(\phi_m-\phi_e)} \end{bmatrix} \approx k_f u \begin{bmatrix} -\theta_m+\theta_e \\ \phi_m-\phi_e \\ -1 \end{bmatrix} \tag{4.65}$$

where $u = F_3^v/k_f$. Based on (2.6), (4.52) and (4.65) we write the height subsystem as

$$\dot{s}_3 = -\frac{1}{p_3^{v*}} v_3^v \tag{4.66a}$$

$$\dot{v}_3^v = g - k_f u/m \tag{4.66b}$$

In this subsystem the control input is chosen as the thrust u and the control law is designed as

$$u = \left(K_{h2} + \hat{C}_z K_{h1}^2\right)\delta_{h2} + \hat{C}_g \tag{4.67}$$

with

$$\dot{\hat{C}}_g = K_{hg}\delta_{h2} \tag{4.68}$$

where

$$\delta_{h1} = s_3 - 1 \tag{4.69}$$

$$\delta_{h2} = \frac{v_3^v}{K_{h1}} - \delta_{h1} \tag{4.70}$$

K_{h1}, K_{h2}, K_{hg} are positive control gains, $\hat{C}_g$ is the estimate of $C_g = \frac{mg}{k_f}$, and $\hat{C}_z$ is the estimate of $C_z = \frac{m}{p_3^{v*} k_f}$. The update laws for $\hat{C}_g$ is in (4.68) and $\hat{C}_z$ is given in (4.76).

Again from (2.6), (4.52) and (4.65) the lateral subsystem dynamics is

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \end{bmatrix} = -\frac{\lambda}{Z^{v*}} \begin{bmatrix} v_1^v \\ v_1^v \end{bmatrix} + \begin{bmatrix} s_2 \\ -s_1 \end{bmatrix} \dot{\psi} \tag{4.71a}$$

$$\begin{bmatrix} \dot{v}_1^v \\ \dot{v}_2^v \end{bmatrix} = \frac{k_f u}{m} \begin{bmatrix} -\theta_m + \theta_e \\ \phi_m - \phi_e \end{bmatrix} - \begin{bmatrix} -v_2^v \\ v_1^v \end{bmatrix} \dot{\psi} \tag{4.71b}$$

In this subsystem we choose ϕ and θ as inputs and the control is taken as

$$u \begin{bmatrix} \theta_m \\ -\phi_m \end{bmatrix} = u \begin{bmatrix} \hat{\theta}_e \\ -\hat{\phi}_e \end{bmatrix} + \lambda \hat{C}_z K_{l1}^2 \delta_{l2} + K_{l2} \delta_{l2} \tag{4.72}$$

where $\hat{\theta}_m$ and $\hat{\phi}_m$ are the estimates of θ_e and ϕ_e, respectively. The update law for $\hat{\theta}_m$ and $\hat{\phi}_m$ is

$$\begin{bmatrix} \dot{\hat{\theta}}_e \\ \dot{\hat{\phi}}_e \end{bmatrix} = u K_{lrp} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \delta_{l2} \tag{4.73}$$

where K_{l1}, K_{l2}, K_{lrp} are positive gains, and

$$\delta_{l1} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \tag{4.74}$$

$$\delta_{l2} = \frac{1}{K_{l1}} \begin{bmatrix} v_1^v \\ v_2^v \end{bmatrix} - \delta_{l1} \tag{4.75}$$

The update law for $\hat{C}_z$ in (4.67) and (4.72) is

$$\dot{\hat{C}}_z = K_z \left(\lambda K_{l1} \delta_{l2}^T \delta_{l2} + K_{h1} \delta_{h2}^2 \right) \tag{4.76}$$

where $K_z > 0$.

Inner Loop	
Parameter	**Value**
$K_{p\phi}$	0.3390
$K_{p\theta}$	0.3459
$K_{p\psi}$	0.3900
$K_{i\phi}$	0.0890
$K_{i\theta}$	0.0990
$K_{i\psi}$	0.0890
$K_{d\phi}$	0.0390
$K_{d\theta}$	0.0390
$K_{d\psi}$	0.0059

Outer loop	
Parameter	**Value**
K_{h1}	0.9000
K_{h2}	0.0900
K_{hg}	0.0400
K_{l1}	0.3182
K_{l2}	0.0700
K_{lrp}	0.0200
K_{ψ}	0.5000
K_z	0.0100

Table 4.4: Control gains for adaptive virtual camera experiment.

In order to make s_4 converge to zero the yaw rate reference is set as

$$\psi = K_\psi \int_0^t s_4(\xi) d\xi \tag{4.77}$$

where $K_\psi > 0$.

Theorem 4.3. *The equilibrium* $[\delta_{h1}, \delta_{h2}, \delta_{l1}, \delta_{l2}, s_4]^T = 0$ *of the closed-loop system* (4.66), (4.71), (4.52b) *with control law* (4.67), (4.72), (4.77) *and parameter update laws* (4.68), (4.73), (4.76) *is GAS.*

See [56] for a proof.

4.3.3 Simulation Results

The visual servoing objective is to make the vehicle hover at a constant position and yaw by regulating the image feature error. We consider two cases in simulation. The first case is the proposed control law with parameter update laws (4.68) and (4.73) turned off. This case extends the state transformation results in [54] and serves as a reference for comparison to the adaptive case. In the second case we simulate the complete proposed adaptive control law. We use the model parameters in Table 3.1, take thrust coefficient $k_f = 40$ N/ms^2, and desired depth $p_3^{v*} = 1.335$ m. The initial displacement of the vehicle in the camera frame is $[0.15, -0.15, -1]^T$ m with zero roll and pitch angles, and the initial value of yaw is $\pi/3$ rad. The initial values of v^v and ω^c are zero. Both ϕ_e and θ_e are set to 2°. The estimates of C_g, ϕ_e, θ_e used in the control law are 0.38, 0, 0, respectively. We use two target points located at coordinates $[-0.18, 0, 0]^T$ and $[0.18, 0, 0]^T$ in $\mathcal{N}$. The control gains are given in Table 4.4. The image feature error is denoted

$$e_s = [e_{s1}, e_{s2}, e_{s3}, e_{s3}]^T = s - s^*$$

The desired 3 D translational displacement of the UAV is denoted p^{n*} and the translational error is

$$e_p = [e_{p1}, e_{p2}, e_{p3}]^T = p^n - p^{n*}$$

where $p^{n*} = [0.0477, -0.0468, -1.335]^T$ m. This value of p^{n*} corresponds to $e_s = 0$. The desired yaw of the UAV is ψ^* and yaw tracking error is

$$e_\psi = \psi - \psi^*$$

The trajectories of e_s, e_p, and e_ψ are shown in Figure 4.21. From this figure we observe that image features e_s converges to $[-0.4991, 0.4989, -0.1378]^T$ when the adaptive law is off. The vehicle position error is $e_p = [0.2310, -0.2266, 0.2083]^T$ m. This significant 3 D position error is due to the additive input disturbance ϕ_e, θ_e, and the inaccurate estimate of k_f/m. Next we turn on the parameter update laws to ensure e_s and e_p converge to zero.

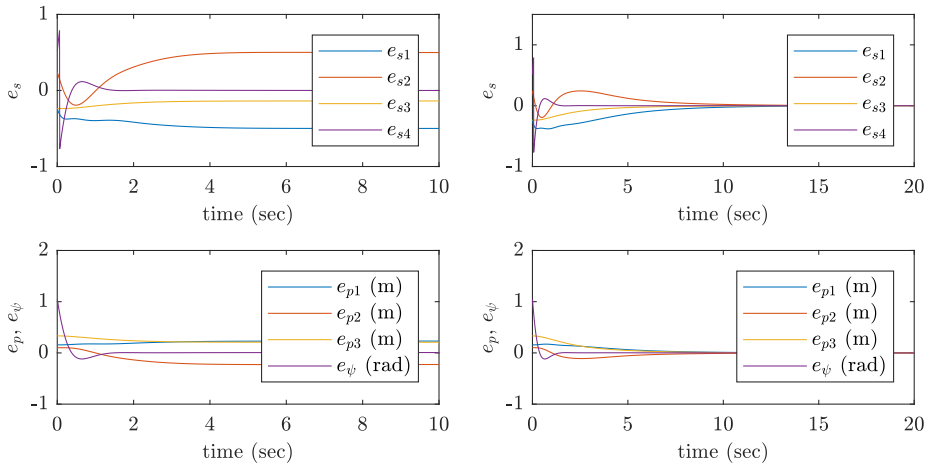
The estimates $\hat{\phi}_e$, $\hat{\theta}_e$, $\hat{C}_g$ and $\hat{C}_z$ are shown in Figure 4.22. It can be seen that both $\hat{\phi}_e$ and $\hat{\theta}_e$ converges to their actual value $2^\circ = 0.0349$ rad. The estimate $\hat{C}_g$ also converges to 0.3924 which corresponds to the amount of thrust needed to compensate gravity. The value of $\hat{C}_z$ converges to 0.0075 kg·ms^2/m·N. Since there is no uncertainty in the yaw kinematics, in both simulation cases e_{s4} and e_ψ converge to zero. Hence, the simulations show that the proposed adaptive control compensates for constant additive input disturbances and uncertainty in k_f and p_3^{v*}.

4.3.4 Experimental Results

In practice the state measurements contain noise which would lead to divergence of the adaptive law in (4.76). A typical solution to this problem is to use the projection algorithm as in [114]. This requires a range of C_z as prior knowledge. However, we observe that if $K_{l2} > \lambda C_z K_{l1}^2, K_{h2} > C_z K_{h1}^2$, and the term $\hat{C}_z$ in (4.67), (4.72) is deleted, the closed loop remains asymptotically stable. Since both solutions assume a known range for C_z, we choose the method which eliminates $\hat{C}_z$ and tune the value of K_{h2} and K_{h1}. This leads to a simpler implementation. Hence, the control law (4.67) and parameter update (4.68) are implemented as

$$u = K_{h2}\delta_{h2} + K_{hg}\int_0^t \delta_{h2}(\xi)d\xi \tag{4.78}$$

Next, we note that when the vehicle is hovering u is approximately constant to compensate gravity. Thus, we assume u in (4.72) is constant, and the control (4.72)



(a) Non-adaptive case (b) Adaptive case

Figure 4.21: Trajectories of image feature error e_s, vehicle's 3 D position error e_p, and e_ψ.

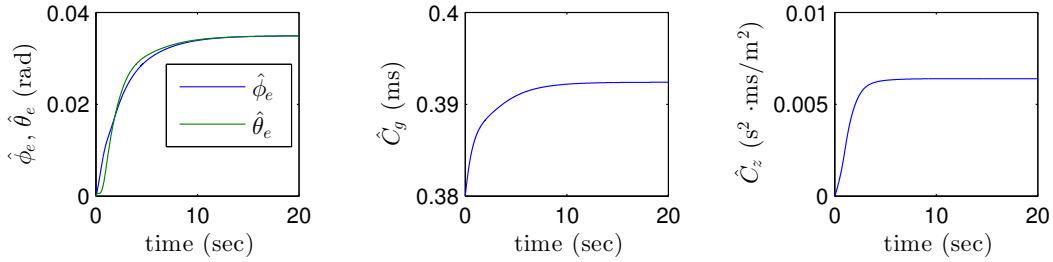


Figure 4.22: Trajectories of estimated parameters $\hat{\phi}_e$, $\hat{\theta}_e$, $\hat{C}_g$, and $\hat{C}_z$.

and parameter update law (4.73) are implemented as

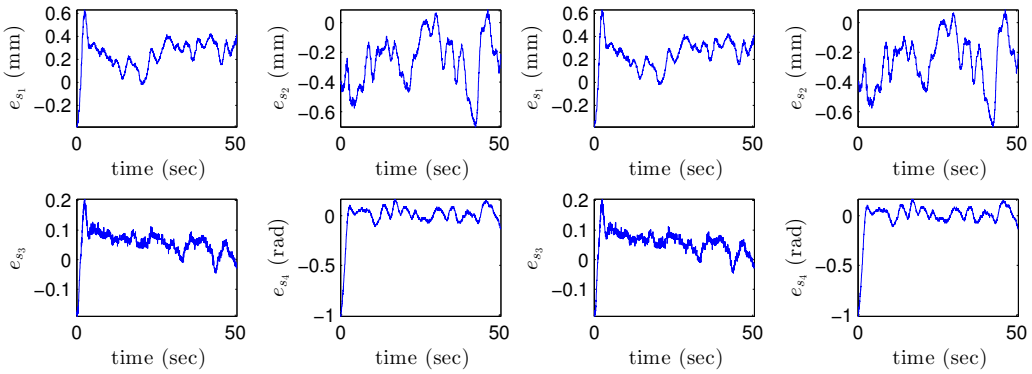
$$\begin{bmatrix} \theta_m \\ -\phi_m \end{bmatrix} = K_{l2}\delta_{l2} + K_{lrp}\int_0^t \delta_{l2}(\xi)d\xi \tag{4.79}$$

The values of the control gains in (4.78), (4.79), and (4.77) are the same as in the simulation and are given in Table 4.4. The inner loop control gains are also given in Table 4.4.

As in the simulation we consider two cases in experiment, i.e., with the adaptive law (4.68) and (4.73) on and off. In the adaptive case, the initial values of $\hat{\phi}_e$, $\hat{\theta}_e$ are set to zero. The estimate value of C_g in the non-adaptive case is 0.435, which is obtained from a manual flight test, and this value is the initial value of $\hat{C}_g$ in the adaptive case. The trajectory of the image feature error e_s for the non-adaptive case and the adaptive case is given in Figure 4.23a and Figure 4.23b, respectively. The trajectories of two image points are shown in Figure 4.24, where the initial coordinates are shown as a square and the circle denotes the final point. Figure 4.25a and Figure 4.25b show the corresponding 3 D translational error e_p and e_ψ. Typical inner-loop tracking performance is shown in Figure 4.26. Figure 4.27 illustrates the estimates $\hat{\phi}_e$, $\hat{\theta}_e$, and $\hat{C}_g$. Based on the above mentioned plots, both cases reach steady state in about 15 seconds. Table 4.5 and Table 4.5 give the mean and standard deviation of e_s, e_p and e_ψ after 15 seconds. It can be seen from Figure 4.23 that the e_{s3} in the non-adaptive case is slowly reducing because the voltage of battery is slowly dropping while in the adaptive case the e_{s3} stays around 0. We also observe from Table 4.5 that in the adaptive case the mean value of e_{s1} and e_{s2} are significantly reduced with similar standard deviations. The trajectories of e_{s4} converge to zero for both cases, and the performance for regulating the yaw motion in the two cases are similar. This is to be expected since the same control law (4.77) is used. The trajectories in the image plane shown in Figure 4.24 are consistent with Table 4.5. That is, the lower mean values of e_{s1}, e_{s2} correspond to a more centred steady state image coordinates. Figure 4.25 and Table 4.5 show the 3 D position errors in the adaptive case are much smaller than the non-adaptive case. Hence, we conclude that the proposed control provides improved visual servoing motion control.

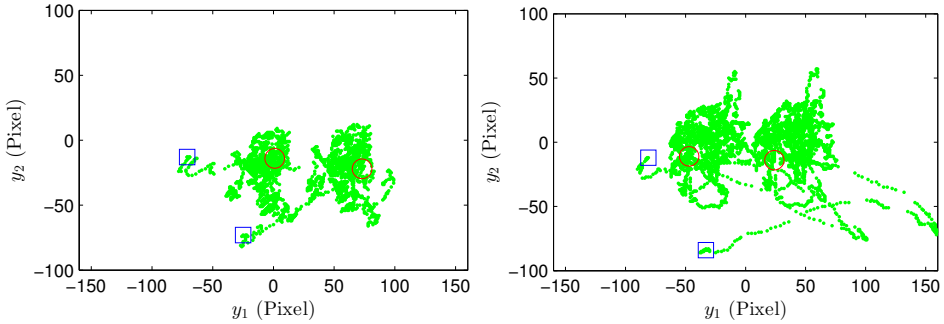
4.3.5 Summary

We have presented an adaptive DIBVS control which uses image moment features from projected points in a virtual camera image. The image moment features in the virtual camera image lead to a simple interaction matrix. The proof of global asymptotic stability of the error dynamics is given for the case of target points in



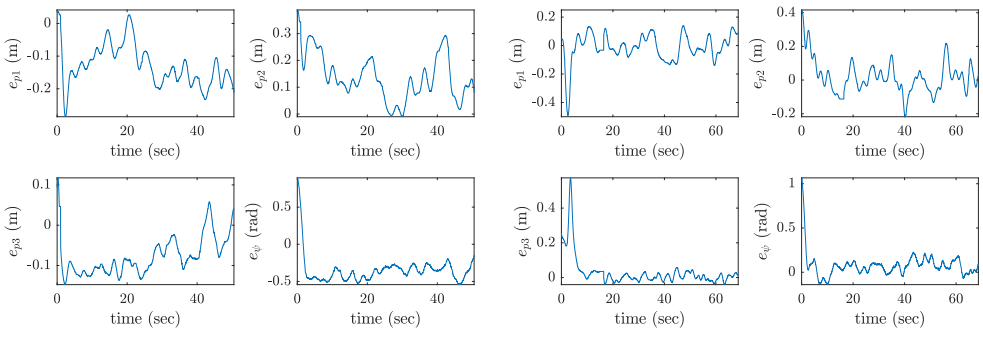
(a) Non-adaptive case (b) Adaptive case

Figure 4.23: Experimental trajectories of image feature error e_s.



(a) Non-adaptive case (b) Adaptive case

Figure 4.24: Images points trajectories. The starting position is denoted with a square and the final position is denoted with a circle.



(a) Non-adaptive case (b) Adaptive case

Figure 4.25: Experimental error trajectories of vehicle's 3 D position error e_p and e_ψ.

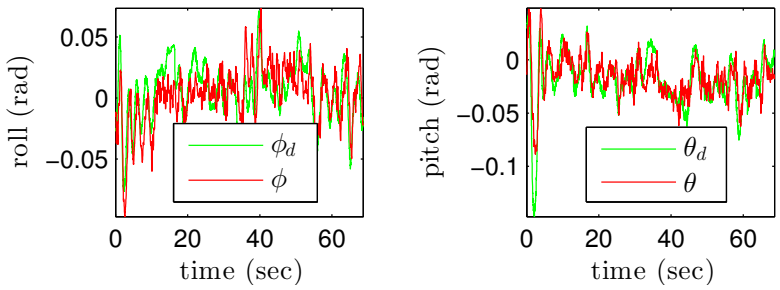


Figure 4.26: Roll and pitch tracking performance during adaptive virtual camera experiment.

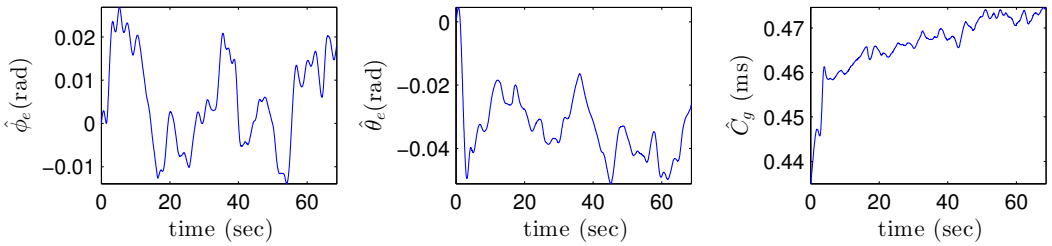


Figure 4.27: Experimental trajectories of estimated parameters $\hat{\phi}_e$, $\hat{\theta}_e$, and $\hat{C}_g$.

	Adaptive control		**Non-adaptive control**	
Image Feature	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**
e_{s1}	0.0041	0.1291	0.2456	0.1042
e_{s2}	-0.0067	0.1847	-0.2250	0.1694
e_{s3}	-0.0047	0.0196	0.0517	0.0297
e_{s4}	0.0171	0.0652	0.0192	0.0602
e_{p1}	-0.0077	0.0718	-0.1276	0.0586
e_{p2}	-0.0015	0.0770	0.1158	0.0696
e_{p3}	0.0035	0.0214	-0.0740	0.0437
e_{ψ}	0.0703	0.0613	-0.1083	0.0824

Table 4.5: Image feature error for adaptive virtual camera experiment.

a horizontal plane. The proposed control does not require an estimate of depth, known mass, and thrust coefficient. The method is robust to measurement bias in roll and pitch. An indoor quadrotor platform is used to validate the control. The results demonstrates improved performance with lower mean image and position error. We explain how the proposed control can be extended to the non-horizontal target case for motion control in the translational degrees of freedom.

4.4 Extensions to Virtual Camera-Based Approach

In this section we present two extensions for the virtual camera-based DIBVS approach that were originally publish in [57].

4.4.1 Moving Targets

Similar to Section 4.2.3, the control objective in this section is to develop a control law that regulates the relative position and yaw of the vehicle to a target. However, in this section the target is now free to move. Once again to simplify the presentation we make the small angle assumption, the diagonal inertial matrix assumption, and the assumption that camera frame $\mathcal{C}$ and the body frame $\mathcal{B}$ are identical. The dynamics of the quadrotor and the target in the virtual camera frame $\mathcal{V}$ is

$$\dot{\tilde{v}}^v = -\dot{\tilde{\psi}}\,\mathrm{sk}(e_3)\tilde{v}^v + ge_3 + F^v/m - \bar{a}_t^v \tag{4.80}$$

where $\bar{a}_t^v$ is the acceleration of the target. We assume knowledge of $\dot{\psi}_t$ and $\bar{v}_t^v$ (e.g., transmitted from a friendly vehicle or measured from optical flow). To derive a control law we write the dynamics (4.80) with the image moment kinematics (4.52) as

$$\dot{s}_1 = -\lambda k_p \tilde{v}_1^v + s_2\dot{\psi} \tag{4.81a}$$
$$\dot{s}_2 = -\lambda k_p \tilde{v}_2^v - s_1\dot{\psi} \tag{4.81b}$$
$$\dot{s}_3 = -k_p \tilde{v}_3^v \tag{4.81c}$$
$$\dot{\tilde{v}}_1^v = \tilde{v}_2^v\dot{\psi} + u_\theta \tag{4.81d}$$
$$\dot{\tilde{v}}_2^v = -\tilde{v}_1^v\dot{\psi} + u_\phi \tag{4.81e}$$
$$\dot{\tilde{v}}_3^v = u_F \tag{4.81f}$$
$$\dot{s}_4 = u_\psi \tag{4.81g}$$

where $\delta_3 = s_3 - 1$, $k_p = 1/p_3^{v*}$, and the inputs are $u_\theta = -F_3^v\theta/m - \tilde{v}_2\dot{\psi}_t - \bar{a}_{t1}^v$, $u_\phi = F_3^v\phi/m + \tilde{v}_1\dot{\psi}_t - \bar{a}_{t2}^v$, $u_F = -F_3^v/m + g - \bar{a}_{t3}^v$, and $u_\psi = -\dot{\tilde{\psi}}$.

