

**Autonomous Navigation and Tracking of Dynamic  
Surface Targets On-board a Computationally  
Impoverished Aerial Vehicle**

by

William Clayton Selby

B.S., United States Naval Academy (2009)

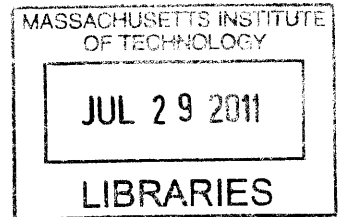
Submitted to the Department of Mechanical Engineering  
in partial fulfillment of the requirements for the degree of  
Masters of Science in Mechanical Engineering

at the

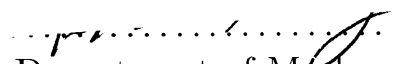
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011


© Massachusetts Institute of Technology 2011. All rights reserved.



**ARCHIVES**

Author .....   
Department of Mechanical Engineering  
May 6, 2011

Certified by .....  
Daniela L. Rus  
Professor of Electrical Engineering and Computer Science  
Thesis Supervisor

Certified by .....   
John J. Leonard  
Professor of Mechanical Engineering  
Mechanical Engineering Faculty Reader

Accepted by ..   
David E. Hardt  
Chairman, Department Committee on Graduate Students  
Mechanical Engineering Department

THIS PAGE INTENTIONALLY LEFT BLANK

# Autonomous Navigation and Tracking of Dynamic Surface Targets On-board a Computationally Impoverished Aerial Vehicle

by

William Clayton Selby

Submitted to the Department of Mechanical Engineering  
on May 6, 2011, in partial fulfillment of the  
requirements for the degree of  
Masters of Science in Mechanical Engineering

## Abstract

This thesis describes the development of an independent, on-board visual servoing system which allows a computationally impoverished aerial vehicle to autonomously identify and track a dynamic surface target. Image segmentation and target tracking algorithms are developed for the specific task of monitoring whales at sea. The computer vision algorithms' estimates prove to be accurate enough for quadrotor stabilization while being computationally fast enough to be processed on-board the platform. This differs from current techniques which require off-board processing of images for vehicle localization and control. The vision algorithm is evaluated on video footage to validate its performance and robustness.

The quadrotor is then modeled to motivate and guide the development of Linear Quadratic Regulator (LQR) controllers for maneuvering the quadrotor. The controllers are tuned using a motion capture system which provides ground truth state measurements. The vision system is integrated into the control scheme to allow the quadrotor to track an iCreate. Additionally, an Extended Kalman Filter (EKF) fuses the vision system position estimates with attitude and acceleration measurements from an on-board Inertial Measurement Unit (IMU) to allow the quadrotor to track a moving target without external localization.

Thesis Supervisor: Daniela L. Rus

Title: Professor of Electrical Engineering and Computer Science

Mechanical Engineering Faculty Reader: John J. Leonard

Title: Professor of Mechanical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK



## Acknowledgments

This thesis would not have been possible without the support of a large group of friends, family, and colleagues. I would primarily like to thank my advisor, Daniela Rus, whose seemingly unlimited energy kept me motivated and excited throughout this research. Her wide ranging interests in all aspects of robotics allowed me unique opportunities to apply this research to a multitude of related projects. I would also like to thank my computer vision mentor, Peter Corke. Peter's wide range and experience in visual servoing helped guide key design choices in vision as well as control. I am especially grateful for the time spent at his lab at QUT in Australia.

I am especially grateful to my fellow Distributed Robotics Lab colleagues who took time from their own research to take critical looks at mine. Daniel Soltero was invaluable as a second hand when running experiments. Kyle Gilpin and Marek Doniec provided very valuable hardware advice that saved countless hours of debugging. I would also like to thank Abe Bacharach of the Robust Robotics Group at MIT for access to several key software components for this research and his extensive quadrotor control knowledge. I am especially grateful to Brian Julian for his experience in dealing with the quadrotor platform as well as his large knowledge of control and estimation theory. I wish all my lab mates the most of luck in their future endeavors.