Theorem 4.4. *The subsystem* (4.81a) *to* (4.81f) *is GAS and the subsystem* (4.81g) *is GES with the input*

$$u_F = k_{hd} v_3^v - k_{hp}\delta_3 - k_{hi}\int_0^t \delta_3(\epsilon)\, d\epsilon \tag{4.82a}$$

$$u_\theta = -k_{\ell 2}(\tilde{v}_1^v/k_{\ell 1} - s_1) \tag{4.82b}$$

$$u_\phi = -k_{\ell 2}(\tilde{v}_2^v/k_{\ell 1} - s_2) \tag{4.82c}$$

$$u_\psi = k_\psi s_4 \tag{4.82d}$$

where k_{hp}*,* k_{hd}*,* k_{hi}*,* $k_{\ell 1}$*,* $k_{\ell 2}$ *and* k_ψ *are positive constant gains such that* $k_{hi} < k_{hp}k_{hd}$ *and* $P_3^{v*}k_{\ell 2} > k_{\ell 1}^2$*.*

Proof. First we consider the height motion subsystem (4.81c) and (4.81f) and we define the following error signal

$$\delta_3 = s_3 - 1$$

In addition we augment the state by an integral term

$$\zeta = \int_0^t \delta_3(\epsilon)\ \mathrm{d}\epsilon$$

Let $u_F = k_{hp}\delta_3 - k_{hd}\dot{\tilde{v}}_3^v + k_{hi}\zeta$ then the closed loop dynamics are

$$\begin{bmatrix} \dot{\delta}_3 \\ \dot{v}_3^v \\ \dot{\zeta} \end{bmatrix} = \begin{bmatrix} 0 & -k_p & 0 \\ k_{hp} & -k_{hd} & k_{hi} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_3 \\ \tilde{v}_3^v \\ \zeta \end{bmatrix}$$

Then if $k_{hp}, k_{hd}, k_{hi} > 0$ and $k_{hi} < k_p k_{hp} k_{hd}$ the closed loop origin of the height motion subsystem is GES. Next, we consider the translational motion subsystem (4.81a), (4.81b), (4.81c), and (4.81d) and define error signals

$$\begin{aligned} \delta_1 &= \frac{1}{\lambda} s_1 \\ \delta_2 &= \frac{1}{\lambda} s_2 \\ \delta_4 &= \tilde{v}_1^v/k_{\ell 1} - \delta_1 \\ \delta_5 &= \tilde{v}_2^v/k_{\ell 1} - \delta_2 \end{aligned}$$

and consider the Lyapunov function candidate

$$V = \frac{1}{2}\delta_1^2 + \frac{1}{2}\delta_2^2 + \frac{1}{2}\delta_4^2 + \frac{1}{2}\delta_5^2$$

then

$$\dot{V} = (u_\theta \delta_4 + u_\phi \delta_5)/k_{\ell 1} + k_{\ell 1} k_p \left(-\delta_1^2 - \delta_2^2 + \delta_4^2 + \delta_5^2 \right)$$

Let $u_\theta = -k_{\ell 2}\delta_4$ and $u_\phi = -k_{\ell 2}\delta_5$ then

$$\dot{V} = -k_{\ell 1} k_p (\delta_1^2 + \delta_2^2) - (k_{\ell 2} - k_p k_{\ell 1}^2)(\delta_4^2 + \delta_5^2)/k_{\ell 1}$$

If $k_{\ell 2} > k_{\ell 1}^2 k_p$ then $\dot{V}_2$ is negative definite. Therefore the translational subsystem (4.81a) to (4.81f) is GAS. For the yawsubsystem (4.81g) we substitute the control law (4.82d) into the dynamics (4.52b) and obtain

$$\dot{s}_4 = -k_\psi s_4$$

which is GES. □

The control law requires an initialization and main phase. In the initialization phase the quadrotor is manually hovered so that the target is in the camera's field of view. The user then selects the target, indicates its orientation in the camera frame and the desired distance to the target. Note that the desired distance to the target is not specified in metres, but is an image feature. That is, the user selects the desired scale in the image: larger (closer), the same (maintain distance), or smaller (further away). This feature can be changed at anytime during flight.

4.4.2 Non-horizontal Targets - Applying the Homography

In previous work [47, 55] it was assumed that the planar target was horizontal. This assumption was made so that the concept of a virtual camera as used in [55] could be applied. If the virtual camera is applied to non-horizontal targets it has been shown that steady state error results in the image features, especially in the third component s_3 which is related to relative height. In this section we compensate for non-horizontal target planes using a homography decomposition which can be used to estimate the orientation between the camera image plane and target plane.

After computing the homography decomposition we use it to define a new virtual camera frame $\mathcal{V}'$. The following discussion explains how this new frame compensates for non-horizontal targets. The following assumptions are made to provide a precise description of the pose of $\mathcal{V}'$ without having to know the geometry of the target and the relative pose from the camera to the target.

Assumptions:

1. The target rotates about its geometric centre.

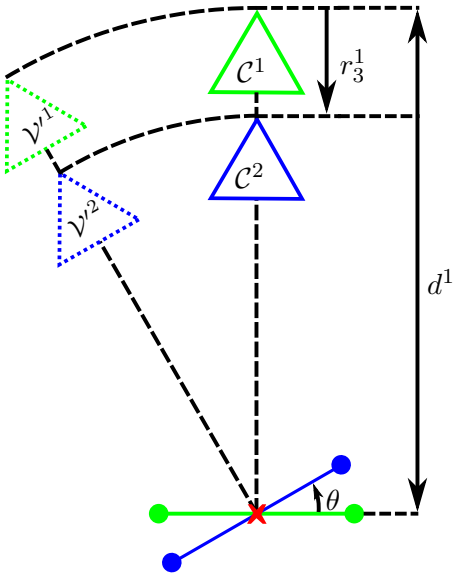


Figure 4.28: An initial desired image is taken from $\mathcal{C}^1$ of the horizontal target. The virtual camera associated with $\mathcal{C}^1$ and the rotated target is $\mathcal{V}'^1$. The virtual camera associated with $\mathcal{C}^2$ and the rotated target is $\mathcal{V}'^2$.

2. The target plane in the desired image is parallel to the camera plane, i.e., $n^1 = [0, 0, 1]^T$.

3. The camera is centred laterally above the target, i.e., the geometric centre of the target lies on the c_3 axis.

The frame $\mathcal{V}'$ has the same lateral position and orientation relative to the target as the camera to the desired scene. That is, only the height of the camera can differ relative to the desired case. Using $\mathcal{V}'$ we can estimate the area of the target independent of the target and camera's orientation. We remark that virtual camera $\mathcal{V}$ introduced in Section 4.2 remains parallel to the ground. Using a combination of $\mathcal{V}$ and $\mathcal{V}'$ we can define image features that behave as if the target is always parallel to the ground.

Figure 4.28 illustrates the idea when relative motion is restricted to one DoF in the c_3 direction, and the target plane can rotate by an angle θ. The 3 D case is restricted to relative camera motion in the c_3 direction and the target can rotate in two rotational DoF. The extension of the 2 D case to the 3 D case is straightforward. In the figure an initial desired image Y^1 is taken from the camera $\mathcal{C}^1$ of the target in its desired position (green). Next, suppose the target rotates about its geometric centre (denoted by a red "x") by an angle θ. At this point if another image were taken with $\mathcal{C}^1$ then the length (area in 3 D) of the target in the image will have decreased. Given this condition, the control law would attempt to lower the camera to increase the length of the target. However, the control objective is to regulate

the height above the target independent of the target's orientation. To solve this problem a homography matrix H is estimated between the desired image (camera $\mathcal{C}^1$ with a horizontal target) and the current image (camera $\mathcal{C}^1$ with a non-horizontal target). Next, the homography is decomposed into the rotation and scaled translation of the camera. Since the decomposition and estimation of the homography assumes the target is static and only the camera moves, we can effectively rotate the camera about the target's geometric centre by an angle θ to ${\mathcal{V}'}^1$. An image taken from virtual camera ${\mathcal{V}'}^1$ will show the same target length as the desired image and thus the control law will not command the camera to move. If at some point in time the camera were to move to $\mathcal{C}^2$ and take an image of the non-horizontal target, the length of the target in the vehicle could be exactly the same as its length in the desired image. A control law based on the original virtual camera $\mathcal{V}$ would incorrectly not command the camera to move upwards. A homography is calculated between the image taken in $\mathcal{C}^1$ of the horizontal target and the image in $\mathcal{C}^2$ of the non-horizontal target. This homography is decomposed to define the virtual camera ${\mathcal{V}'}^2$. The length of the target in ${\mathcal{V}'}^2$ will be less than in the desired image and the camera will be commanded to rise.

The image $Y' \in \mathbb{R}^3$ in the virtual camera frame $\mathcal{V}'$ is approximately given by

$$Y' \sim A \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1-(\frac{1}{d^1} r_3^1)^2 \end{bmatrix} A^{-1} Y^* \tag{4.83}$$

where $Y^* \in \mathbb{R}^3$ denotes the image in the desired camera frame. The scaled displacement $\frac{1}{d^1} r_3^1$ is obtained from the homography decomposition as explained in Section 2.2.4. Relation (4.83) is equivalent to

$$y^{v'} = \frac{1}{1-(\frac{1}{d^1} r_3^1)^2} y^* \tag{4.84}$$

where $y^{v'} \in \mathbb{R}^2$ denotes the image in $\mathcal{V}'$, and $y^* \in \mathbb{R}^2$ is the desired image.

The feature is given by

$$s_3' = \sqrt{\frac{\mu_{20}'^* + \mu_{02}'^*}{\mu_{20}' + \mu_{02}'}} \tag{4.85}$$

which simplifies to

$$s_3' = \sqrt{\frac{\sum_{k=1}^K ({y_{1k}^{v'}}^*)^2 + ({y_{2k}^{v'}}^*)^2}{\sum_{k=1}^K (y_{1k}^{v'})^2 + (y_{2k}^{v'})^2}}$$

Parameter	Value
$[k_{hp}, k_{hd}, k_{hi}]$	$[4.4, 1.0, 1.0]$
$[k_{\ell 1}, k_{\ell 2}]$	$[1.2, 0.4]$
k_ψ	0.60
α	0.5
$[k_{p\phi}, k_{i\phi}, k_{d\phi}]$	$[10, 1, 3]$
$[k_{p\theta}, k_{i\theta}, k_{d\theta}]$	$[10, 1, 3]$
$[k_{p\psi}, k_{i\psi}, k_{d\psi}]$	$[6, 1, 3]$
k_f	$18 - 0.02t$ N
$p^n(0)$	$[0.25, 0.25, -1.0]$ m
p^{n*}	$[0, 0, -1.5]$ m
$\psi(0)$	45°
$\psi_t(t)$	0°

Table 4.6: Simulation parameters for the virtual camera experiments with moving and non-horizontal targets.

since $y_{1g}^{v'} = y_{2g}^{v'} = 0$ by assumption. The kinematics of the feature is

$$\dot{s}_3' = \frac{-v_3^{v'}}{P_3^{v'*}} \tag{4.86}$$

Both virtual camera frames $\mathcal{V}$ and $\mathcal{V}'$ rely on assumptions. That is, $\mathcal{V}$ assumes that the target is horizontal, but makes no assumptions about the pose of the camera. On the other hand $\mathcal{V}'$ allows for non-horizontal targets, but assumes the camera is constrained to move vertically above the target. In order to exploit the benefits of both virtual frames, we combine the area moment features into one feature $\tilde{s}_3$. The resulting image feature vector $\tilde{s}$ is defined as

$$\begin{aligned} \tilde{s}_1 &= s_3' s_1 / s_3 \\ \tilde{s}_2 &= s_3' s_2 / s_3 \\ \tilde{s}_3 &= \alpha s_3 + (1-\alpha) s_3' \\ \tilde{s}_4 &= s_4 \end{aligned} \tag{4.87}$$

where $\alpha \in [0, 1]$ is a tuning parameter. In practice we have found that an equal weighting (i.e., $\alpha = 1/2$) of two features to yield good results. The algorithm is summarized in Algorithm 1.

4.4.3 Simulation Results

In the non-horizontal target case complexity of the control law increases given the homography decomposition is computed. At the time of publication of [54] these computations could not be implemented on the ANCL quadrotor platform. Hence, a

Algorithm 1 Dynamic Visual Servoing

function MAIN LOOP
    Takeoff
    Hover
    while User wishes to track object **do**
        $\{y^1, n^1, a^*\} \leftarrow$ Ini()
        Track($\{y^1, n^1, a^*\}$)
    end while
    Land
end function

Initialization Phase:

function INI
    $y^1 \leftarrow$ User select target in image
    $n^1 \leftarrow$ User select orientation of target in image
    $a^* \leftarrow$ User selects desired distance of target
end function

Main Phase:

function TRACK($\{y^1, n^1, a^*\}$)
    $y \leftarrow$ Take Image ▷ (2.33)
    $y^2 \leftarrow$ Track points in image(y)
    $H \leftarrow$ Estimate Homography(y^1,y^2)
    $R, \frac{1}{d^1} r^1 \leftarrow$ Decompose Homography(H,n^1)
    $R_{\theta\phi} \leftarrow$ Get attitude from AHRS
    $s \leftarrow$ Calc Image Features(y^2,$R_{\theta\phi}$) ▷ (4.51)
    $s' \leftarrow$ Estimate target area(y^1,$\frac{1}{d} r^1$) ▷ (4.83)
    $\tilde{s} \leftarrow$ Update Image Features(s,s') ▷ (4.87)
    F^b,$\eta^* \leftarrow$ Outer Loop($\tilde{s}$,v^v,v^{v*},$\dot{\psi}^*$) ▷ (4.82)
    $\tau \leftarrow$ Inner Loop(η,η^*,ω) ▷ (2.19)
    Repeat
end function

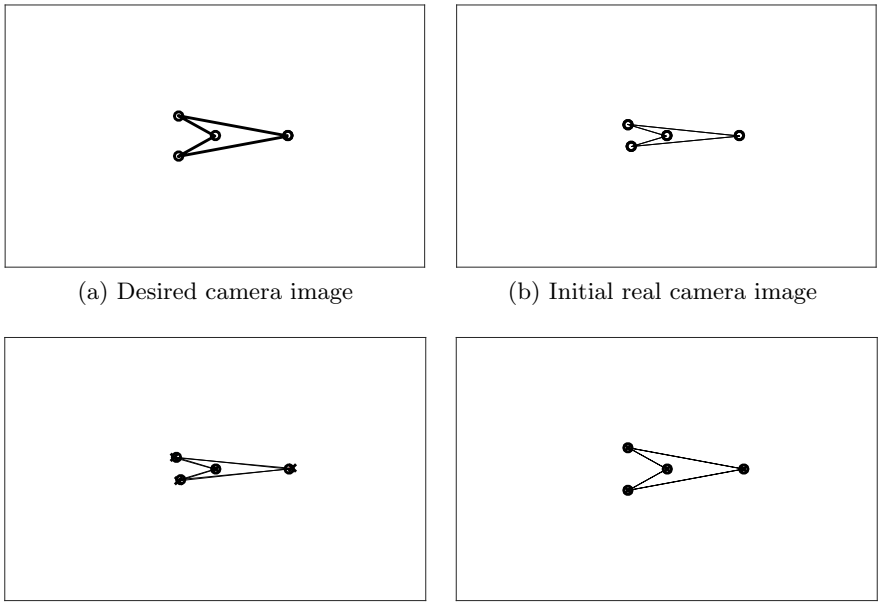
Simulink simulation was used for validation. The simulations include the rigid body dynamics and kinematics (2.6), the pinhole camera (2.33), the virtual cameras (4.55) and (4.84), the image features (4.51) and their modifications (4.87). The algorithm used is described in Algorithm 1. We chose the visual target to be a "car" which is represented by four points in the shape of an arrow that points in the forward direction of the car. There is no requirement to know additional information about the target's geometry such as its dimensions. We simulate a control assuming the target acceleration is zero. This avoids the requirement of measuring this quantity and makes the control more practical.

In the first simulation the quadrotor was initialized directly above the stationary car that was rotated by one radian (57°) about its roll axis. The control objective was to regulate height above the car and the proposed method is compared with that previously developed in [55]. The camera images for the simulation are given in Figure 4.29. The motion is such that initial (shown as circle) and final images (shown as cross) are almost identical. We observe that in both the initial real camera image (Figure 4.29b) and the trajectory of the virtual camera image for $\mathcal{V}$ (Figure 4.29c) the car appears compressed in the vertical axis of the image due to the inclined target. This leads to a reduced area which is measured by moment feature s_3 and thus the quadrotor flies closer to car to increase the target's area in the image. Both the desired image (Figure 4.29a) and the image in the new virtual camera $\mathcal{V}'$ are identical in this case.

Figure 4.30 shows height trajectories for a number of cases. Figure 4.30a compares height using the proposed method (green) and the previous approach (blue). The desired height is 1.5 m and the proposed algorithm provides a smaller steady state error relative to the previous work. Figure 4.30b shows the effect of time varying thrust coefficient when an integrator is not used in the control ($k_{hi} = 0$). In this case height for both of the algorithms drops significantly. Figure 4.30c shows the result with the integrator enabled ($k_{hi} = 1$). Figure 4.31 shows the result of the homography decomposition with roll and pitch estimated to less then one degree of error.

In the second simulation we define a surface on which the car travels:

$$z(x,y) = c_1 e^{\frac{-(x-x_{01})^2}{c_{x1}} + \frac{-(y(t)-y01)^2}{c_{y1}}} + c_2 e^{\frac{-(x-x_{02})^2}{c_{x2}} + \frac{-(y-y_{02})^2}{c_{y2}}}$$



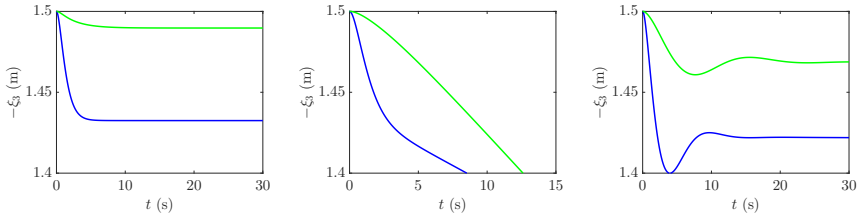
(a) Desired camera image

(b) Initial real camera image

(c) Virtual camera $\mathcal{V}$ image

(d) New virtual camera $\mathcal{V}'$ image

Figure 4.29: Simulated camera images using the proposed method and the approach in [55]. The control objective is to regulate height above the target.

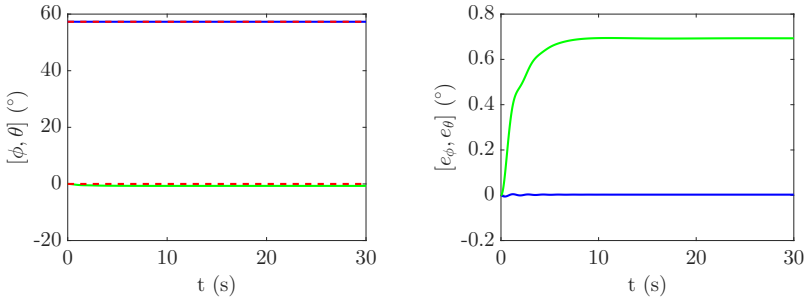


(a) Ideal battery

(b) Real battery without integrator compensation

(c) Real battery with integrator compensation

Figure 4.30: Simulated height of the camera using the proposed method (green) and the previous approach in [55] (blue).



(a) Roll and pitch estimates (solid red) and actual values (dashed red)

(b) Error in estimation

Figure 4.31: Roll (blue) and pitch (green) estimation using a homography decomposition.

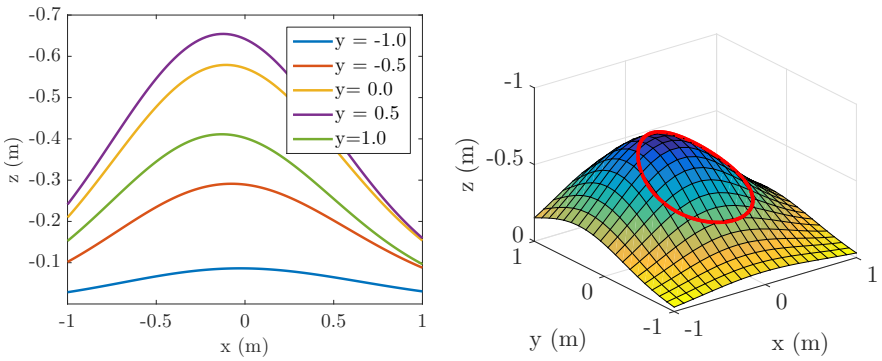


Figure 4.32: The ground plane and the trajectory of the car (red).

The car's path $\bar{P}^n(t)$ is defined by

$$\bar{P}_1^n(t) = \begin{cases} \frac{rt}{t_0} & 0 \leq t < t_0 \\ r\cos(t/c_t - t_0) & t \geq t_0 \end{cases}$$

$$\bar{P}_2^n(t) = \begin{cases} 0 & 0 \leq t < t_0 \\ r\sin(t/c_t - t_0) & t \geq t_0 \end{cases}$$

$$\bar{P}_3^n(t) = z(\bar{P}_1^n(t), \bar{P}_2^n(t))$$

The car's attitude is such that it is level to the tangent of the surface z. Its yaw angle is zero until after the car completes one full circle then it follows a sine wave. The attitude of the car is given by

$$\phi(x, y, t) = 2\left(\frac{x(t) - x_{01}}{c_{x1}} + \frac{x(t) - x_{02}}{c_{x2}}\right) z(x(t), y(t))$$

$$\theta(x, y, t) = -2\left(\frac{y(t) - y_{01}}{c_{y1}} + \frac{y(t) - y_{02}}{c_{y2}}\right) z(x(t), y(t))$$

$$\psi(t) = \begin{cases} 0 & 0 \leq t < t_1 \\ \sin(t/c_{t\psi} - t_1) & t \geq t_1 \end{cases}$$

The surface z and the car's trajectory $[c_x, c_y, c_z]^T$ (red) is shown in Figure 4.32. The constants used are in Table 4.7.

We ran the car following simulation using the previous algorithm where the velocity of the target is assumed to be zero and the target is assumed horizontal [55] and the proposed algorithm where the velocity of the target is no longer assumed to be zero and the target can be non-horizontal. The result of the simulations are shown in Figures 4.33 and 4.35.

Parameter	Value
c_1	-0.5
c_2	-0.2
(x_{01}, y_{01})	$(-0.2, 0.4)$
(y_{02}, y_{02})	$(0.2, 0.2)$
c_{x1}	0.7
c_{y1}	0.8
c_{x2}	1
c_{y3}	1
c_t	5
$c_{t\psi}$	$\frac{2\pi}{5}$
t_0	3
t_1	$2\pi c_t + t0$
r	0.5

Table 4.7: Simulation parameters used to define ground plane and car trajectory.

Figure 4.33 shows relatively poor performance of the previous method. From Figure 4.33a the point features of the car in the image exhibit large variation, and this causes large error in the moment image features as shown in Figure 4.33c. Figure 4.33f shows loss of yaw tracking for $t \geq t_1$. Although in Figure 4.33d the vehicle shows bounded lateral position tracking error, the magnitudes of error are large. The non-zero tracking error is from assuming the target accelerations $\bar{a}_t^v$ is zero.

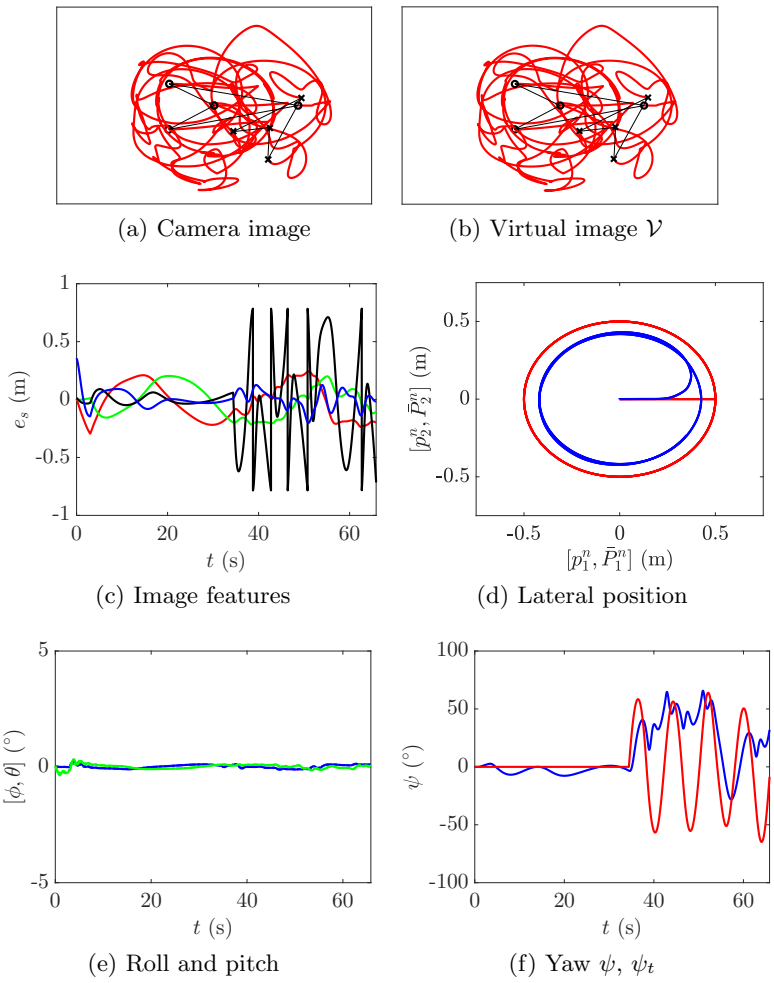
Figure 4.35 shows increased tracking performance using the proposed method. We observe from Figure 4.35a that the point features of the car exhibit small variation, and image feature error is reduced as shown in Figure 4.35d. Figures 4.35e and 4.35g show good tracking performance for lateral position and yaw. Figure 4.34a shows the roll and pitch computed from the homography decomposition versus actual roll and pitch. Figure 4.34b gives the error trajectories for these angles, and we observe magnitudes less than about 5°. Another measure of error for a homography matrix is the backward and forward projection error:

$$e_f = HY_1 - Y_2, \quad e_b = H^{-1}Y_2 - Y_1$$

These errors are less then one pixel over the entire simulation.

4.4.4 Summary

In this section we investigated a nonlinear DIBVS for a quadrotor which regulates the relative position and yaw to a moving target. The control relies on four or more visual feature points in a plane. These point features are a relatively easy to track, and the target geometry can be generically defined and is not assumed

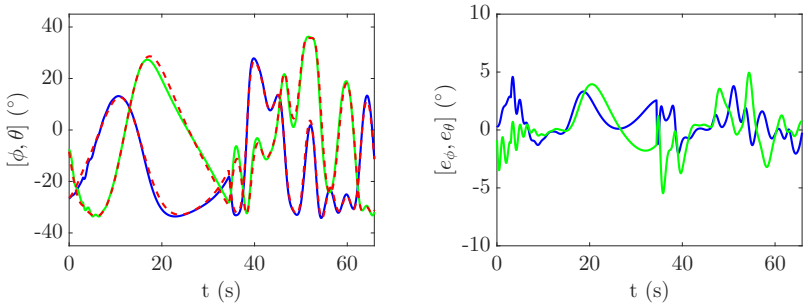


(a) Camera image (b) Virtual image $\mathcal{V}$

(c) Image features (d) Lateral position

(e) Roll and pitch (f) Yaw ψ, ψ_t

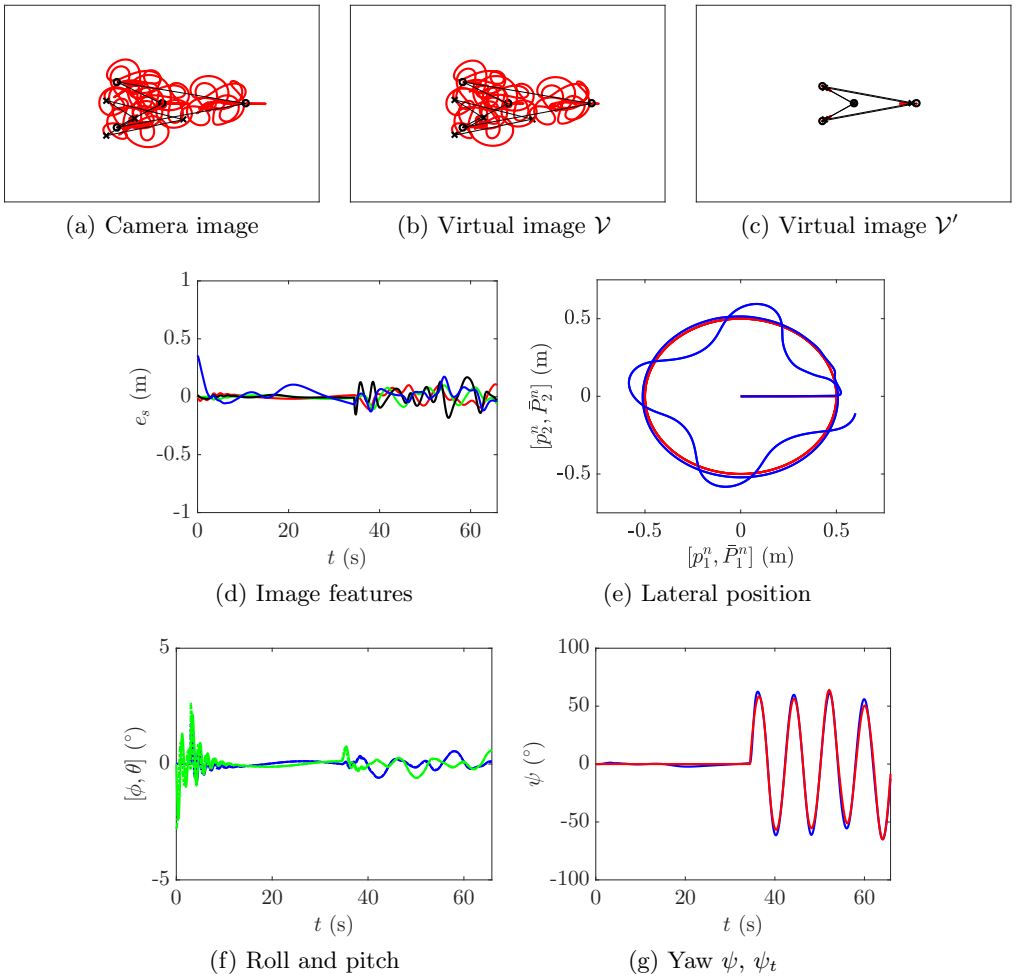
Figure 4.33: Experiment using a dynamic IBVS based on the virtual camera $\mathcal{V}$. The quadrotor (blue) follows a car (red).



(a) Roll and pitch estimates (solid) and actual values (dashed red)

(b) Error in estimation

Figure 4.34: Roll (blue) and pitch (green) estimation using a homography decomposition.



(a) Camera image (b) Virtual image $\mathcal{V}$ (c) Virtual image $\mathcal{V}'$

(d) Image features (e) Lateral position

(f) Roll and pitch (g) Yaw ψ, ψ_t

Figure 4.35: Simulation while using the proposed algorithm where the quadrotor (blue) followed a car (red).

known. The general nature of the target makes the visual servoing law broadly applicable. Simulation results demonstrate the proposed algorithm's performance relative to previously published work in [54] where the target was assumed horizontal and static.

4.5 Conclusion

In this chapter we investigated the control of a quadrotor using nonlinear DIBVS to regulate the lateral position of the vehicle relative to a static and moving visual feature point on the ground. All of the control laws are *dynamic* in that they directly account for the vehicles dynamics. All of the proposed methods shares the same idea of rewriting the dynamics into a form where the design is simplified. A summary of the proposed control laws are show in Table 4.8.

Another important aspect of this work is that it is validated in simulation and experiment. In contrast to most of the work on DIBVS, our approach is validated experimentally which demonstrates the method is robust to model uncertainty (e.g., errors in modelling aerodynamic forces and changes in height and yaw which are neglected in design). When compared to other DIBVS control laws in the literature the proposed control laws demonstrate similar performance and benefit from reduced computational complexity. This is an important attribute for onboard implementation with inexpensive microcontrollers.

Name	Equation	Stability Result	Target	Additional Assumptions
Generalized State Transformation	Theorem 4.1	LES	$n = 1$ point (blob), stationary	• Perfect control of v_3^b and ω_3^b • Small coupling terms $y_1 y_2 \dot{\phi}$ and $y_1 y_2 \dot{\theta}$ • Known quadrotor parameters m, k_τ, k_f
Virtual Camera	Theorem 4.2	GAS	$n \geq 2$ points (blobs), stationary, horizontal	• Known quadrotor parameters m, k_τ, k_f
Adaptive Virtual Camera	Theorem 4.3	GAS	$n \geq 2$ points (blobs), stationary, horizontal	• Constant or slowly moving parameters
Moving Targets	Theorem 4.4	GAS	$n \geq 2$ points (blobs), moving, horizontal	• Known target velocity • Known quadrotor parameters m, k_τ, k_f
Non-horizontal Targets	Algorithm 1	-	$n \geq 4$ points (blobs), stationary	• Target plane in the desired image is parallel to the camera plane • The camera is centred laterally above the target • The target rotates about its geometric centre

Table 4.8: A summary of the proposed DIBVS methods. All of the methods assume: the image features are in the FoV and being tracked at all times, the quadrotor is symmetrical (Diagonal inertia matrix), small angle assumption η and a known vehicle attitude and velocity.

Chapter 5

Visual State Estimation

Visual odometry (VO) is the process of determining a camera's pose with respect to its environment [16, 17]. While there are many different methods of VO it typically involves finding visual features in an image, tracking or matching the features in subsequent images, and then motion estimation based on triangulation of the matched features. Closely related to VO is visual simultaneous localization and mapping (VSLAM). In addition to the localization, VSLAM also computes a map of the environment. One of the main benefits of which is loop closure. A survey on state of the art VSLAM methods is in [115].

VSLAM has been solved using a variety of cameras including stereo [116, 117], monocular [34, 118], and depth sensing [119, 120]. In this work we chose to use a monocular camera system and an inertial measurement unit (IMU) for visual state estimation (VSE) for practical reasons, i.e., visual inertial simultaneous localization and mapping (VISLAM). Monocular camera systems suffer from a lack of metric depth information. This would appear to motivate stereo cameras, however, stereo cameras need the observed scene to be within a small range according the displacement between the cameras. Once the scene is outside of this range the two images are the same and the metric depth information is lost. Inertial sensors are a natural choice to use as an aiding measurement as they are already onboard most unmanned aerial vehicles (UAVs). It is interesting to note the visual inertial systems are inspired by biology given that humans have the same sensors.

There is much work in the literature trying to solve the visual navigation problem. Here we highlight a few relevant papers. While work in [27] does not use inertial sensors they solve the scale problem by placing a calibration object with known measurements in front on camera. Unfortunately this solution is not robust to VSLAM reinitialization. When VSLAM systems lose tracking they have to reset. They can lose tracking due to occlusions (i.e., flying through a cloud), fast motion, by looking at uniform surfaces (e.g., a white floor), or non-static environ-

ments (i.e., flying over water). A truly autonomous system will need the ability to handle outages. In [121] and [118] the authors propose extended Kalman filter (EKF)-based VISLAM methods. In [36] the authors present an optimization-based VISLAM method. In all of the papers the authors present good experimental results, but require linearization of the state equations and provide no observability analysis. If an observability analysis is presented in the literature (e.g., [34]) they commonly use *local weak observability* defined in [122]. The downside of this approach is that the results are local, do not show when the system is unobservable, and do not lead to an observer design.

Developing VSLAM systems for UAVs presents a number of challenges. Onboard processing is required since radio links and ground station processing would introduce delay and limit autonomy. For outdoor flights the system must operate in large outdoor environments where weather, time of day, and fast six degree of freedom (DoF) motion lead to blurred or underexposed images. Existing research is often restricted to simplified environments, e.g., hand-held cameras in small-scale indoor environments [27]. Our work proposes an approach which is feasible to implement using hardware typically found onboard UAVs.

We focus on the estimation of linear components of the state and the proposed observers assume the rotational components of the state are estimated by an onboard attitude and heading reference system (AHRS). This assumption is not typically made in existing work. However, it is not restrictive as all UAVs have an AHRS onboard which is required for attitude stabilization. A typical application for the proposed observers is to maintain an accurate position measurement during intervals where global navigation satellite system (GNSS) or motion capture system (MCS) measurements are lost. An AHRS is typically implemented by fusing 3 axis accelerometer, 3 axis gyroscope, and 3 axis magnetometer measurements. Without external aiding measurements (e.g., GNSS) and under the usual assumption of near hover flight, the estimate of such an AHRS will slowly drift. However, we have found experimentally that this error is negligible over a typical quadrotor battery charge and hence ignored. For aggressive maneuvers with large linear acceleration the VSLAM attitude can be used to generate an improved attitude estimate. However, including attitude as a state in the VISLAM problem is left for future work.

5.1 Monocular Visual Inertial SLAM

System modelling requires the four frames of references shown in Figure 5.1. The navigation frame $\mathcal{N}$ which is assumed inertial, the body frame $\mathcal{B}$ which is located at the centre of mass (CoM) of the UAV, the camera frame $\mathcal{C}$ located at the camera's optical centre and is rigidly attached to the UAV, and the vision frame $\mathcal{V}$

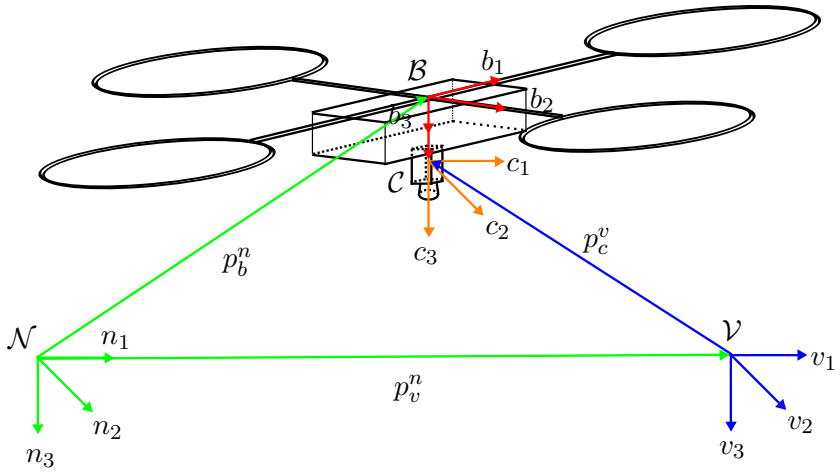


Figure 5.1: The reference frames used for VISLAM modelling. A navigation frame $\mathcal{N}$, body frame $\mathcal{B}$, camera frame $\mathcal{C}$, and vision frame $\mathcal{V}$ are shown.

which is the reference frame defined by the VSLAM system. The basis of $\mathcal{N}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{V}$ are $\{n_1, n_2, n_3\}$, $\{b_1, b_2, b_3\}$, $\{c_1, c_2, c_3\}$, and $\{v_1, v_2, v_3\}$, respectively. The frames $\mathcal{B}$ and $\mathcal{C}$ are assumed offset by a known constant displacement $p_c^b \in \mathbb{R}^3$ and rotation $R_b^c \in \mathrm{SO}(3)$ which can be computed in an initial calibration experiment. Superscripts on a vector indicate the frame in which it is represented.

We denote $p_b^n, v_b^n, a_b^n \in \mathbb{R}^3$ as the position, velocity, and acceleration of the origin of $\mathcal{B}$ expressed in $\mathcal{N}$, respectively. An onboard IMU contains an accelerometer which measures specific force $f_s^b \in \mathbb{R}^3$.

$$f_s^b = a^b - g^b$$

where $g^b \in \mathbb{R}^3$ is acceleration due to gravity. We model the accelerometer measurements as including a constant bias $b^b \in \mathbb{R}^3$. Hence, a_b^n can be expressed in terms of the accelerometer measurement f_s^b as

$$a_b^n = R_b^n(f_s^b + b^b) + g^n \tag{5.1}$$

where $R_b^n \in \mathrm{SO}(3)$ is the rotation between $\mathcal{B}$ and $\mathcal{N}$, and $g^n \in \mathbb{R}^3$ is acceleration due to gravity. The value of R_b^n is measured by the onboard AHRS and g^n is assumed known.

The output $y \in \mathbb{R}^3$ that VSLAM systems provide is a scaled measurement of the position of the camera in $\mathcal{V}$ which is denoted $p_c^v \in \mathbb{R}^3$. The scale parameter is denoted $\lambda > 0$. It also measures R_c^v which is the rotation between $\mathcal{C}$ and $\mathcal{V}$. The constant scale factor λ is due a monocular VSLAM system's inability to sense depth of features. Although we consider the specific monocular parallel tracking and mapping (PTAM) VSLAM system, any algorithm can be used in its place.

The value of λ is different every time PTAM starts or after re-initialization when tracking fails. PTAM can lose tracking due to occlusions (i.e., flying through a cloud), fast motion, by looking at uniform surfaces (e.g., a white floor), or non-static environments (i.e., flying over water). Hence, it is essential for the VISLAM system to be able to reconverge to its new value.

Since R_b^n and R_c^v are measured and R_b^c is available from a pre-flight calibration procedure, we can compute

$$R_n^v = R_c^v R_b^c R_n^b \tag{5.2}$$

The dynamics of the system in terms of the AHRS, IMU, and VSLAM measurements is

$$\begin{aligned} \dot{p}_b^n &= v_b^n \\ \dot{v}_b^n &= R_b^n(f_s^b + b^b) + g^n \\ \dot{b}^b &= 0 \\ \dot{\lambda} &= 0 \\ \dot{p}_v^n &= 0 \\ y &= \lambda p_c^v = \lambda R_n^v(R_b^n p_c^b + p_b^n - p_v^n) \end{aligned} \tag{5.3}$$

We have not included measurement noise in (5.3) which is present in the physical system. This is done to simplify the presentation of the observer design and observability analysis. In the simulations measurement noise is included. For convenience we define

$$\zeta = R_b^n f_s^b + g^n$$

We remark that the output measurement y depends on the relative pose between the origins of $\mathcal{N}$ and $\mathcal{V}$. This constant offset is unknown. However it can be determined algebraically when a reference measurement at a known position is available, e.g., the vehicle at time t_{ref} is placed at a reference position $p_b^n(t_{\text{ref}})$:

$$p_v^n = R_b^n(t_{\text{ref}})p_c^b + p_b^n(t_{\text{ref}}) - R_v^n \frac{y(t_{\text{ref}})}{\lambda} \tag{5.4}$$

Having a convergent observer for the scale factor λ and the relative position $(p_b^n - p_v^n)$ is required to make use of this algebraic relationship and to obtain the position p_b^n of the UAV in an absolute reference frame such as GNSS. In presence of noise a simple low pass filter can be used alternatively to (5.4):

$$\dot{\hat{p}}_v^n = \frac{1}{T_p}\left(R_b^n(t_{\text{ref}})p_c^b + p_b^n(t_{\text{ref}}) - \hat{R}_v^n \frac{y(t_{\text{ref}})}{\hat{\lambda}} - \hat{p}_v^n\right) \tag{5.5}$$

where $T_p > 0$ is a time constant. For rotation between $\mathcal{N}$ and $\mathcal{V}$ also a low pass filter could be used as an alternative to (5.2). In order to assure fast convergence

and steady state accuracy in presence of noise we decided to use a filter with an one dimensional Kalman gain $k_\eta > 0$

$$\begin{aligned} \dot{\hat{\eta}}_n^v &= k_\eta \left(\eta_n^v - \hat{\eta}_n^v\right) \\ \dot{k}_\eta &= -k_\eta^2 + q_\eta \end{aligned} \tag{5.6}$$

where $q > 0$, $\eta_n^v \in \mathbb{R}^3$ is $R_n^v = R_c^v R_b^c R_n^b \in \mathrm{SO}(3)$ expressed in the ZYX Euler angles and $k_\eta(0) > 0$.