Finally, this work could not be done without the support of my family. The support of my mother Julie, father Doug and sister Taylor was invaluable and I appreciate it greatly.

THIS PAGE INTENTIONALLY LEFT BLANK

# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Contributions . . . . .	23
1.2	Thesis Overview . . . . .	24
<b>2</b>	<b>Related Work</b>	<b>27</b>
2.1	General Background . . . . .	27
2.2	Quadrotor Development and Applications . . . . .	28
2.3	Object Recognition . . . . .	30
2.4	Visual Control of Aerial Vehicles . . . . .	31
2.5	Context for This Research . . . . .	33
<b>3</b>	<b>Experimental Set-up</b>	<b>35</b>
3.1	Laboratory System Architecture . . . . .	36
3.1.1	Experimental Hardware . . . . .	37
3.1.2	Software Architecture . . . . .	39
3.2	Camera Model and Calibration . . . . .	40
3.2.1	Camera and Lens Selection . . . . .	41
3.2.2	Pinhole Camera Model . . . . .	42
3.2.3	Lens Distortions . . . . .	43
3.2.4	Camera Calibration . . . . .	45
3.2.5	Calibration Results . . . . .	47
3.2.6	Camera Settings Configuration . . . . .	49
3.3	Quadrotor Dynamic Model . . . . .	52

3.3.1	Model Adaption . . . . .	56
3.3.2	Parameter Estimation and Model Verification . . . . .	58
3.4	Extended Kalman Filter . . . . .	61
3.4.1	Process Model . . . . .	62
3.4.2	Measurement Model . . . . .	64
<b>4</b>	<b>Object Identification and Tracking</b>	<b>69</b>
4.1	Color Models . . . . .	70
4.2	Pixel Classification . . . . .	71
4.3	Image Segmentation and Object Description . . . . .	72
4.4	Tracking Multiple Objects . . . . .	75
4.5	Position Vector Calculation . . . . .	76
<b>5</b>	<b>Controller Design</b>	<b>79</b>
5.1	LQR Derivation . . . . .	79
5.2	LQR Implementation . . . . .	82
<b>6</b>	<b>Experimental Results</b>	<b>89</b>
6.1	Experimental Set-up, Methodology, and Metrics for Object Identification	90
6.2	Results of Object Identification Algorithms . . . . .	92
6.2.1	Tracking with an Overhead Perspective . . . . .	92
6.2.2	Tracking at an Angled Perspective . . . . .	93
6.2.3	Tracking from the Surface . . . . .	95
6.2.4	Performance Comparison of View Angle . . . . .	96
6.2.5	Boat Identification . . . . .	97
6.3	Experimental Set-up, Methodology, and Metrics for Visual Servoing .	98
6.4	Results of Visual Servoing Experiments . . . . .	100
6.4.1	Visual Servoing With Motion Capture Output . . . . .	101
6.4.2	Visual Servoing with EKF Output . . . . .	101
6.5	Experimental Set-up, Methodology, and Metrics for Dynamic Object Tracking . . . . .	102

6.6	Results of Dynamic Object Tracking Experiments . . . . .	103
6.6.1	Dynamic Object Tracking with Motion Capture Output . . .	103
6.6.2	Dynamic Object Tracking with EKF Output . . . . .	104
<b>7</b>	<b>Conclusions</b>	<b>107</b>
7.1	Lessons Learned . . . . .	108
7.2	Future Works . . . . .	109