5.2 Observability

We remark that most works on VSLAM analyzes the nonlinear observability of a system using the concept of *local weak observability* defined in [122]. In this work we separate the rotational dynamics from the translational dynamics and propose a coordinate transform that allows us to analyze the observability of the system using linear system theory. One of the main advantages of this approach is that local weak observability only states that a neighbourhood exists where the system is observable. From [123, Theorem 9.10] we are able to derive stronger conclusions about when the system is observable and equally as important when the system is not observable.

Observability Definitions

First, we recall some pertinent definitions and theorems about observability. We consider the system Σ

$$\Sigma : \begin{cases} \dot{x} = f(x, u) \\ y = g(x) \end{cases}$$

where $u \in \Omega$, a subset of $\mathbb{R}^m$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^p$ and f and g are C^∞ functions. For linear systems

$$\begin{aligned} \dot{x} &= A(t)x + B(t)u \\ y &= C(t)x \end{aligned} \tag{5.7}$$

Definition 5.1. [123, Definition 9.7] The linear system (5.7) is *observable* on $[t_0, t_1]$ if any initial state $x(t_0) = x_0$ is uniquely determined by the corresponding response $y(t)$ for $t \in [t_0, t_1]$.

Theorem 5.1. *[123, Theorem 9.10] Let q be a positive integer such that, for $t \in [t_0, t_1]$, $C(t)$ is q-times continuously differentiable, and $A(t)$ is $(q-1)$-times differentiable. Then the linear state equation* (5.7) *is observable on $[t_0, t_1]$ if for*

some $t_a \in [t_0, t_1]$ *the observability matrix* O *satisfies*

$$\text{rank}(O) = n$$

where

$$O = \begin{bmatrix} N_0(t_a) \\ N_1(t_a) \\ \vdots \\ N_q(t_a) \end{bmatrix}$$

and

$$\begin{aligned} N_0 &= C(t) \\ N_i &= N_{i-1}A(t) + \frac{dN_{i-1}}{dt},\ i = 1, 2, \ldots, q \end{aligned}$$

Definition 5.2. [124] The pair (A, C) is *uniformly detectable* (UD) if there exists constants $S > 0$, $T > 0$, $b > 0$ and $0 \leq d < 1$, such that whenever

$$\|\Phi(t + T, t)x\| \geq d\|x\|$$

for some $x \in \mathbb{R}^n$ and t, then

$$x^T W(t + S, t)x \geq bx^T x$$

where

$$W(t + S, t) = \int_t^{t+S} \Phi^T(s, t)C^T(s)C(s)\Phi(s, t)\,\mathrm{d}s$$

with A and C bounded.

For our observability analysis we use the notion of *uniform complete observability* (UCO) which ensures convergence of the Kalman Filter [125]. We denote $\Phi(s, t)$, $s, t \in \mathbb{R}$ as the transition matrix associated with A. We have

$$\begin{aligned} \tfrac{\partial \Phi}{\partial s}(s, t) &= A(s)\Phi(s, t) \\ \Phi(t, t) &= I \end{aligned}$$

The observability Gramian for (A, C) is

$$M(s, t) = \int_t^s \Phi^T(\tau, t)C^T(\tau)\Sigma(\tau)C(\tau)\Phi(\tau, t)\,\mathrm{d}\tau$$

where $s \geq t \geq 0$, and $\Sigma(t) \in \mathbb{R}^{m \times m}$ is a bounded symmetric positive definite matrix.

Definition 5.3. The pair (A, C) is UCO if there exists positive constants α, β, T such that

$$\forall t \geq 0 : \quad \alpha I \leq M(t+T, t) \leq \beta I$$

where A and C are bounded.

As UCO is difficult to check, following Sontag [126], a necessary and sufficient rank condition can be given which is based on the observability matrix O_k.

Theorem 5.2. *The matrix pair (A, C) is UCO if for every nontrivial interval $[t_0, t_1] \subseteq \mathbb{R}$ such that $0 \leq t_0 \leq t_1$ there exists a $t_* \in [t_0, t_1]$ and some non-negative integer k such that*

$$\operatorname{rank} O_k(t_*) = n \tag{5.8}$$

where

$$O_k = [C_0^T, C_1^T, \cdots, C_k^T]^T$$

and

$$C_i = \begin{cases} C, & i = 0 \\ C_{i-1}A + \dfrac{dC_{i-1}}{dt}, & 1 \leq i \leq k \end{cases}$$

with A and C bounded and sufficiently smooth. Furthermore, if A and C are analytic the rank condition is also necessary.

In other words, we can say the pair (A, C) is not observable if there exists some nontrivial interval $[t_0, t_1] \subseteq \mathbb{R}$ such that $0 \leq t_0 \leq t_1$, for all $t_* \in [t_0, t_1]$: $\operatorname{rank} O_k(t_*) < n$ and (A, C) is analytic.

Now we examine nonlinear observability from [122]. We consider the system Σ and the input-output map of the pair (Σ, x_0).

Definition 5.4. [122] Σ is *locally weakly observable* at x_0 if there exists a neighbourhood U of x_0 such that for every open neighbourhood V of x_0 contained in U, $I_V(x_0) \cap U = \{x_0\}$; and Σ is locally weakly observable if it is so at every $x \in M$.

Theorem 5.3. *[122, Theorem 3.1] If Σ satisfies the observability rank condition at x^0 then Σ is locally weakly observable at x^0.*

$$\operatorname{rank}(O) = n$$

where

$$O = d\mathcal{G}(x_0)$$

where the elements of $d\mathcal{G}$ are the finite linear combinations of $dL_{f^1}(\cdots(L_{f^k}(g_i))$ and $f^j(x) = f(x, u^j)$ for some constant $u^j \in \Omega$, $\mathcal{L}_f g_i(x)$ is the Lie derivative of g_i with respect to f at $x \in M$,

$$\mathcal{L}_{f^j} g_i(x) = \frac{\partial g_i(x)}{\partial x} f^j(x)$$

and

$$\mathcal{L}_{f^1}\mathcal{L}_{f^2} g_i(x) = \frac{\partial \mathcal{L}_{f^2} g_i(x)}{\partial x} f^1(x)$$

One DoF Observability

We start by examining observability for the one DoF VISLAM problem. This simpler setting provides insight for treating the general three DoF case. The dynamics is

$$\begin{aligned} \dot{p}^n &= v^n \\ \dot{v}^n &= a^n \\ y &= \lambda p^n \end{aligned}$$

where $p^n \in \mathbb{R}$ is position, $v^n \in \mathbb{R}$ is velocity, $a^n \in \mathbb{R}$ is the acceleration of the vehicle, and $y \in \mathbb{R}$ is the scaled position from the VSLAM system. For simplicity we neglect gravity from specific force: $f_s^b = a^b - b^b$, where $b^b \in \mathbb{R}$ denotes bias. The bias and the VSLAM map scale are included in the system state for observability analysis. The extended dynamics is

$$\begin{aligned} \dot{p}^n &= v^n \\ \dot{v}^n &= b^n + u \\ \dot{b}^n &= 0 \\ \dot{\lambda} &= 0 \\ y &= \lambda p^n \end{aligned} \tag{5.9}$$

where $u = f_s^n$. We note that in the one DoF model $f_s^n = f_s^b$ and $b^n = b^b$. In the observability analysis we treat u as a parameter. We propose the following change of coordinates

$$\begin{aligned} z_1 &= \lambda p^n \\ z_2 &= \lambda v^n \\ z_3 &= \lambda b^n \\ z_4 &= \lambda \end{aligned}$$

This change of coordinates is globally defined provided $\lambda > 0$. Here we have take z_1, z_2, z_3 as the Lie derivatives of the output function λp^n. These derivatives are often

used for normal form observer design. The dynamics of (5.9) in the z coordinates is

$$\begin{aligned}\dot{z} &= Az \\ y &= Cz\end{aligned}$$

where

$$\begin{aligned}A &= \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & u\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \\ C &= \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\end{aligned}$$

Using Theorem 5.1 we calculate the observability matrix to be

$$O = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & u\\ 0 & 0 & 0 & \dot{u}\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & 0 & u^{(q-2)}\end{bmatrix} \in \mathbb{R}^{(q+1)\times 4}$$

A sufficient condition for observability is that O be full rank. The matrix is full rank, i.e., $\operatorname{rank} O = 4$, when u is non constant. This implies that when the vehicle is accelerating the system is observable.

Three DoF Observability

In order to simplify the observability analysis and later the observer design, we introduce the state transformation

$$z = \lambda[(p_b^n - p_v^n)^T, (v_b^n)^T, (b^b)^T, 1]^T \tag{5.10}$$

As a result of the discussion in the previous section we have chosen $z_1 = \lambda(p_b^n - p_v^n)$ as a component of the state and not directly λp_b^n. The absolute position estimate $\hat{p}_b^n$ can be calculated algebraically by using $\hat{z}$ and (5.5) once the observer for z converges.

Expressed in the z-coordinates, (5.3) is an linear time-varying (LTV) dynamics

$$\begin{aligned}\dot{z} &= A(t)z \\ y &= C(t)z\end{aligned} \tag{5.11}$$

where $z \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $n = 10$, $m = 3$,

$$A(t) = \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & R_b^n(t) & \zeta(t) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$C(t) = \begin{bmatrix} R_n^v & 0 & 0 & R_n^v R_b^n(t) p_c^b \end{bmatrix} \tag{5.12}$$

and where we have introduced the time-varying signal $\zeta(t) = R_b^n(t) f_s^b(t) + g^n$. In (5.12) we have indicated which quantities are explicitly time-varying. To simplify notation below we drop this time-dependence.

Using Theorem 5.2 and the VISLAM dynamics defined in (5.11) and (5.12), we have

$$O_k = \left[\begin{array}{cc:cc} R_n^v & 0 & 0 & R_n^v R_b^n p_c^b \\ 0 & R_n^v & 0 & R_n^v \dot{R}_b^n p_c^b \\ \hdashline 0 & 0 & R_n^v R_b^n & R_n^v \zeta + R_n^v \ddot{R}_b^n p_c^b \\ 0 & 0 & R_n^v \dot{R}_b^n & R_n^v \dot{\zeta} + R_n^v R_b^{n(3)} p_c^b \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & R_n^v R_b^{n(k-2)} & R_n^v \zeta^{(k-2)} + R_n^v R_b^{n(k)} p_c^b \end{array}\right]$$

where $O_k \in \mathbb{R}^{m(k+1) \times n}$. Before examining the rank of O_k we first define the Guttman rank additivity formula [127]. Given the matrix $M \in \mathbb{R}^{m \times n}$ and the submatrices M_{ij} of appropriate size such that

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

and M_{11} is square and nonsingular then the Guttman rank additivity formula is

$$\operatorname{rank} M = \operatorname{rank} M_{11} + \operatorname{rank}(M_{22} - M_{21} M_{11}^{-1} M_{12}) \tag{5.13}$$

Now, to compute the rank of O_k it has been partitioned as shown above. The upper left submatrix has rank of six. Thus, only the rank of the lower right submatrix has to be considered. We apply (5.13) to the lower right submatrix of O_k with $M_{11} = R_n^v R_b^n$ to obtain

$$\operatorname{rank} O = 9 + \operatorname{rank} \delta$$

and $\operatorname{rank}\delta = \operatorname{rank}\bar{\delta}$ where

$$\begin{aligned}\delta &= \text{blkdiag}(R_n^v, \ldots, R_n^v)\cdot\bar{\delta}\\ \bar{\delta} &= \begin{bmatrix} \dot{\zeta} + R_b^{n(3)}p_c^b - \dot{R}_b^n R_n^b(\zeta + \ddot{R}_b^n p_c^b) \\ \vdots \\ \zeta^{(k-2)} + R_b^{n(k)}p_c^b - R_b^{n(k-2)}R_n^b(\zeta + \ddot{R}_b^n p_c^b)\end{bmatrix}\end{aligned}$$

and $\delta, \bar{\delta} \in \mathbb{R}^{3(k-2)\times 1}$. The system is UCO if $\bar{\delta}$ is nonzero. Taking the derivative of $\bar{\delta}_\ell(t)$ leads to

$$\dot{\bar{\delta}}_\ell(t) = \delta_{\ell+1}(t) - R_b^{n(\ell)}(t)R_n^b(t)\bar{\delta}_1(t).$$

Thus if $\bar{\delta}_1(t) = 0$ then $\bar{\delta}(t) = 0$ and therefore $\bar{\delta}_1(t)$ must be nonzero for $O_k(t)$ to have full rank. We can see when there is no camera offset (i.e., $p_c^b = 0$) then the system is UCO when $||\dot{\zeta} - \text{sk}(\omega^n)\zeta|| > 0$, where $\omega^n \in \mathbb{R}^3$ is the angular velocity of the vehicle in $\mathcal{N}$ and $\dot{R}_b^n = \text{sk}(\omega^n)R_b^n$. On the other hand when if there is no acceleration (i.e., $p_c^b = 0$ and $\dot{v}^n = 0$) and $R_b^n(t)$ is analytic then the system is not observable. We can also see that the system is not observable when it is accelerating constantly and not rotating ($\zeta = $ const. and $\omega^b = 0$). In these cases the observability matrix O_k is a constant and thus the rank condition with $k = n - 1$ becomes necessary and sufficient for observability [126].

Three DoF Nonlinear Observability

Most works in the literature (e.g., [34]) use the condition for nonlinear observability from Theorem 5.3. For completeness we present the observability analysis here. Given the extended dynamics in (5.3) we have

$$\begin{aligned} f(x) &= \begin{bmatrix} v_b^n \\ R_b^n b \\ 0_{3\times 1} \\ 0 \end{bmatrix} \\ g(x) &= \begin{bmatrix} g_1 & g_2 & g_3 \end{bmatrix} \\ &= \begin{bmatrix} 0_{3\times 3} \\ R_b^n \\ 0_{1\times 3} \\ 0_{3\times 3} \end{bmatrix} \\ h(x) &= \lambda R_n^v(R_b^n p_c^b + p_b^n - p_v^n) \end{aligned} \tag{5.14}$$

where

$$x = [(p_b^n - p_v^n)^T, (v_b^n)^T, (b^b)^T, \lambda]^T$$

To calculate the observability matrix we first calculate

$$\mathcal{G} = [\mathcal{G}_1^T, \mathcal{G}_2^T, \mathcal{G}_3^T, \mathcal{G}_4^T]^T \tag{5.15}$$

where

$$\mathcal{G}_1 = \begin{bmatrix} h \\ \mathcal{L}_f h \\ \mathcal{L}_f^2 h \\ \mathcal{L}_f^3 h \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda R_n^v (R_b^n p_c^b + p_b^n - p_v^n) \\ \lambda R_n^v v^n \\ \lambda R_n^v R_b^n b^b \\ 0 \\ \vdots \end{bmatrix}, \quad \mathcal{G}_2 = \begin{bmatrix} \mathcal{L}_{g1}\mathcal{L}_f h \\ \mathcal{L}_{g1}\mathcal{L}_{g1}\mathcal{L}_f h \\ \mathcal{L}_{g2}\mathcal{L}_{g1}\mathcal{L}_f h \\ \mathcal{L}_{g3}\mathcal{L}_{g1}\mathcal{L}_f h \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda R_n^v {r_1}_b^n \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

$$\mathcal{G}_3 = \begin{bmatrix} \mathcal{L}_{g2}\mathcal{L}_f h \\ \mathcal{L}_{g1}\mathcal{L}_{g2}\mathcal{L}_f h \\ \mathcal{L}_{g2}\mathcal{L}_{g2}\mathcal{L}_f h \\ \mathcal{L}_{g3}\mathcal{L}_{g2}\mathcal{L}_f h \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda R_n^v {r_2}_b^n \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \quad \mathcal{G}_4 = \begin{bmatrix} \mathcal{L}_{g3}\mathcal{L}_f h \\ \mathcal{L}_{g1}\mathcal{L}_{g3}\mathcal{L}_f h \\ \mathcal{L}_{g2}\mathcal{L}_{g3}\mathcal{L}_f h \\ \mathcal{L}_{g3}\mathcal{L}_{g3}\mathcal{L}_f h \\ \vdots \end{bmatrix} = \begin{bmatrix} \lambda R_n^v {r_3}_b^n \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

where $R_b^n = [{r_1}_b^n, {r_1}_b^n, {r_3}_b^n]$. Then

$$O = \mathrm{d}\mathcal{G} = [\mathrm{d}\mathcal{G}_1^T, \mathrm{d}\mathcal{G}_2^T, \mathrm{d}\mathcal{G}_3^T, \mathrm{d}\mathcal{G}_4^T]^T \tag{5.16}$$

where

$$\mathrm{d}\mathcal{G}_1 = \begin{bmatrix} \lambda R_n^v & 0 & 0 & R_n^v (R_b^n p_c^b + p_b^n - p_v^n) \\ 0 & \lambda R_n^v & 0 & R_n^v v_b^n \\ 0 & 0 & \lambda R_n^v R_b^n & R_n^v R_b^n b^b \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$\mathrm{d}\mathcal{G}_2 = \begin{bmatrix} 0 & 0 & 0 & R_n^v {r_1}_b^n \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathrm{d}\mathcal{G}_3 = \begin{bmatrix} 0 & 0 & 0 & R_n^v {r_2}_b^n \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad \mathrm{d}\mathcal{G}_4 = \begin{bmatrix} 0 & 0 & 0 & R_n^v {r_3}_b^n \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Lastly, we calculate the $\mathrm{rank}(O) = 10$ and thus O is full rank and the system is locally weakly observable. However, when compared to the observability analysis in Section 5.2 it is difficult to draw conclusions on the effect that u and p_c^b have on

the observability. In fact it appears to observable regardless of u. Furthermore, this analysis does not lead to an observer design where as the previous section's rank condition is necessary and sufficient for UCO and UCO leads to multiple observer designs.

5.3 Observer Design

The typical observer used in the literature is an EKF. The EKF equations are

$$\dot{\hat{x}} = f(\hat{x}, u) + K(y - h(\hat{x})) \tag{5.17}$$
$$\dot{P} = -PH^T R^{-1} CP + FP + PF^T + Q \tag{5.18}$$
$$K = PH^T R^{-1} \tag{5.19}$$

where $Q \in \mathbb{R}^{n \times n}$ is the process noise covariance, $R \in \mathbb{R}^{m \times m}$ is the measurement noise covariance and

$$F = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}}$$
$$H = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}}$$

When applied to the dynamics for the VISLAM problem (5.3) we obtain

$$\begin{aligned} f(x,u) &= \begin{bmatrix} v_b^n \\ R_b^n b^b + u \\ 0 \\ 0 \end{bmatrix} \\ h(x) &= \lambda R_n^v (R_b^n p_c^b + p_b^n - p_v^n) \\ F &= \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & R_b^n & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ H &= \begin{bmatrix} R_n^v \hat{\lambda} & 0 & 0 & R_n^v R_b^n p_c^b + R_n^v (p_b^n \hat{-} p_v^n) \end{bmatrix} \end{aligned} \tag{5.20}$$

where $x = [(p_b^n - p_v^n)^T, (v_b^n)^T, (b^b)^T, \lambda]^T$ and $u = \zeta = R_b^n f_s^b + g^n$. We note that $f(x,u)$ is LTV and does not require linearization, however, $h(x)$ is nonlinear.

A better observer is to use the coordinate transform in (5.10) to obtain the

dynamics in (5.3) and then use a Kalman filter (KF). The KF equations are

$$\begin{aligned} \dot{\hat{z}} &= A\hat{z} + K(y - C\hat{z}) \\ \dot{P} &= -KCP + AP + PA^T + Q \\ K &= PC^T R^{-1} \end{aligned} \tag{5.21}$$

where $P \in \mathbb{R}^{n\times n}$ is the covariance matrix, $K \in \mathbb{R}^{n\times m}$ is the Kalman gain, $Q \in \mathbb{R}^{n\times n}$ is the process noise covariance and $R \in \mathbb{R}^{m\times m}$ is the measurement noise covariance. We remark UCO is a sufficient condition to ensure the stability of a KF for output error systems [128].

Lastly, we propose an adaptive observer (AO) for (5.11) following the design in [129] which is an extension of [130]. We consider a general system form

$$\begin{aligned} \dot{\bar{z}} &= \bar{A}\bar{z} + \Psi\theta \\ y &= \bar{C}\bar{z} + \Xi\theta \end{aligned} \tag{5.22}$$

where $\bar{z} \in \mathbb{R}^{\bar{n}}$, $\bar{A} \in \mathbb{R}^{\bar{n}\times\bar{n}}$, $\bar{C} \in \mathbb{R}^{m\times\bar{n}}$, $\Psi \in \mathbb{R}^{\bar{n}\times p}$, and $\Xi \in \mathbb{R}^{m\times p}$ are known time-varying matrices and $\theta \in \mathbb{R}^p$ is a vector of unknown constant parameters.

In [129, 130] two assumptions are given which are necessary to show convergence of the observer error dynamics. For these assumptions no conditions had been given such that they hold. Hence in the following theorem we determine a sufficient condition that assures convergence of the adaptive observer presented afterwards.

Theorem 5.4. *If (A, C) with A and C bounded is UCO then:*

1. *There exists a bounded, locally integrable, time-varying matrix $\bar{K}(t) \in \mathbb{R}^{\bar{n}\times m}$ such that the system*

$$\dot{\eta} = (\bar{A} - \bar{K}\bar{C})\eta \tag{5.23}$$

is exponentially stable.

2. *Let $\Upsilon \in \mathbb{R}^{\bar{n}\times p}$ be generated by*

$$\dot{\Upsilon} = (\bar{A} - \bar{K}\bar{C})\Upsilon + \Psi - \bar{K}\Xi \tag{5.24}$$

The signal $\Psi - \bar{K}\Xi$ is persistently exciting (PE), i.e., there exists constants α, β, T and a bounded symmetric positive-definite matrix Σ such that

$$\alpha I \leq \int_t^{t+T} Z^T(\tau)\Sigma(\tau)Z(\tau)d\tau \leq \beta I, \quad t \geq 0 \tag{5.25}$$

where $Z(\tau) = \Xi(\tau) + \bar{C}(\tau)\Upsilon(\tau)$.

Proof.

1. Following [124, Corollary 5.1] and interpreting it for the continuous time case there exists a bounded, locally integrable, time-varying matrix $\bar{K}(t)$ such that (5.23) is exponentially stable if and only if the pair $(\bar{A}, \bar{C})$ is UD Definition 5.2. Clearly, UCO is a sufficient condition for UD. It is easy to see that

$$\Phi(s,t) = \begin{bmatrix} \bar{\Phi}(s,t) & \int_t^s \bar{\Phi}(s,\tau)\Psi(\tau)\,\mathrm{d}\tau \\ 0 & 1 \end{bmatrix}$$

where $\bar{\Phi}$ is the transition matrix associated with $\bar{A}$. Thus we get

$$M(s,t) = \begin{bmatrix} \bar{M}(s,t) & * \\ * & * \end{bmatrix}$$

where $\bar{M}(s,t)$ is the observability Gramian for the pair $(\bar{A}, \bar{C})$. Hence if (A, C) is UCO it immediately it follows that $(\bar{A}, \bar{C})$ is UCO and therefore UD. This proves Condition 1.

2. We introduce the state matrix

$$X(t) = [\Upsilon^T(t), I]^T \in \mathbb{R}^{n\times p}$$

and rewrite (5.24) as

$$\begin{aligned} \dot{X} &= \begin{bmatrix} \bar{A} - \bar{K}\bar{C} & \Psi - \bar{K}\Xi \\ 0 & 0 \end{bmatrix} X \\ &= (A - KC)X \end{aligned}$$

where $K(t) = [\bar{K}^T(t), 0]^T \in \mathbb{R}^{n\times m}$ is bounded and locally integrable. Let the pair (A, C) be UCO, then the pair $(A - KC, C)$ is also UCO [131]. Thus there exists positive constants α, β, T such that for every bounded symmetric positive definite matrix $\Sigma(s) \in \mathbb{R}^{m\times m}$

$$\alpha I \leq \int_t^{t+T} \tilde{\Phi}^T(s,t)C^T(s)\Sigma(s)C(s)\tilde{\Phi}(s,t)\,\mathrm{d}s \leq \beta I \tag{5.26}$$

where $\tilde{\Phi}$ denotes the transition matrix associated to $A - KC$. Left and right

sided multiplication of (5.26) with $X^T(t)$ and $X(t)$ respectively yields

$$\bar{\alpha}I \leq \int_t^{t+T} X^T(s)C^T(s)\Sigma(s)C(s)X(s)\,\mathrm{d}s \leq \bar{\beta}I \tag{5.27}$$

where $\bar{\alpha}$, $\bar{\beta}$ are positive constants. The lower bound $\bar{\alpha}$ exists since $X^T(t)X(t) \geq I$. The boundedness of X implies $\bar{\beta}$ exists. Applying the definitions of C and X to (5.27) provides

$$\bar{\alpha}I \leq \int_t^{t+T} Z^T(\tau)\Sigma(s)Z^T(\tau)\,\mathrm{d}s \leq \bar{\beta}I$$

Hence we have proven Condition 2.

□

Theorem 5.5. *If (A, C) is UCO then*

$$\begin{aligned}\dot{\hat{\bar{z}}} &= \bar{A}\hat{\bar{z}} + \Psi\hat{\theta} + (\bar{K} + \Upsilon\Gamma Z\Sigma)(y - \bar{C}\hat{\bar{z}} - \Xi\hat{\theta}) \\ \dot{\hat{\theta}} &= \Gamma Z^T\Sigma(y - \bar{C}\hat{z} - \Xi\hat{\theta})\end{aligned} \tag{5.28}$$

where Γ is a positive definite symmetric matrix, is a globally exponentially stable (GES) joint state and parameter observer for system (5.22).

For completeness we present a proof based on [129, 130] that requires the following lemma [131, 132].

Lemma 5.1. Let $\Omega(t) \in \mathbb{R}^{m\times p}$ be bounded for all $t \geq 0$ and $\Gamma \in \mathbb{R}^{p\times p}$ be any symmetric positive definite matrix. If there exists constants $\delta, T > 0$ such that for all $t \geq 0$ then $\dot{x}(t) = \Gamma\Omega^T(t)\Omega(t)x(t)$ where $x(t) \in \mathbb{R}^p$ is exponentially stable.

Proof. We combine the equations from (5.28) to obtain

$$\dot{\hat{\bar{z}}} = \bar{A}\hat{\bar{z}} + \Psi\hat{\theta} + \bar{K}(y - \bar{C}\hat{\bar{z}} - \Xi\hat{\theta}) + \Upsilon\dot{\hat{\theta}}$$

Next, we define the error signals $\tilde{z} = \bar{z} - \hat{\bar{z}}$ and $\tilde{\theta} = \theta - \hat{\theta}$. Given that $\dot{\theta} = 0$ we obtain

$$\dot{\tilde{z}} = (\bar{A} - \bar{K}\bar{C})\tilde{z} + \Psi\tilde{\theta} - \bar{K}\Xi\tilde{\theta} + \Upsilon\dot{\tilde{\theta}} \tag{5.29}$$

Next, we define the linear combination $\eta = \tilde{z} - \Upsilon\tilde{\theta}$ and thus

$$\begin{aligned}\dot{\eta} &= (\bar{A} - \bar{K}\bar{C})(\eta + \Upsilon\tilde{\theta}) + \Psi\tilde{\theta} - \bar{K}\Xi\tilde{\theta} - \dot{\Upsilon}\tilde{\theta} \\ &= (\bar{A} - \bar{K}\bar{C})\eta + \Big((\bar{A} - \bar{K}\bar{C})\Upsilon + \Psi - \bar{K}\Xi - \dot{\Upsilon}\Big)\tilde{\theta}\end{aligned}$$

Substitution of $\dot{\Upsilon}$ into $\dot{\eta}$ yields

$$\dot{\eta} = (\bar{A} - \bar{K}\bar{C})\eta$$

By assumption $(\bar{A} - K\bar{C})$ is asymptotically stable, thus $\eta(t) \to 0$ as $t \to \infty$. Next, we examine the behavior of $\tilde{\theta}$:

$$\begin{aligned}\dot{\tilde{\theta}} &= -\Gamma Z^T \Sigma(y - \bar{C}\hat{z} + \Xi\tilde{\theta}) \\ &= -\Gamma Z^T \Sigma(\bar{C}\tilde{z} + \Xi\tilde{\theta}) \\ &= -\Gamma Z^T \Sigma\left(\bar{C}(\eta + \Upsilon\tilde{\theta}) + \Xi\tilde{\theta}\right)\end{aligned} \tag{5.30}$$

Because $\eta(t) \to 0$ as $t \to \infty$ and all quantities in (5.30) are bounded it is sufficient for asymptotic stability of $\tilde{\theta}$ to consider the dynamics

$$\begin{aligned}\dot{\tilde{\theta}} &= -\Gamma Z^T \Sigma\left(\bar{C}\Upsilon\tilde{\theta} + \Xi\tilde{\theta}\right) \\ &= -\Gamma Z^T \Sigma Z\tilde{\theta}\end{aligned}$$

Finally we apply Lemma 5.1 with $\Omega = \Sigma^{\frac{1}{2}} Z$ and $x = \tilde{\theta}$ and we see that $\tilde{\theta}(t) \to 0$ when $t \to \infty$. □

Next, we examine the non-deterministic system.

$$\begin{aligned}\dot{\bar{z}} &= \bar{A}\bar{z} + \Psi\theta + n_z \\ \dot{\theta} &= n_\theta \\ y &= \bar{C}\bar{z} + \Xi\theta + n_y\end{aligned} \tag{5.31}$$

with noise terms $n_z \in \mathbb{R}^{\bar{n}}$, $n_\theta \in \mathbb{R}^p$ and $n_y \in \mathbb{R}^m$.

Theorem 5.6. *If the matrix pair $(\bar{A}, \bar{C})$ with $\bar{A}(t)$ and $\bar{C}(t)$ bounded is UCO and the noise terms n_z, n_θ, and n_y are bounded then the state and parameter estimation errors of the adaptive observer* (5.28) *applied to the system* (5.31) *are also bounded.*

Proof. The proof is based on [129] and then we apply Theorem 5.5. First, we redefine the error signals from (5.29) and (5.30) to include the noise terms:

$$\begin{aligned}\dot{\tilde{z}} &= (\bar{A} - \bar{K}\bar{C})\tilde{z} + \Psi\tilde{\theta} - \bar{K}\Xi\theta + n_z - \bar{K}n_y - \Upsilon n_\theta \\ \dot{\tilde{\theta}} &= -\Gamma Z^T \Sigma\left(\bar{C}(\eta + \Upsilon\tilde{\theta} + n_y) + \Xi\tilde{\theta}\right) + n_\theta\end{aligned} \tag{5.32}$$

From Theorem 5.5 the deterministic versions of $\tilde{z}$ and $\tilde{\theta}$ have been proven to be exponentially stable hence the homogeneous part of (5.32) is exponentially stable. From [133, Theorem 1 on p. 196] because n_z, n_θ, and n_y are bound and (5.29)

and (5.30) are exponentially stable then the error signals (5.32) are uniformly bounded-input, bounded-output (BIBO) stable. □

For the VISLAM problem the parameters of (5.22) are

$$\begin{aligned}
\bar{A} &= \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & R_b^n \\ 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{9\times 9} \\
\Psi &= \begin{bmatrix} 0 \\ \zeta \\ 0 \end{bmatrix} \in \mathbb{R}^{9\times 1} \\
\bar{C} &= \begin{bmatrix} R_n^v & 0 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 9} \\
\Xi &= R_n^v R_b^n p_c^b \in \mathbb{R}^{3\times 1}
\end{aligned} \tag{5.33}$$

where $\theta = \lambda$ and $\bar{z} = \lambda[(p_b^n - p_v^n)^T, (v_b^n)^T, (b^b)^T]^T \in \mathbb{R}^9$. From Theorem 5.5, the rank condition given in Theorem 5.2 ensure exponential convergence of the error in state and parameter estimates.

Choosing observer gain matrix $\bar{K}$ is a nontrivial problem. As the system is LTV, a suitable constant gain is not straightforward to find. Therefore, a Kalman gain is used. This will be stabilizing as the UCO condition holds. The matrix $\Gamma \in \mathbb{R}^{p\times p}$ is difficult to tune such that the observer (5.28) has a sufficient convergence rate and robustness to noise. This led to a time-varying $\Gamma(t)$ based on a recursive least squares algorithm with exponential forgetting factor proposed in [129]. We adapt this approach and use the adaption scheme

$$\dot{\Gamma} = -\gamma_1 \Gamma(C\Upsilon + \Psi)^T(C\Upsilon + \Psi)\Gamma + \gamma_2(y - C\hat{z} - \Psi\hat{\theta})^T(y - C\hat{z} - \Psi\hat{\theta})\hat{\theta}^T\hat{\theta}\Gamma \tag{5.34}$$

where $\gamma_1 > 0, \gamma_2 > 0$.

5.4 Parallel Tracking and Mapping

Many VSLAM methods have been proposed in the literature [115]. We choose the PTAM method which is a camera tracking system originally developed for augmented reality [24]. PTAM separates tracking and mapping into two separate threads. This allows tracking to run as fast as possible (typically at the frame rate of the camera), and the computationally demanding bundle adjustment (BA) (i.e., nonlinear optimization) is run at a slower rate. PTAM uses a keyframe-based algorithm, which means it optimizes its state over select keyframes and discards all other frames. These keyframes are distributed over space and allow the algorithm to

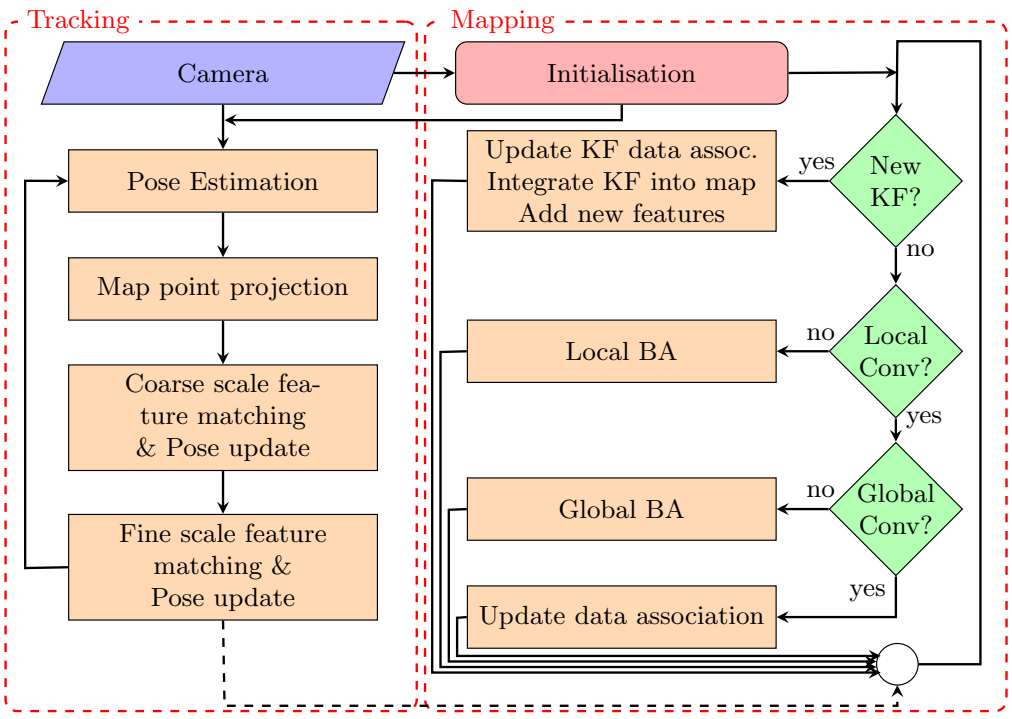


Figure 5.2: A flow diagram of the PTAM algorithm.

remove redundant information within the images. This allows the algorithm to run in real time. Bundle adjustment or optimization-based VSLAM methods have been shown to be more accurate and computationally efficient relative to filter-based approaches [134]. Every time a new image is received, the tracking algorithm attempts to locate itself to the nearest keyframe using coarse-to-fine feature matching. First, an estimate of the camera pose is obtained by using a decaying velocity model. Using this rough pose estimate the algorithm projects a relatively small number of 3 D points back onto the image. A pinhole camera model which includes barrel radial distortion is used for this projection. See Section 2.2.1. After the small number of 3 D features have been projected back onto the image the next step is to match the image features and update the pose. This process is then repeated with a large number of 3 D features to fine tune the pose. Both the coarse and fine pose updates are computed by iteratively solving the problem

$$\arg\min_{p,R} \sum_{j \in S} \text{Obj}\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right)$$

where j is the image feature, S is the set of coarse or fine image features, σ_j is the standard deviation of the measurement noise, $\text{Obj}(\cdot, \sigma_T)$ is the Tukey biweight objective function, and σ_T is a median-based estimate of the distribution's standard deviation derived from all of the residuals. The reprojection error e_j is calculated

by using the camera model (2.33), (2.31). The original PTAM algorithm tracked around fifty features per image, but on modern hardware it can track thousands of features.

The mapping thread goes through multiple stages depending on the amount of processor time it receives. Its first job is to decide when to add another keyframe. If it decides to add a new frame, it will calculate and store the new map features. If the algorithm does not need to add a new keyframe it will do a local BA over the last few images. After a local BA has been completed, PTAM will do a global BA over all keyframes in its map. Lastly, if the BA has converged and no new keyframes are needed, the algorithm will attempt to refine the data associated with the map by making new measurements or revising measurements in old keyframes. Outlier points can also be deleted at this time. A BA is performed by iteratively minimizing the same objective function as in the pose updates, but it is now done over all keyframes (i.e., global BA) or some keyframes (i.e., local BA).

$$\arg\min_{p,R} \sum_{i=1}^{N} \sum_{j \in S_i} \text{Obj}\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right)$$

where N is the number of keyframes and S_i is the set of image features in keyframe i. Outlier points can be deleted at this time.

One of the main problems with using the original PTAM algorithm is that it did not scale well to large environments. Multiple extensions have been proposed (e.g., [34, 118]) to make it more suitable for UAVs. The version of PTAM used in this work is based off the Eidgenössische Technische Hochschule Zürich (ETHZ) PTAM [34] which makes a number of improvements including defining a maximum number of keyframes, automatic re-initialization, and improved tracker speeds. Additionally, a robot operating system (ROS) wrapper was added [103]. A summary of the PTAM Algorithm is show in Figure 5.2.

5.5 Simulation

5.5.1 Simulations Results

To validate the proposed adaptive observer we simulated it in Matlab/Simulink. A comparison was performed with a KF designed in the z-coordinates and an EKF in the original x-coordinates. We simulated the dynamics in (5.3), the observers in (5.17), (5.21) and (5.33), and the filters in (5.5) and (5.6). The vehicle followed

the path

$$p^n(t) = \begin{bmatrix} 0.7 \cdot \cos(0.7 \cdot t) \\ 0.7 \cdot \sin(0.7 \cdot t) \\ 0.1 \cdot \sin(0.35 \cdot t) + 0.03 \cdot t - 1 \end{bmatrix}$$

and had an angular acceleration of

$$\tau^b(t) = \begin{bmatrix} 10^{-4} \sin(t+1) \\ 10^{-4} \sin(t+1.5) \\ 10^{-4} \sin(t+2) \end{bmatrix}$$

The parameters used in the simulation are summarized in Table 5.1.

The results of the simulation demonstrate the stability of the observers and their robustness to noise. All of the measured signals are shown in Figure 5.3. The measured acceleration has large variance and bias. Similarly, the measured scaled position has a large variance. The rotation matrices are represented by the ZYX Euler angles where R_y^x is described by η_y^x. Both the measured attitude from $\mathcal{N}$ to $\mathcal{B}$ and $\mathcal{V}$ to $\mathcal{C}$ also include noise. It is important to note in the figure at $t = 30$ s the VSLAM system lost tracking and was reset. The system can be reset by occlusion of the camera (e.g., flying through a cloud), fast motion, by looking at uniform surfaces (e.g., a white floor), or from a non-static scene (e.g., flying over water). In reality the outage would be for a finite $t > 0$ s where the only option for navigation left for the vehicle is dead reckoning. In the simulation we ignored the outage time as it it not relevant to this paper. Figure 5.4 shows the 3 D position trajectory of the UAV during the simulation.

In Figure 5.5 we see the rotational offset between the navigation frame $\mathcal{N}$ and the vision frame $\mathcal{V}$. We can see that at time $t = 30$ s the orientation of the frame is reset. In the figure the actual signal is blue, the measured signal is red, and the output of the low pass filter is in green. The filter converges sufficiently fast that its does not affect the cascaded observers.

Figure 5.6 shows the state trajectories (position, velocity, bias and scale) for the EKF in the original coordinates, the KF in the z-coordinates, and the proposed adaptive observer. It can be seen that the EKF performs well for the first 30 s, the point where the simulated VSLAM system was forced to reinitialize. Tuning the EKF to get satisfactory convergence for large as well as for small values of the scale λ is challenging. The KF and the AO on the other hand show good performance independent of the value for scale. Furthermore, we can see the negative affect of slow convergence of the scale factor has on the estimated position and velocity.

Next, in Figure 5.7 we see the translational offset from $\mathcal{N}$ to $\mathcal{V}$ that is used to get the estimate of the absolute vehicle position p_b^n out of the relative position

Parameter	Value	Units
$P_{\mathrm{AO}}(0)$	$\mathrm{blkdiag}(I, I, I)$	$\mathrm{diag}(\mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^4, \mathrm{m}^2/\mathrm{s}^4)$
Q_{AO}	$\mathrm{blkdiag}(0, 10^{-4}I, 10^{-4}I)$	$\mathrm{diag}(\mathrm{m}^2/\mathrm{s}^3, \mathrm{m}^2/\mathrm{s}^5, \mathrm{m}^2/\mathrm{s}^5)$
R_{AO}	$0.5 \cdot 10^{-3} I$	$\mathrm{diag}(\mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2)$
Σ	I	$1/(\mathrm{m}^2\,\mathrm{s})$
$\Gamma(0)$	50	-
γ_1	$2 \cdot 10^{-3}$	-
γ_2	$5 \cdot 10^{-3}$	-
$P_{\mathrm{EKF}}(0)$, $P_{\mathrm{KF}}(0)$	$\mathrm{blkdiag}(P_{\mathrm{AO}}(0), 0.1)$	$\mathrm{diag}(\mathrm{m}^2/\mathrm{s}^2, \mathrm{m}^2/\mathrm{s}^4, \mathrm{m}^2/\mathrm{s}^4, \text{-})$
Q_{EKF}, Q_{KF}	$\mathrm{blkdiag}(Q_{\mathrm{AO}}, 10^{-4})$	$\mathrm{diag}(\mathrm{m}^2/\mathrm{s}^3, \mathrm{m}^2/\mathrm{s}^5, \mathrm{m}^2/\mathrm{s}^5, 1/\mathrm{s})$
R_{EKF}, R_{KF}	R_{AO}	$\mathrm{diag}(\mathrm{m}^2, \mathrm{m}^2, \mathrm{m}^2)$

(a) Observer parameters.