THIS PAGE INTENTIONALLY LEFT BLANK

# List of Figures

1-1	Quadrotor Tracking a Dynamic Surface Target . . . . .	16
1-2	Various Forms of Currently Used Unmanned Aerial Vehicles . . . . .	17
1-3	The Ascending Technologies <i>Hummingbird</i> Quadrotor in Flight . . . . .	18
1-4	Sample Frame From Video Footage Recorded in Argentina . . . . .	20
1-5	Tracking of Projected Whale Video Footage . . . . .	22
3-1	Quadrotor Outfitted With fit-PC2 and Firefly MV2 . . . . .	35
3-2	Views of the iCreate Target Used for Hardware Experiments . . . . .	36
3-3	Experimental Network Topology . . . . .	37
3-4	Experimental Software Architecture . . . . .	39
3-5	Pinhole Camera Model . . . . .	42
3-6	Effects of Radial Distortion . . . . .	44
3-7	Effects of Tangential Distortion . . . . .	45
3-8	Checkerboard Images Used for Calibration . . . . .	47
3-9	Camera Centered View of Estimated Checkerboard Locations . . . . .	48
3-10	Raw Image Before and After Distortion Correction . . . . .	49
3-11	Effects of Default White Balance Values . . . . .	50
3-12	Effects of Corrected White Balance Values . . . . .	51
3-13	Quadrotor Structural Diagram . . . . .	52
3-14	Rotational Dynamics Block Diagram . . . . .	57
3-15	Translational Dynamics Block Diagram . . . . .	58
3-16	Comparison of Simulated Rotational Dynamics and Ground Truth . . . . .	59
3-17	Comparison of Simulated Translational Dynamics and Ground Truth . . . . .	60

3-18	Integration Errors Due to IMU Drift . . . . .	61
3-19	Comparison of Acceleration Estimates and Ground Truth Values . . .	64
3-20	Comparison of Altitude Estimates and Ground Truth Values . . . . .	66
4-1	Image Thresholding in the Hue and Saturation Planes . . . . .	71
4-2	Target Recognition Through Contour Identification . . . . .	73
4-3	Simultaneous Tracking of Multiple Targets . . . . .	76
4-4	Weighted vs. Unweighted Position Vector . . . . .	77
5-1	Simulink LQR Controller Block Diagram . . . . .	86
5-2	Simulink Feedback Block Diagram . . . . .	87
5-3	Simulated LQR Trajectory Tracking . . . . .	88
6-1	Sample Frame From Footage Recorded at an Overhead Perspective .	93
6-2	Sample Frame From Footage Recorded at an Angled Perspective . . .	94
6-3	Sample Frame From Footage Recorded at a Surface Perspective . . .	95
6-4	Sample Frame From Boat Footage . . . . .	98
6-5	Comparison of EKF Output to Ground Truth . . . . .	100
6-6	Visual Servoing Utilizing Motion Capture State Feedback . . . . .	101
6-7	Hovering Utilizing EKF State Estimation . . . . .	102
6-8	Quadrotor Target Tracking Ground Truth Positions Utilizing Motion Capture State Feedback . . . . .	103
6-9	Quadrotor Target Tracking Ground Truth Positions Utilizing EKF State Feedback . . . . .	105
6-10	Vision System Performance With EKF State Feedback . . . . .	105
7-1	Future Work Integrating Vision System With AMOUR AUV . . . . .	110



# List of Tables

3.1	Comparison of Camera Calibration Methods . . . . .	49
6.1	Properties of Video Clips Analyzed . . . . .	91
6.2	Vision System Results: Overhead Perspective . . . . .	93
6.3	Vision System Results: Angled Perspective . . . . .	94
6.4	Vision System Results: Surface Perspective . . . . .	96
6.5	Properties of Video Clips Analyzed . . . . .	97
6.6	Vision System Results: Boat Identification . . . . .	98
6.7	Dynamic Object Tracking: Motion Capture State Feedback . . . . .	104
6.8	Dynamic Object Tracking: EKF State Feedback . . . . .	106

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 1

## Introduction

We wish to develop an autonomous control system for a Micro Aerial Vehicle (MAV) which uses visual position estimates to identify and track surface targets without relying on external processing. MAVs are characterized by their small size and payload. Due to the payload constraints, on-board sensing and processing is limited. Such platforms could be used as “remote eyes” to collect data quietly and accurately from a distance. Applications include recording ecosystems without interfering with the behavior of animals, collecting data for situational awareness, or performing targeted surveillance.