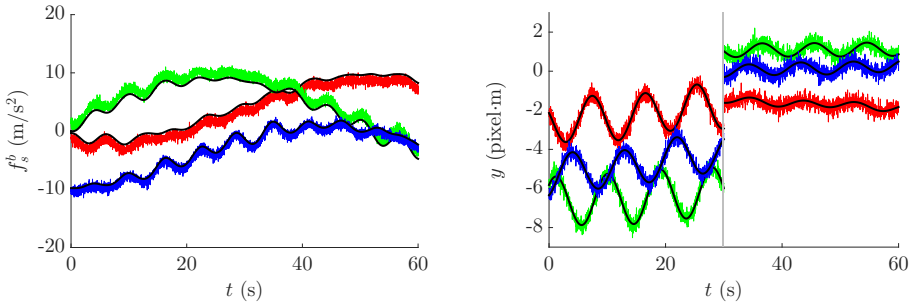
Parameter	Value	Units
$p_b^n(0)$	$[0.7, 0, -1]^T$	m
$v_b^n(0)$	$[0, 0.49, 0.035]^T$	m/s
b^b	$[1, -1, 0.5]^T$	$\mathrm{m/s}^2$
$\lambda(t < 30)$	2	-
$\lambda(t \geq 30)$	0.5	-
$(p_b^n - p_v^n)$	$[0, 0, 0]^T$	m
$\hat{v}_b^n(0)$	$[0, 0, 0]^T$	m/s
$\hat{b}^b(0)$	$[0, 0, 0]^T$	$\mathrm{m/s}^2$
$\hat{\lambda}(0)$	1	-

(b) State and observer initial conditions.

Parameter	Value	Units
g^n	$[0, 0, 9.81]^T$	$\mathrm{m/s}^2$
$\omega^b(0)$	$[0, 0, 0]^T$	rad/s
$R_n^b(0)$	I	-
$\eta_{nv}(t < 30)$	$[20, 40, 60]^T$	deg
$\eta_{nv}(t \geq 30)$	$[-60, -20, 70]^T$	deg
$\hat{p}_b^n(t_{\mathrm{ref}} = 0)$	$p_b^n(t = 0)$	m
$\hat{p}_b^n(t_{\mathrm{ref}} = 30)$	$\hat{p}_b^n(t = 30)$	m

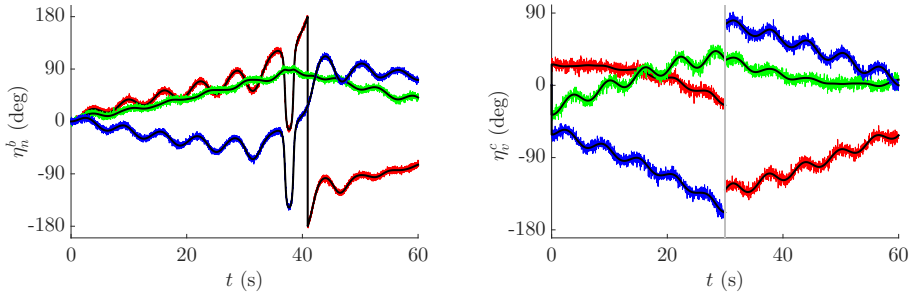
(c) Miscellaneous parameters.

Table 5.1: Parameters used in the VSE simulation.



(a) Specific force in $\mathcal{B}$

(b) VSLAM output - scaled position



(c) Vehicle attitude in $\mathcal{N}$

(d) Camera attitude in $\mathcal{V}$

Figure 5.3: The measured signals from the VISLAM simulation. The actual signal is black and the measured noisy and biased signal is in red, green and blue. At time $t = 30$ s the VSLAM system was forced to reset.

State	**EKF**	**KF**	**AO**
p_b^n (m)	0.50	0.31	**0.29**
v_b^n (m/s)	0.32	0.21	**0.19**
b^b (m/s^2)	**0.09**	0.11	0.13
λ	0.60	0.52	**0.47**

Table 5.2: RMSE of state trajectory for the VSE simulation.

estimate for $(p_b^n - p_v^n)$. It is important to note that the translational subsystem relies on a convergent estimate of the scale factor. Finally, the root mean square error (RMSE) for the states and the different observers over the entire simulation is given in Table 5.2. It also shows that the AO and the KF outperform the EKF.

5.5.2 Summary

Compared to the typical EKF-based approach our approach demonstrated improved performance.

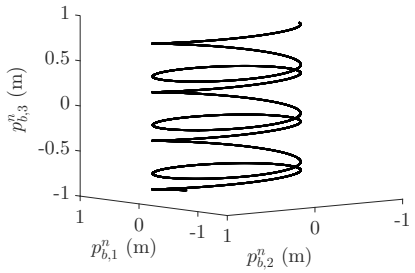


Figure 5.4: 3 D position trajectory of p^n_b for the VISLAM simulation.

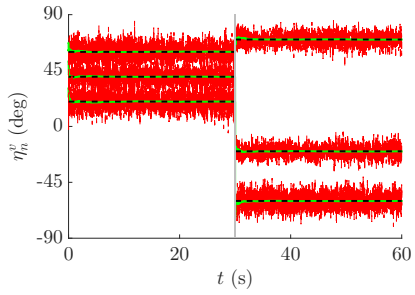
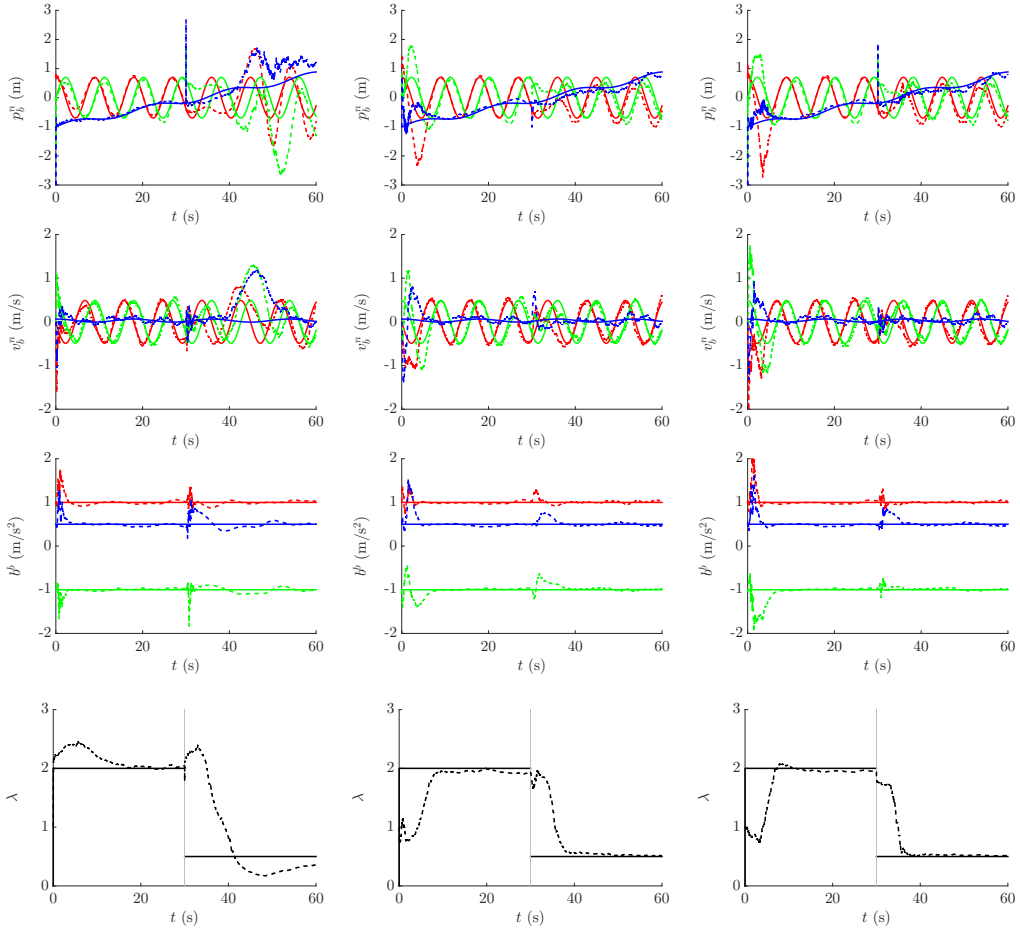
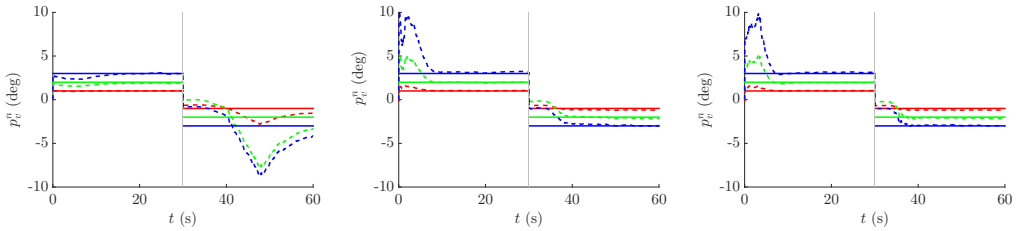


Figure 5.5: The Rotation between the navigation frame $\mathcal{N}$ and the vision frame $\mathcal{V}$. The actual signal is blue, the measured signal is red, and the output of the low pass filter is in green.



(a) EKF (b) KF (c) AO

Figure 5.6: Simulated position, velocity, bias, and scale trajectories. The actual trajectories are denoted by solid lines and the estimated trajectories are dashed.



(a) EKF (b) KF (c) AO

Figure 5.7: Simulated trajectories for position offset p_v^n from the vision frame $\mathcal{V}$ to the navigation frame $\mathcal{N}$. The actual position offset is denoted by solid lines and the estimated offset is dashed.

	KF, EKF	**AO**
Q	$\text{blkdiag}(0, 0, 10^6 \cdot I, 5 \times 10^5)$	$\text{blkdiag}(0, 0, 10^6)$
R	$\text{diag}(4 \times 10^3, 4 \times 10^3, 28)$	$\text{diag}(4 \times 10^3, 4 \times 10^3, 28)$
Γ	-	100
Σ	-	$7 \times 10^3 \cdot R^{-1}$

Table 5.3: Experimental observer parameters for hand held VSE experiment.

5.6 Handheld Experiment

5.6.1 Experimental Results

Using the computer vision system shown in Figure 3.9 we manually moved the box with the camera and IMU around the lab exciting all three axes of the accelerometer. To simulate a UAV flying over a city we placed a play rug of an overhead view of a city onto the laboratory floor. Table 5.3 provides the observer parameters. A screen shot of the camera view during the experiment is shown in Figure 5.8. The coloured points shown in Figure 5.8 are 3 D map points being tracked in the current image. In Figure 5.9 the scaled 3 D map created by PTAM during the experiment is shown. The coloured points are the 3 D map point being tracked and the coordinate axes represent the pose of the keyframes. The acceleration as measured by the vehicle in $\mathcal{B}$ during the experiment is shown in Figure 5.10. The observer estimates are shown in Figure 5.11. The figure shows the AO and KF have convergent position, velocity, and bias estimates. The EKF diverges and no noise parameters could be found that ensure convergence. It is interesting to note the bias estimates for all observers are not constant as predicted by our model (5.3). This is due to a misalignment of the IMU to $\mathcal{B}$ which introduces an error in f_s^b. We observed the KF and AO are robust to this model error whereas the EKF was not.

Figure 5.8: A screen shot of the camera view during the experiment. The coloured points are 3 D points from the map that are being tracked in the image.

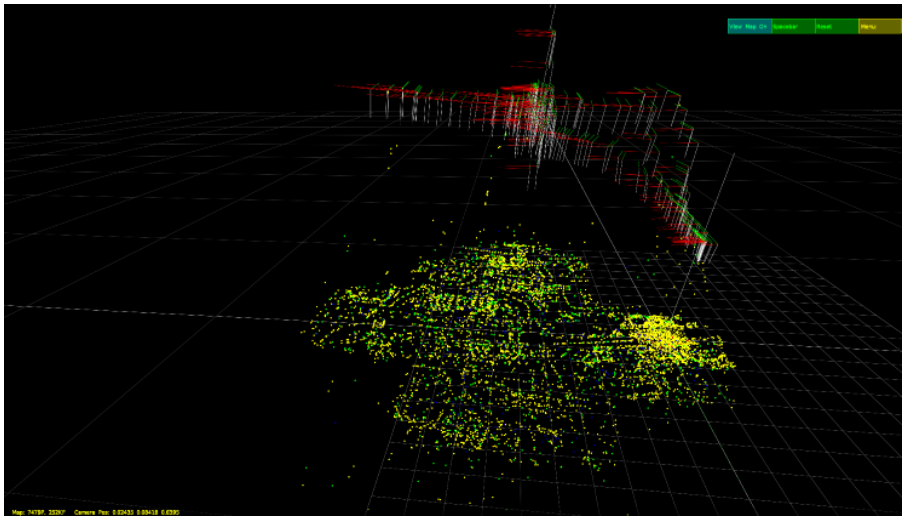


Figure 5.9: A scaled 3 D map created by PTAM during the experiment. The coloured points are 3 D map point being tracking and the red axes represent keyframe poses.

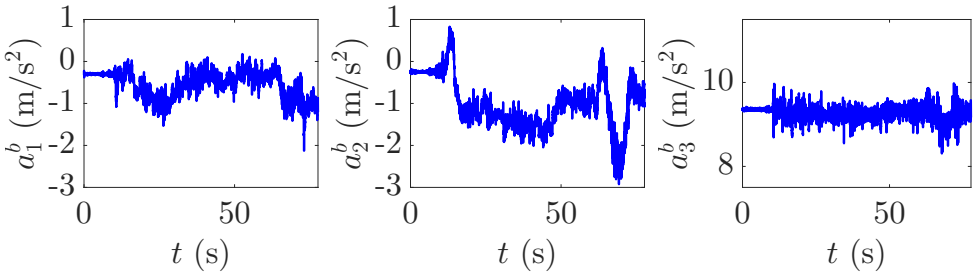


Figure 5.10: Vehicle acceleration during the VISLAM experiment.

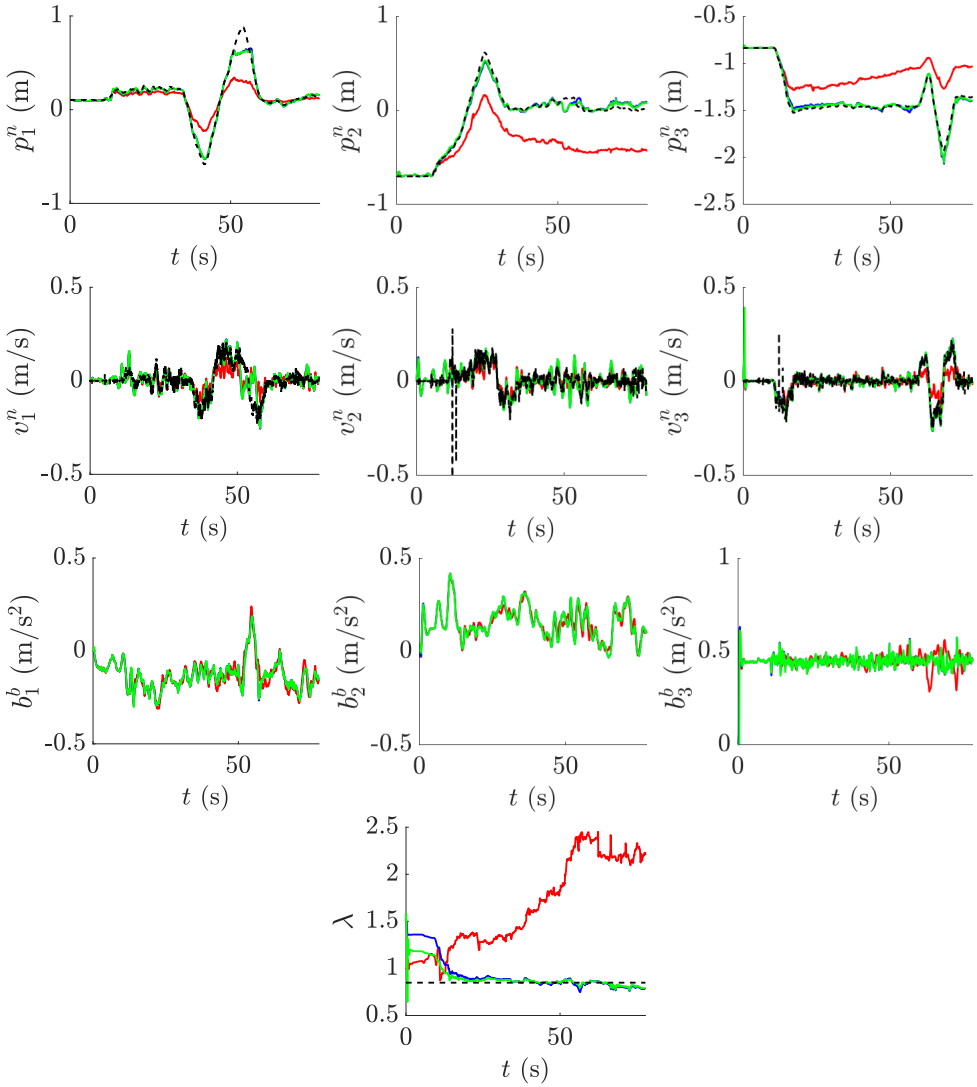


Figure 5.11: State estimates for AO (green), KF (blue), and EKF (red). MCS ground truth is in black.

Observer	Gains
EKF	$R^{-1} = 50I$, $Q = \text{blkdiag}\,(0, 0.01I, 0.05I, 0)$
KF	$R^{-1} = 100I$, $Q = \text{blkdiag}\,(0, 0.01I, 0.05I, 0)$
AKG	$R^{-1} = 1000I$, $Q = \text{blkdiag}\,(0, 0.01I, 0.05I)$,
	$\Sigma = 100I$, $\Gamma = 2$
ACG	$K = [50I, 25I, 32I]^T$, $\Sigma = 100I$, $\Gamma = 36$

Table 5.4: Observer parameters for VSE experiment.

5.6.2 Summary

In this section we presented the experimental validation of the proposed observers including a comparison with an EKF. Results showed the proposed observers converged in face of errors in f_s^b. The experiment showed the EKF did not converge due to this error.

5.7 Flight Experiment

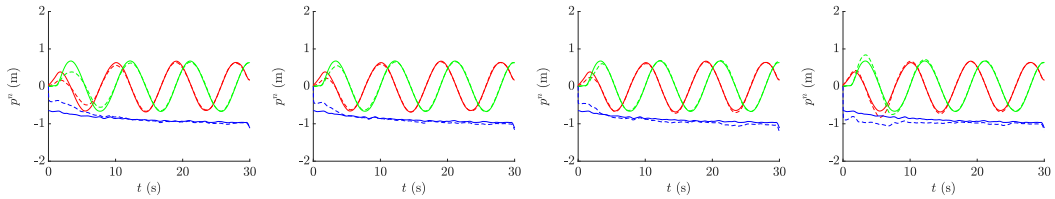
5.7.1 Experimental Results

In experiment we examined the scenario of a quadrotor flying in a circle trajectory:

$$p^{n*}(t) = \begin{bmatrix} a\cos(wt) \\ a\sin(wt) \\ -1 \end{bmatrix}^T$$

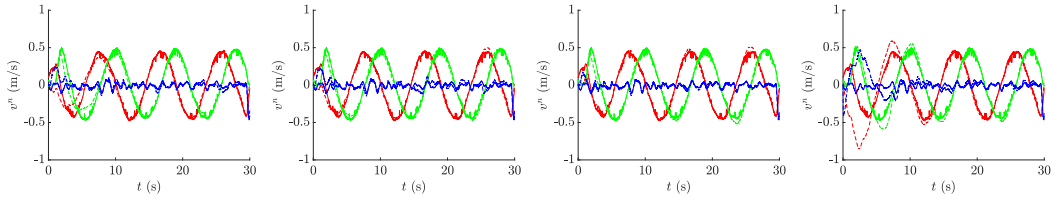
where $p^{n*}(t)$ is the desired position of the quadrotor in $\mathcal{N}$, $a = 0.55$ m and $w = 0.70$ Hz. This trajectory was chosen as it is a simple pattern that excites the accelerometer in two axis, but has zero acceleration in the third. The experiments compared four observers: an EKF (5.17) in the original coordinates, an KF (5.21) in the z coordinates, and two adaptive observers (5.33). One with a constant gain (ACG) and one with a Kalman gain (AKG). The observer parameters used in the experiment are shown in Table 5.4. All of the observers started with an initial estimate

$$\hat{x}(0) = \begin{bmatrix} \hat{p}^n(0) \\ \hat{v}^n(0) \\ \hat{b}^b(0) \\ \hat{\lambda}(0) \end{bmatrix} = \begin{bmatrix} [0,0,0]^T \\ [0,0,0]^T \\ [-0.5,0,0.5]^T \\ 1 \end{bmatrix}$$



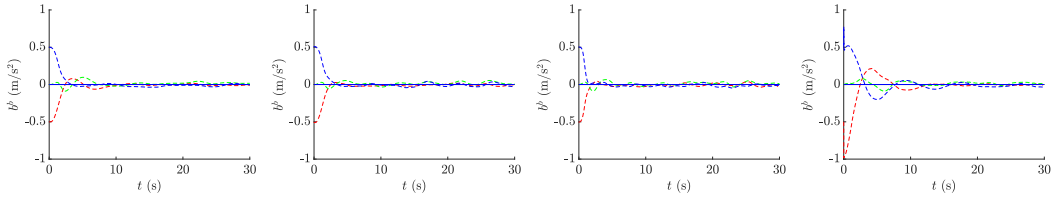
(a) EKF (b) KF (c) AKG (d) ACG

Figure 5.12: Position trajectories. The actual position is solid and estimated is dashed.



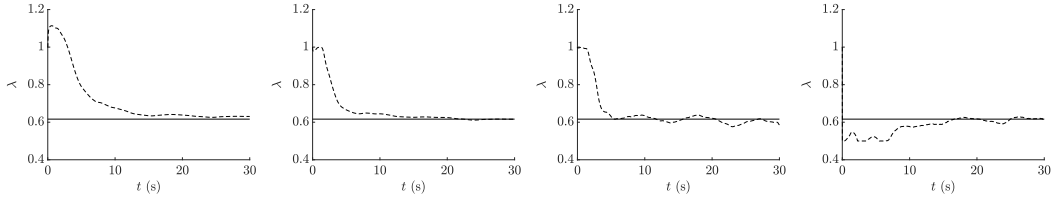
(a) EKF (b) KF (c) AKG (d) ACG

Figure 5.13: Velocity trajectories. The actual velocity is solid and estimated is dashed.



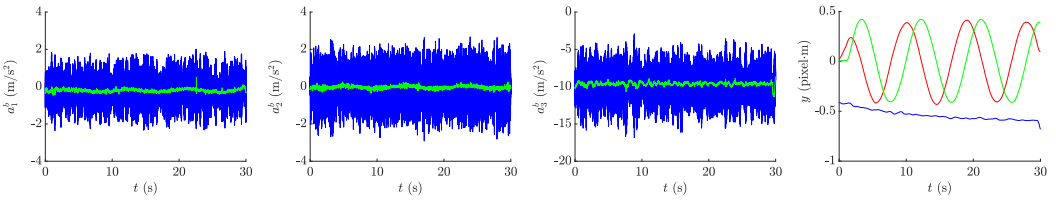
(a) EKF (b) KF (c) AKG (d) ACG

Figure 5.14: Bias trajectories. The actual bias is solid and estimated is dashed.



(a) EKF (b) KF (c) AKG (d) ACG

Figure 5.15: Scale trajectory. The actual scale is solid and estimated is dashed.



(a) Specific Force (b) Specific Force (c) Specific Force (d) Scaled Output

Figure 5.16: The input (specific force) and output (scaled position) of the simulated quadrotor dynamics. The raw accelerometer values are in blue and the filtered values are in green.

Using the Applied Nonlinear Control Lab (ANCL) quadrotor platform we flew the quadrotor in a circle trajectory. To simulate a UAV flying over a city we placed a play rug of an overhead view of a city onto the laboratory floor. A screen shot of the camera view during the experiment is shown in Figure 5.17. The green crosses shown in the figure are the 2 D image features being tracked in the current image. Figure 5.16 shows the specific force and scaled position measured during the experiment. Specific force was low-pass filtered before being used in the observer designs. Figure 5.12–Figure 5.15 show the trajectories of the state estimate for the four observers: position, velocity, bias, and scale, respectively. The figures show all four observers have convergent states estimates. In general all of the observers had slow, but acceptable, convergence of the VSLAM scale. The convergence rate can be increased with larger gains, but the observers become less robust. The RMSE for the states over the entire experiment is shown in Table 5.5. The table shows improved performance of the proposed observers over the existing EKF. The RMSE position error between VSLAM system using the actual scale and the MCS is 0.1 mm for the entire flight. We note that the actual accelerometer bias and VSLAM scale are not known. A pre-calibration was performed on the accelerometer to remove a constant bias and is needed to run the PX4's AHRS. This pre-calibration is not prefect and does not account for the turn on bias. Hence, we assume for these experiments the bias is $b^b \approx [0, 0, 0]^T$. The actual scale was calculated numerically as the scale value that minimizes the position error between the MCS and the VSLAM system. We can calculate the rotation between the navigation frame $\mathcal{N}$ and the vision frame $\mathcal{V}$ algebraically as follows

$$R_v^n = R_b^n R_c^b R_c^{vT} \tag{5.35}$$

where R_c^b is the rotation from $\mathcal{C}$ to $\mathcal{B}$, and R_c^v is the rotation from $\mathcal{C}$ to $\mathcal{V}$. The onboard AHRS measures R_b^n and the VSLAM system measures R_c^v. We roughly estimated

$$R_c^b = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In Figure 5.18 we see the estimated value of R_n^v expressed using ZYX Euler angles η_{nv}. The dashed line was calculated using (5.35) and the solid line was calculated using the nonlinear least squares. There is an approximately $[12, 10, 7]^T$ degree difference between the two estimates. This difference can be explained by small errors in the inter-sensor calibrations. Inter-sensor calibration can be improved by using methods such as [104].

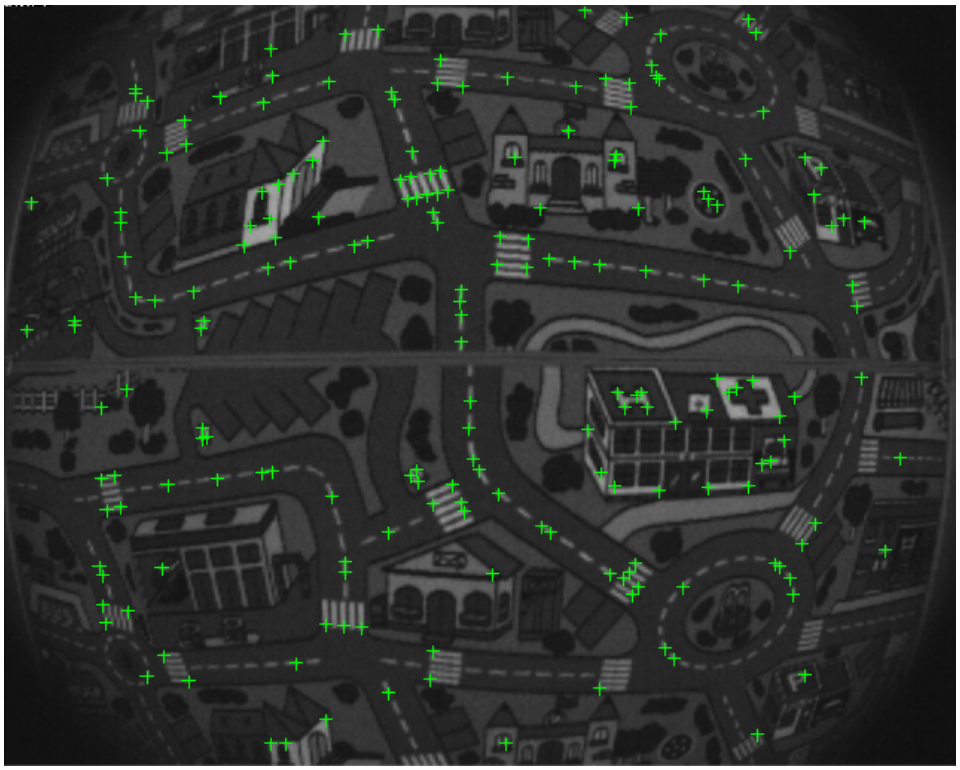


Figure 5.17: Sample image from the flight. The highlighted points are the 2 D image features being tracked.

State	EKF	KF	AKG	ACG
p^n (m)	0.09	0.07	0.06	0.09
v^n (m/s)	0.08	0.07	0.05	0.15
b^b (m/s^2)	0.11	0.11	0.07	0.16
λ	0.19	0.15	0.12	0.06

Table 5.5: RMSE of state estimate error for the VSE experiment

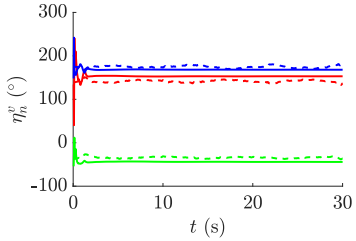


Figure 5.18: The rotation R_n^v between the navigation frame $\mathcal{N}$ and the vision frame $\mathcal{V}$. The values are calculated using nonlinear least square (solid) and algebraic (dashed) methods.

Observer	**EKF**	**KF**	**AKG**	**ACG**
States	10	10	9	9
Parameters	0	0	1	1
Covariance Matrix	10×10	10×10	9×9	-
Υ	0	0	9	9
Total	110	110	100	19
Minimum Representation	65	65	64	19

Table 5.6: Number of states per VSE observer.

5.7.2 Summary

Experimental results of the observers showed accurate position and velocity estimates even when processing noisy flight data.

5.8 Conclusion

In this chapter we presented an observability analysis for the VISLAM problem and proposed an adaptive observer. We fused the scaled position measurement from a VSLAM system with IMU measurements. We then introduced a coordinate transformation which put the system into a LTV coordinates. We presented conditions for the stability of the adaptive observer given in terms of UCO. Compared to the typical EKF-based approach our approach demonstrated improved performance, did not require linearization of the state equations, and we provided a stability proof for the observer. An important difference between the four observers is the numbers of states required. For the EKF and KF there are ten states for the dynamics plus a $n \times n$ covariance matrix. Due to the covariance matrix being symmetric is can be represented by $n(n+1)/2$ states. The adaptive observers have one less state and one parameter. They require an extra n states due to Υ. Additionally the adaptive observer with a Kalman gain has a covariance matrix. A summary of the number of states is shown in Table 5.6. Experimental results of the observers showed accurate position and velocity estimates even when processing noisy flight data.

Chapter 6

Conclusion

6.1 Conclusions

With the goal of creating a fully autonomous unmanned aerial vehicle (UAV), we created the Applied Nonlinear Control Lab (ANCL) quadrotor platform. The quadrotor can be used for any number of applications such as surveillance or infrastructure inspection. This thesis helps develop some of the tools necessary for almost all autonomous missions. When global navigation satellite system (GNSS) is not available, computer vision (CV) is one of the best options for stable flight. CV can provide accurate state estimation for both position-based visual servoing (PBVS) and image-based visual servoing (IBVS).

In this thesis we presented the ANCL quadrotor platform. It has been designed with both open source and open hardware concepts in mind. Like most modern hardware platforms it is in a constant state of change as newer technology emerges. Increasingly more complicated CV algorithms and more advanced control strategies are being made possible onboard UAVs. It has proven itself to be a powerful research platform in nonlinear controls and CV as it has been used in many published experiments. Furthermore, it has been an essential education tool for understanding CV and control theory. An important aspect of this work is that all of the algorithms were validated in experiment whereas most of the work in the literature on visual servoing (VS) has not.

In this thesis we investigated the control of a quadrotor using nonlinear dynamic image-based visual servoing (DIBVS) to regulate the lateral position of the vehicle relative to a static and moving visual feature point on the ground. All of the control laws are *dynamic* in that they directly account for the vehicles dynamics. They also share the same idea of rewriting the dynamics into a form where the design is simplified. Our approach is validated experimentally which demonstrates the method is robust to model uncertainty (e.g., errors in modelling aerodynamic forces). When

compared to other DIBVS control laws in the literature the proposed control laws demonstrate similar performance and benefit from reduced computational complexity. This is an important attribute for onboard implementation with inexpensive microcontrollers.

Lastly, in this thesis we presented an observability analysis for the visual inertial simultaneous localization and mapping (VISLAM) problem and proposed an adaptive observer. We fused the scaled position measurement from a visual simultaneous localization and mapping (VSLAM) system with inertial measurement unit (IMU) measurements. We then introduced a coordinate transform that made the system linear time-varying (LTV). We presented conditions for the stability of the adaptive observer given in terms of uniform complete observability (UCO). Compared to the typical extended Kalman filter (EKF)-based approach our approach demonstrated improved performance, did not require linearization of the state equations, and provided a stability proof for the observer. An important difference between the four observers is the numbers of states required. For the EKF and Kalman filter (KF) there are ten states for the dynamics plus a $n \times n$ covariance matrix. Due to the covariance matrix being symmetric is can be represented by $n(n+1)/2$ states. The adaptive observers have one less state and one parameter. They require an extra n states due to Υ. Additionally the adaptive observer with a Kalman gain has a covariance matrix. Experimental results of the observers showed accurate position and velocity estimates even when processing noisy flight data.

6.2 Future Work

There is still a lot of research left to be done in CV for UAVs. The algorithms developed throughout this thesis are the basic building blocks for many UAV missions. However, there are a few remaining challenges before they become widely used in industry. First and foremost is the reliability and robustness of CV tracking algorithms. There is no perfect tracking algorithm and the aviation industry has very high safety standard for UAVs. If CV is being used to follow a target and the trackers loses the target then the UAV can land and there are not a lot of safety concerns. However, if the UAV is using CV for its state estimate and then it loses tracking it can cause flyways or crashes. Another problem with CV happens with occlusion. It is relatively easy to have a third party object come between the target/environment and the camera. This is also almost entirely out of the control of the UAV. Thirdly, trackers rely on the target being tracked to be visually constant. This means the object can not be transparent, change in shape, size or colour. Due to varying light conditions the colour of the target is constantly changing. Lastly, a lot of CV rely on a static environment. For example, with VSLAM it is not possible

over a busy street or water as most of the image will not be static, hence almost all of the points being tracked in the image will be noise. Despite these challenges the future of CV looks promising and is always improving.

The IBVS control laws presented in this thesis are sound building blocks for more intricate control schemes. Current research is moving towards auxiliary topics such as input saturation or output feedback. Other current area of research are optical flow algorithms and sensorless velocity output feedback. This is because the current IBVS laws require a linear velocity estimate. Additional research is also being done on more complicated missions such as coordinated flight or slung loads.

VSLAM methods are continuously being improved and can offer up to centimetre accuracy. Algorithms are now able to handle much larger areas with faster camera movement. While VSLAM methods have made great improvements they still suffer from the same problems as described above with CV. VISLAM methods obviously rely on good quality measurements from the VSLAM methods. Sensor fusion with multiple sensors are making VISLAM methods more robust. Future research will also focus on tightly coupled methods which greatly increases the complexity of the algorithm, but can also account for all of the correlations between the sensors and thus increase the accuracy.

Bibliography

[1] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, Mar./Apr. 2012.

[2] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for UAVs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, Jul. 2017.

[3] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[4] O. Shakernia, Y. Ma, T. J. Koo, and S. Sastry, "Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control," *Asian Journal Control*, vol. 1, no. 3, pp. 128–145, Sep. 1999.

[5] E. Altug, J. Ostrowski, and C. Taylor, "Control of a quadrotor using dual camera visual feedback," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 3, Taipei, Taiwan, Sep. 2003, pp. 4294–4299.

[6] A. D. Wu, E. N. Johnson, and A. A. Proctor, "Vision-aided inertial navigation for flight control," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 348–360, 2005.

[7] L. Mejias, P. Campoy, S. Saripalli, and G. S. Sukhatme, "A visual servoing approach for tracking features in urban areas using an autonomous helicopter," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 2503–2508.

[8] S. Azrad, F. Kendoul, and K. Nonami, "Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm," *Journal of System Design and Dynamics*, vol. 4, no. 2, pp. 255–268, 2010.

[9] L. Garcia Carrillo, E. Rondon, A. Sanchez, A. Dzul, and R. Lozano, "Stabilization and trajectory tracking of a quad-rotor using vision," *Journal of Intelligent Robotic Systems*, vol. 61, no. 1, pp. 103–118, 2011.

[10] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a

quadrotor MAV," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, Oct. 2012, pp. 4557–4564.

[11] I. Sa and P. Corke, "Close-quarters quadrotor flying for a pole inspection with position based visual servoing and high-speed vision," in *Proceedings of the 2014 International Conference on Unmanned Aircraft Systems*, May 2014, pp. 623–631.

[12] B. Tamadazte, E. Marchand, S. Dembélé, and N. L. Fort-Piat, "CAD model-based tracking and 3D visual-based control for MEMS microassembly," *The International Journal of Robotics Research*, vol. 29, no. 11, pp. 1416–1434, 2010.

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

[14] S. Lin, M. A. Garratt, and A. J. Lambert, "Monocular vision-based real-time target recognition and tracking for autonomously landing an uav in a cluttered shipboard environment," *Autonomous Robots*, vol. 41, no. 4, pp. 881–901, Apr. 2017.

[15] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, 2008.

[16] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part I: The first 30 years and fundamentals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[17] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part II: Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, Jun. 2012.

[18] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, Dec. 2015.

[19] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[20] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[21] J. Neira, A. J. Davison, and J. J. Leonard, "Guest editorial special issue on visual SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 929–931, Oct. 2008.

[22] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[23] J. Civera, A. J. Davison, and J. M. Martínez Montiel, *Structure from Motion using the Extended Kalman Filter*. Springer Berlin Heidelberg, 2012, vol. 75.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 2007 IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 225–234.

[25] J. Civera, A. Davison, and J. Montiel, "Inverse depth parametrization for monocular SLAM," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 932–945, Oct. 2008.

[26] T. Pire, T. Fischer, G. Castro, P. DeCristforis, J. Civera, and J. JacoboBerlles, "S-PTAM: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, no. Supplement C, pp. 27 – 42, 2017.

[27] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

[29] R. Mur-Artal and J. D. Tards, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[31] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, Sep. 2009.

[32] A. Quattrini Li, A. Coskun, S. M. Doherty, S. Ghasemlou, A. S. Jagtap, M. Modasshir, S. Rahman, A. Singh, M. Xanthidis, J. M. O'Kane, and I. Rekleitis, *Experimental Comparison of Open Source Vision-Based State Estimation Algorithms*. Cham: Springer International Publishing, 2017, pp. 775–786.

[33] M. Bryson and S. Sukkarieh, *Inertial Sensor-Based Simultaneous Localization and Mapping for UAVs*. Dordrecht: Springer Netherlands, 2015, pp. 401–431.

[34] S. M. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 2012.

[35] S. Leutenegger, "Design and algorithms for efficient and robust autonomous operation," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 2014.

[36] A. Concha, G. Loianna, V. Kumar, and J. Civera, "Visual-inertial direct SLAM," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation*, Stockholm, Sweeden, May 2016.

[37] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," in *Proceedings of the 2009 IEEE and ACM International Symposium on Mixed and Augmented Reality*, Oct. 2009, pp. 145–151.

[38] T. Dick, C. Perez, M. Jagersand, and A. Shademan, "Realtime registration-based tracking via approximate nearest neighbour search," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, Jun. 2013.

[39] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 187–198, 2002.

[40] ——, "Image based visual servo control for a class of aerial robotic systems," *Automatica*, vol. 43, no. 11, pp. 1975–1983, Nov. 2007.

[41] F. L. Bras, R. Mahony, T. Hamel, and P. Binetti, "Dynamic image-based visual servo control for an aerial robot: Theory and experiments," *International Journal of Optomechatronics*, vol. 2, no. 3, pp. 296–325, 2008.

[42] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3D Vision: From Images to Geometric Models.* New York, NY: Springer-Verlag, 2003.

[43] N. Metni and T. Hamel, "Visual tracking control of aerial robotic systems with adaptive depth estimation," *International Journal of Control, Automation and Systems*, vol. 5, no. 1, pp. 51–60, 2007.

[44] H. de Plinval, P. Morin, P. Mouyon, and T. Hamel, "Visual servoing for underactuated VTOL UAVs: a linear, homography-based framework," *International Journal of Robust. Nonlinear Control*, vol. 24, no. 16, pp. 2285–2308, Apr. 2013.

[45] R. Ozawa and F. Chaumette, "Dynamic visual servoing with image moments for an unmanned aerial vehicle using a virtual spring approach," *Advanced Robotics*, vol. 27, no. 9, pp. 683–696, 2013.

[46] D. Lee, H. Lim, H. Kim, Y. Kim, and K. Seong, "Adaptive image-based visual servoing for an underactuated quadrotor system," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1335–1353, 2012.

[47] H. Jabbari Asl, G. Oriolo, and H. Bolandi, "An adaptive scheme for image-based visual servoing of an underactuated UAV," *International Journal of Robotics and Automation*, vol. 29, no. 1, pp. 92–104, 2014.

[48] H. Jabbari Asl and J. Yoon, "Robust image-based control of the quadrotor unmanned aerial vehicle," *Nonlinear Dynamics*, vol. 85, no. 3, pp. 2035–2048, Aug. 2016.

[49] H. Xie, "Dynamic visual servoing of rotary wing unmanned aerial vehicles," Ph.D. dissertation, University of Alberta, Edmonton, AB, Feb. 2016.

[50] H. Xie, K. H. Low, and Z. He, "Adaptive visual servoing of unmanned aerial vehicles in gps-denied environments," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2554–2563, Dec. 2017.

[51] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," *Computing Research Repository*, vol. abs/1703.05884, 2017. [Online]. Available: http://arxiv.org/abs/1703.05884

[52] P. Liang, Y. Wu, and H. Ling, "Planar object tracking in the wild: A benchmark," *Computing Research Repository*, 2017. [Online]. Available: http://arxiv.org/abs/1703.07938

[53] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[54] G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Nonlinear dynamic visual servoing of a quadrotor," *Journal of Unmanned Vehicle Systems*, vol. 3, no. 1, pp. 1–21, 2015.

[55] ——, "Experimental validation of dynamic visual servoing for a quadrotor using a virtual camera," in *Proceedings of the 2015 International Conference on Unmanned Aircraft Systems*, Denver, CO, Jun. 2015, pp. 1231–1240.

[56] H. Xie, G. Fink, A. F. Lynch, and M. Jagersand, "Adaptive dynamic visual servoing of a UAV," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2529–2538, 2016.

[57] G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Dynamic visual servoing for a quadrotor using a virtual camera," *Unmanned Systems*, vol. 5, no. 1, pp. 1–17, 2017.

[58] G. Fink, M. Franke, and A. F. Lynch, "Visual inertial slam: Application to unmanned aerial vehicles," in *Proceedings of the 2016 Unmanned Systems Canada Conference*, Edmonton, AB, Nov. 2016, pp. 1–6, student paper award.

[59] G. Fink, M. Franke, A. F. Lynch, K. Röbenack, and B. Godbolt, "Visual inertial SLAM: Application to unmanned aerial vehicles," in *Procedings of the 20th IFAC World Congress*, Toulouse, France, Jul. 2017, pp. 2001–2006.

[60] ——, "Observer design for visual inertial SLAM scale on a quadrotor UAV," in *Proceedings of the 2017 International Conference on Unmanned Aircraft Systems*, Miami, FL, Jun. 2017, pp. 830–839.

[61] G. Fink, M. Franke, A. F. Lynch, and K. Röbenack, "Observer design for monocular visual inertial SLAM," *Automatisierungstechnik*, vol. 66, no. 3, pp. 246–257, Mar. 2018.

[62] G. Fink, A. Othmane, M. Konz, D. Kastelan, and A. F. Lynch, "Motion control of unmanned aerial vehicles (UAV's) with time delay compensation," in *Proceedings of the 2015 Unmanned Systems Canada Conference*, Halifax, NS, Nov. 2015, student paper finalist.

[63] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modelling and Control*. New York, NY: Wiley, 2006.

[64] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control Engineering Practice*, vol. 19, no. 9, pp. 1023–1036, 2011.

[65] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Nonlinear control of VTOL UAVs incorporating flapping dynamics," in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Nov. 2013, pp. 2419–2425.

[66] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole Polytechnique Federale de Luasanne, Lausanne, Switzerland, 2007.

[67] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*. New Jersey, USA: Princeton University Press, 2012.

[68] M. Bangura, M. Melega, R. Naldi, and R. Mahony, "Aerodynamics of rotor blades for quadrotors," *ArXiv e-prints*, p. 42, Jan. 2016. [Online]. Available: http://arxiv.org/abs/1601.00733

[69] P. Castillo, R. Lozano, and A. Dzul, *Modelling and control of mini flying machines*. New York City, USA: Springer-Verlag, 2005.

[70] R. Mahony, R. W. Beard, and V. Kumar, *Modeling and Control of Aerial Robots*. Cham: Springer International Publishing, 2016, pp. 1307–1334.

[71] B. Godbolt, "Experimental nonlinear control of a helicopter unmanned aerial vehicle (UAV)," Ph.D. dissertation, University of Alberta, Edmonton, AB, 2013.

[72] L. Meier, "PX4 autopilot," http://pixhawk.org/ [accessed 01 Jan 2018], Institute for Visual Computing, Swiss Federal Institute of Technology Zurich, 2016. [Online]. Available: http://pixhawk.org/

[73] F. Kendoul, I. Fantoni, R. Lozano *et al.*, "Asymptotic stability of hierarchical inner-outer loop-based flight controllers," in *Procedings of the 17th IFAC World Congress*, 2008, pp. 1741–1746.

[74] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., ser. Cambridge Books Online. Cambridge, England: Cambridge University Press, 2003, vol. 1.

[75] F. Devernay and O. Faugeras, "Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured enviroments," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, Aug. 2001.

[76] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, Aug. 2006.

[77] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Jun. 1994, pp. 593–600.

[78] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[79] E. Rosten and T. Drummond, *Machine Learning for High-Speed Corner Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[80] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[81] M. Agrawal, K. Konolige, and M. R. Blas, *CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 102–115.

[82] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, *Adaptive and Generic Corner Detection Based on the Accelerated Segment Test*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–196.

[83] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2564–2571.

[84] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, *BRIEF: Binary Robust Independent Elementary Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[85] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2548–2555.

[86] G. Levi and T. Hassner, "LATCH: Learned arrangements of three patch codes," in *Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision*, Mar. 2016, pp. 1–9.

[87] B. Herissé, T. Hamel, R. Mahony, and F. X. Russotto, "Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, Feb. 2012.

[88] I. Sa, S. Hrabar, and P. Corke, "Inspection of pole-like structures using a vision-controlled VTOL UAV and shared autonomy," in *Proceedings of the*

2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep. 2014, pp. 4819–4826.

[89] F. Chaumette and S. Hutchinson, "Visual servo control. II. Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[90] O. Bourquardez, R. Mahony, T. Hamel, and F. Chaumette, "Stability and performance of image based visual servo control using first order spherical image moments," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 2006, pp. 4304–4309.

[91] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 2, pp. 331–340, 2008.

[92] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control*, vol. 130, no. 1, pp. 1–12, 2007.

[93] H. de Plinval, P. Morin, P. Mouyon, and T. Hamel, "Visual servoing for underactuated VTOL UAVs: A linear, homography-based approach," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 3004–3010.

[94] O. Faugeras, Q.-T. Luong, and T. Papadopoulou, *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. Cambridge, MA, USA: MIT Press, 2001.

[95] S. Lupashin, A. Schollig, M. Hehn, and R. D'Andrea, "The flying machine arena as of 2010," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2970–2971.

[96] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[97] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, Sep. 2010.

[98] J. How, B. Bethihke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, Apr. 2008.

[99] L. Meier, "QGroundControl," http://www.qgroundcontrol.org/ [accessed 01 Jan 2018], Institute for Visual Computing, Swiss Federal Institute of Technology Zurich, 2017. [Online]. Available: http://www.qgroundcontrol.org/

[100] S. Kirby, "SimonK Firmware," https://github.com/sim-/tgy [accessed 01 Jan 2018], 2018. [Online]. Available: https://github.com/sim-/tgy

[101] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.

[102] "Pixy CMUcam5," http://www.cmucam.org/projects/cmucam5/wiki/ [Accessed Jan. 1, 2018], 2018. [Online]. Available: http://www.cmucam.org/projects/cmucam5/wiki/

[103] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[104] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 1280–1286.

[105] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," *Control Systems, IEEE*, vol. 21, no. 1, pp. 84–99, Feb. 2001.

[106] E. Delaleau and W. Respondek, "Lowering the orders of derivatives of controls in generalized state space systems," *Journal of Mathematical Systems, Estimation, and Control*, vol. 5, no. 3, pp. 1–27, 1995, (Summary: 375–378).

[107] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive, and Robust*. Hertfordshire, United Kingdom: Prentice Hall, 1995.

[108] A. Isidori, *Nonlinear Control Systems*. New York City, USA: Springer-Verlag, 1991.

[109] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[110] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 743–749, Jun. 2009.

[111] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1116–1127, Dec. 2005.

[112] ——, "Image moments: generic descriptors for decoupled image-based visual servo," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 2, Apr. 2004, pp. 1185–1190Vol.2.

[113] H. Xie and A. F. Lynch, "State transformation-based dynamic visual servoing for an unmanned aerial vehicle," *International Journal of Control*, vol. 89, no. 5, pp. 892–908, 2016.

[114] H. K. Khalil, *Nonlinear Systems*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[115] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, Jan. 2015.

[116] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa, "Stereo vision based indoor/outdoor navigation for flying robots," in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 3955–3962.

[117] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, Dec. 2014.

[118] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014, special Issue on Visual Control of Mobile Robots.

[119] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 3748–3754.

[120] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proceedings of the 2011 International Symposium of Robotic Research*, Flagstaff, AZ, 2011, pp. 235–252.

[121] A. Mourikis and S. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007, pp. 3565–3572.

[122] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, Oct. 1977.

[123] W. Rugh, *Linear System Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[124] B. D. O. Anderson and J. Moore, "Time-varying feedback laws for decentralized control," *IEEE Transactions on Automatic Control*, vol. 26, no. 5, pp. 1133–1139, Oct. 1981.

[125] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Transactions of the ASME, series D, Journal of Basic Engineering*, vol. 83, pp. 95–108, Mar. 1961.

[126] E. D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, 2nd ed. New York, NY: Springer-Verlag, 1998.

[127] S. Puntanen and G. P. H. Styan, "Historical introduction: Issai Schur and the early development of the Schur Complement," in *The Schur Complement and Its Applications*, F. Zhang, Ed. Boston, MA: Springer US, 2005, pp. 1–16.

[128] B. Ni and Q. Zhang, "Stability of the Kalman filter for continuous time output error systems," *Systems and Control Letters*, vol. 94, pp. 172–180, 2016.

[129] X. Li, Q. Zhang, and H. Su, "An adaptive observer for joint estimation of states and parameters in both state and output equations," *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 9, pp. 831–842, 2011.

[130] Q. Zhang, "Adaptive observer for multiple-input-multiple-output (MIMO) linear time-varying systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 3, pp. 525–529, Mar. 2002.

[131] B. D. O. Anderson, R. R. Bitmead, C. R. Johnson, Jr., P. V. Kokotovic, R. L. Kosut, I. M. Mareels, L. Praly, and B. D. Riedle, *Stability of Adaptive Systems: Passivity and Averaging Analysis*. Cambridge, MA, USA: MIT Press, 1986.

[132] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[133] R. Brockett, "4: Stability," in *Finite Dimensional Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1970, pp. 183–227.

[134] H. Strasdat, J. Montiel, and A. Davison, "Real-time monocular SLAM: Why filter?" in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010, pp. 2657–2664.

Appendix A

ANCL Platform Pinouts

Table A.1: Pixhawk Connector Pinouts

	Pin	Function	Description
*P*2 TELE 1&2	1	VCC_5V	5 V power supply
	2	TX	UART5 Console Port - Transmit Output
	3	RX	UART5 Console Port - Receive Input
	4	CTS	Clear to Send
	5	RTS	Ready To Send
	6	GND	Ground
*P*5 SPI	1	VCC_5V	5 V power supply
	2	SPI_EXT_SCK	Serial Clock
	3	SPI_EXT_MISO	sending data from a master to a slave
	4	SPI_EXT_MOSI	sending data from a slave to a master
	5	!SPI_EXT_NSS	!master-to-slave and slave-to-master permutation
	6	!GPIO_EXT	!General-purpose input/output
	7	GND	Ground
*P*6 PWR	1	VCC	5 V power supply
	2	VCC	5 V power supply
	3	CURRENT	3.3 V
	4	VOLTAGE	3.3 V
	5	GND	Ground
	6	GND	Ground
*P*8 ADC 3.3	1	VCC_5V	5 V power supply
	2	ADC IN	Up to 3.3v
	3	GND	Ground
	4	ADC IN	Up to 3.3v
	5	GND	Ground
*P*9 ADC 6.6	1	VCC_5V	5 V power supply
	2	ADC IN	Up to 6.6v
	3	GND	Ground
*P*10 I2C	1	VCC_5V	5 V power supply
	2	SCL	Serial Clock Line
	3	SDA	Serial Data Line
	4	GND	Ground

Continued on next page

Table A.1 – Pixhawk Connector Pinouts (Continued)

	Pin	Function	Description
$P11$ CAN	1	VCC_5V	5 V power supply
	2	CAN_H	CAN High
	3	CAN_L	CAN Low
	4	GND	Ground
$P12$ GPS	1	VCC_5V	5 V power supply
	2	TX	UART5 Console Port - Transmit Output
	3	RX	UART5 Console Port - Receive Input
	4	CAN2 TX	CAN Transmit Output
	5	CAN2 RX	CAN Receive Input
	6	GND	Ground
$P15$ SWITCH	1	VCC_5V	5 V power supply
	2	!IO_LED_SAFETY	
	3	SAFETY	Ground

Table A.2: PX4FMU Connector Pinouts

	Pin	Function	Description
$F1$ GPS	1	VDD_GPS (5V default)	5 V power supply
	2	USART6_TX	USART6 Console port - Transmit output
	3	USART6_RX	USART6 Console port - Receive input
	4	NC	Not connected
	5	GND	Ground
$F2$ Multi Port	1	VDD-5V	5 V power supply
	2	VDD-3V3	3 V power supply
	3	GPIO_EXT1 / AR.S1	General-purpose input/output,external interrupt??
	4	GPIO_EXT2 / AR.S2	General-purpose input/output,??
	5	BATTERY MONITOR (3-18V)	Battery monitor
	6	PPM_INPUT (3-5V)	Pulse Position Modulation Input
	7	USART1_RX	USART1 Console Port - Receive input
	8	USART1_TX	USART1 Console Port - Transmit output
	9	UART2_RX / SRV1 / AR.RX	UART2 Console Port - Recive Input / SERVO1 /
	10	USART2_RTS / SRV2 / AR.S3	USART2 Console Port - Ready To Send / SERVO2 /
	11	USART2_CTS / SRV3 / AR.S4	USART2 Console Port - Clear To Send / SERVO3 /
	12	USART2_TX / SRV4 / AR.TX	USART4 Console Port - Transmit Output / SERVO4 /
	13	I2C1_SDA	I2C1- Serial Data Line
	14	I2C1_SCL	I2C1- Serial Clock Line
	15	GND	Ground

Continued on next page

Table A.2 – PX4FMU Connector Pinouts (Continued)

	Pin	Function	Description
	1	VDD-5V	5 V power supply
	2	VDD-5V	5 V power supply
	3	GND	Ground
	4	GND	Ground
	5	CAN2_TX	Single ended TX output of CAN port 2
	6	CAN2_RX	Single ended RX input of CAN port 2
	7	USART1_TX	USART1 Console Port- Transmit Output
	8	USART1_RX	USART1 Console Port- Receive Input
	9	I2C3_SCL	I2C3- Serial Clock Line
	10	I2C3_SDA	I2C3- Serial Data Line
	11	ADC123_IN10	Analog to Digital Converter ?
	12	-	
	13	UART6_TX	USART6 Console Port- Transmit Output
	14	UART6_RX	USART5 Console Port - Receive Input
$F8$	15	UART5_TX	USART5 Console Port- Transmit Output
BUS	16	UART5_RX	USART5 Console Port - Receive Input
	17	I2C2_SCL	I2C2- Serial Clock Line
	18	I2C2_SDA	I2C2- Serial Data Line
	19	USART2_CTS	USART2 Console Port - Clear to send
	20	USART2_RTS	USART2 Console Port - Ready To Send
	21	USART2_TX	USART2 Console Port- Transmit Output
	22	USART2_RX	USART2 Console Port - Receive Input
	23	PPM_INPUT	Pulse Position Modulation Input
	24	GPIO_EXT1	General-purpose input/output?
	25	GPIO_EXT2	General-purpose input/output?
	26	BUZZER	
	27	GND	Ground
	28	ADC123_IN11	Analog to Digital Converter ?
	29	ADC123_IN12	Analog to Digital Converter ?
	30	ADC123_IN13	Analog to Digital Converter ?

Table A.3: PX4IO Connector Pinouts

	Pin	Function	Description
	1	VCC_5V	5 V power supply
$I1$	2	TX	UART5 Console Port - Transmit Output
FMU	3	RX	UART5 Console Port - Receive Input
UART5	4	NC	Not Connected
	5	GND	Ground
$I2$	1	VCC_5V	5 V power supply
FMU Pres	2	SIGNAL	
	3	GND	Ground
	1	VCC_5V	5 V power supply
$I3$	2	TX	USART2 Console Port - Transmit Output
FMU	3	RX	USART2 Console Port - Receive Input
USART5	4	NC	Not Connected
	5	GND	Ground
	1	VCC_5V	5 V power supply
$I4$	2	SCL	Serial Clock Line
FMU I2C3	3	SDL	Serial Data Line
	4	GND	Ground

Continued on next page

Table A.3 – PX4IO Connector Pinouts (Continued)

	Pin	Function	Discription
*I*5&6	1	C	
RL1&2	2	NO	
*I*7&8	1	VCC_5	5 v Power supply
AC1&2	2	GND	Ground
	1	VCC_5V	5 V power supply
	2	SPI2_SCK	SPI Serial Clock
	3	SPI2_MISO	sending data from the master to a slave
*I*9	4	SPI2_MOSI	sending data from the slave to a master
FMU SPI	5	SPI2_NSS	master-to-slave and slave-to-master permutation
	6	NC	Not connected
	7	GND	Ground
	1	VCC_3V	3 V power supply
*I*10	2	SPECTRUM	
SPEK IN	3	GND	Ground
	1	VCC_5V	5 V power supply
*I*11	2	CAN_L	
FMU CAN	3	CAN_H	
	4	GND	Ground
	1	VCC_5V	5 V power supply
*I*12	2	TX	USART2 Console Port - Transmit Output
IO	3	RX	USART2 Console Port - Receive Input
USART2	4	NC	Not connected
	5	GND	Ground
	1	TX _5V	
*I*14	2	RX	
BOOT	3	GND	Ground
	1	PPM/S.BUS IN	
*I*15	2	VCC_5V	5v Power supply
PPM	3	GND	Ground
	1	SERVO 1	
	2	SERVO 2	
	3	SERVO 3	
*I*16	4	SERVO 4	
SERVOS	5	SERVO 5	
	6	SERVO 6	
	7	SERVO 7	
	8	SERVO 8	
*I*17	1	VBAT	(6.3-18V)
BAT	2	GND	Ground
	1	S.BUS OUT _5V	
*I*18	2	VCC_5V	5v Power supply
SBUS	3	GND	Ground
*I*20	1	POSITIVE	
ALARM	2	NEGATIVE	
	1	VCC_3V3	
*I*21	2	LED3	
SFTY SW	3	SAFETY	

Table A.4: ANCL Custom Power Distribution Board Pinouts

	Pin	Function	Description
J1	1	VDD	Battery VDD
	2	GND	Ground
J2	1	VDD	Motor VDD
	2	GND	Ground
J3	1	VDD	5 V
	2	GND	Ground
J4	1	VDD	Battery VDD
	2	GND	Ground

Table A.5: Motor Power Distribution Board Connector Pinouts

	Pin	Function	Description
D1-D4	1	VDDS	12 V Power Supply
	2	GND	Ground
D5-D8	1	-	-
	2	-	-
	3	-	-
D10	1	PWM Signal	Motor 1 PWM Signal
	2	PWM Signal	Motor 2 PWM Signal
	3	PWM Signal	Motor 3 PWM Signal
	4	PWM Signal	Motor 4 PWM Signal

Table A.6: RM024 Pinouts

	Pin	Function	Description
R1 XBee Adapter Board	1	VCC_3.3 V	2.3 - 3.6 V ±50 mV ripple (must be connected)
	2	TXD	Asynchronous serial data output from transceiver
	3	RXD	Asynchronous serial data input to transceiver
	4	GIO_6	Reserved for future use. Do not connect.
	5	Reset	Module reset.
	6	GIO_1	Generic Output
	7	GIO_0	Generic Output / Hop_Frame
	8	DNC	Do Not Connect
	9	SLEEP	P9
	10	GND	Signal Ground
	11	P11	CMD/Data
	12	CTS	Clear to Send
	13	GIO_5	Reserved for future use. Do not connect.
	14	GIO_4	Generic Input
	15	P15	In Range
	16	RTS	Request to Send
	17	GIO_2	RS485 Driver Enable
	18	GIO_8	Generic Input
	19	GIO_3	PWM Output
	20	GIO_7	Analog to Digital input
R2	1	U.FL Connector	RF connector for high-frequency signals up to 6 GHz

Table A.7: RN-XV Pinouts

	Pin	Function	Description
R1 XBee Adapter Board	1	VDD_3V3	3.3V regulated power input to the module
	2	UART_TX	UART TX, 8mA drive, 3.3V tolerant
	3	UART_RX	UART RX, 3.3V tolerant
	4	GPIO 8	GPIO, 24mA drive, 3.3V tolerant
	5	RESET	Optional Module Reset Signal (active low), 100k Pull up, apply pulse of at least 160us, 3.3V Tolerant
	6	GPIO 5	GPIO, 24mA drive, 3.3V tolerant
	7	GPIO 7	GPIO, 24mA drive, 3.3V tolerant
	8	GPIO 9	Enable Adhoc mode, Restore factory defaults, 8mA drive, 3.3V tolerant
	9	GPIO 1	GPIO, 8mA drive, 3.3V tolerant
	10	GND	Ground
	11	GPIO 14	GPIO, 8mA drive, 3.3V tolerant
	12	UART_RTS	UART RTS flow control, 8mA drive, 3.3V tolerant
	13	GPIO 4/SEN 6	GPIO, 24mA drive, 3.3V tolerant/ADC input , (3.3V tolerant). Defaults to GPIO 4
	14	Not Used	
	15	GPIO 6/SEN 7	GPIO, 24mA drive, 3.3V tolerant/ADC input, (3.3V tolerant). Defaults to GPIO 6
	16	UART_CTS	UART CTS flow control, 3.3V tolerant
	17	SENSOR 5	Sensor Interface, Analog input to module, (3.3V tolerant)
	18	GPIO 3/SEN 4	GPIO, 8mA drive, 3.3V tolerant/ADC input (3.3V tolerant). Defaults to GPIO 3
	19	GPIO 2/SEN 3	GPIO, 8mA drive, 3.3V tolerant/ADC input (3.3V tolerant). Defaults to SEN 3
	20	SEN 2	Sensor Interface, Analog input to module, 3.3V tolerant

Table A.8: 3DR Radio Pinouts

	Pin	Function	Description
R1 TTL Pinouts	1	RTS	Request To Send Control Output / Handshake signal
	2	TX	Transmit Asynchronous Data output
	3	RX	Receive Asynchronous Data input
	4	VCC	5 V power supply
	5	CTS	Clear to Send Control input / Handshake signal
	6	GND	Ground
R2 Antenna	1	Antenna	U.FL Radio Connector

Table A.9: FTDI_PCB Pinouts

	Pin	Function	Description
P1 TTL Pinouts	1	VCC	3.3 V power supply
	2	TXD	Transmit Asynchronous Data output
	3	RXD	Receive Asynchronous Data input
P2 USB	1	GND	Ground
	2	USB1-D-	USB 2.0 data
	3	USB1-D+	USB 2.0 data
	4	5 V	5 V power

Continued on next page

Table A.9 – FTDI_PCB Pinouts (Continued)

	Pin	Function	Description
P3	1	RTS #	Request To Send Control Output / Handshake signal
TTL	2	GND	Ground
Pinouts	3	CTS #	Clear to Send Control input / Handshake signal

Table A.10: FTDI Pinouts

	Pin	Function	Description
P1 TTL Pinouts	1	GND	Ground (Black)
	2	CTS #	Clear to Send Control input / Handshake signal (Brown)
	3	VCC	3.3 V power supply (Red)
	4	TXD	Transmit Asynchronous Data output (Orange)
	5	RXD	Receive Asynchronous Data input (Yellow)
	6	RTS #	Request To Send Control Output / Handshake signal (Green)
P2 USB	1	5 V	5 V power
	2	USB1-D+	USB 2.0 data
	3	USB1-D-	USB 2.0 data
	4	GND	Ground