### **Current UAVs and Their Applications**

Research and development in the field of Unmanned Aerial Vehicles (UAVs) has been progressing at an astonishing pace. The availability to own and operate such systems has increased while the price has been dropping dramatically. The wide-spread availability of Global Positioning Satellite (GPS) systems has enabled the development of advanced control systems, allowing novice pilots to simply provide high-level waypoint commands to the vehicles. Low materials and sensor costs in addition to the development of open source software has given amateur developers access to sophisticated tools which have dramatically increased the breadth and depth of research in the field as well as opened the platforms for a wide array of applications. While the most well known applications of UAVs are for military operations such as Intelligence, Surveillance and Reconnaissance (ISR), UAVs have been used to measure atmospheric



Figure 1-1: Quadrotor Tracking a Dynamic Surface Target

conditions and pollution, facilitate disaster relief and emergency management, serve as a communications relay, and assist in search and rescue operations.

Currently, fixed-wing UAVs are used for the majority of these missions since the UAVs typically fly in open airspace void of any real obstacles. These vehicles are usually large, can carry substantial payloads allowing them to perform all computation on-board, and rely on GPS for position estimation. The MQ-1 Predator UAV shown in Figure 1-2(a) is a well-known fixed-wing platform used in military operations. However, their size and reliance on GPS make fixed-wing UAVs unsuitable for applications in an urban environment. The dynamics of fixed-wing platforms require forward movement for continuous flight making navigation in an urban environment difficult. Additionally, their reliance on GPS for position information becomes a hazard in urban areas where large buildings may cause severe GPS dropouts. Autonomous indoor navigation would be extremely difficult due to the size and maneuverability of current fixed-wing UAVs as well as the lack of GPS reception.

Rotary-wing UAVs seem uniquely suited to operate in this cluttered urban environment. Rotary-wing UAVs have several advantages over their fixed-wing counterparts. Primarily, a rotary-wing vehicle is not required to maintain a forward velocity



(a) MQ-1 Predator Fixed-Wing UAV [9]



(b) Fire Scout UAV [23]

Figure 1-2: Various Forms of Currently Used Unmanned Aerial Vehicles

to sustain flight. The ability to hover permits the vehicle to make heading changes in cluttered environments which would not be possible with the turning radius of a fixed-wing vehicle. Additionally, the ability to hover allows the platform to maintain a sensor's field of view on a single target for extended periods of time. The ability to vertically takeoff and land (VTOL) reduces the launch and recover footprint of the vehicle, thus making it easier to deploy and retrieve. VTOL capability also eliminates the need for landing strips or other launch assisting devices such as catapults. The MQ-8A Fire Scout shown in Figure 1-2(b) is a typical UAV of the traditional helicopter design.

Traditional helicopters are mechanically complex systems. A large main rotor is used to provide lift as well as translational motion. A swash plate assembly on the main rotor hub allows the pilot to adjust the angle of the main rotor blades during flight. In order to prevent the body of the vehicle from spinning due to the counter torque of the main rotor, an additional tail rotor is needed. To drive the tail rotor, a long drive shaft runs from the main rotor's transmission to a smaller transmission in the tail rotor through the tail boom. Furthermore, the tail rotor extends beyond the footprint of the main rotor which increases the area required to maneuver the helicopter safely.

The fundamental quadrotor design has remained relatively simple and eliminates some of the mechanical complexities characteristic of the traditional helicopter design. Four rotors are mounted in a symmetrical pattern equidistant from the vehicle's

center. The motors directly drive the rotors which eliminates the need for a gear box. Since combustion engines typically have a slow response, electric motors are used to adjust the rotor speed at the high frequency needed for stable control. While multiple motors increase the payload capacity of the platform, they also increase the vehicle's weight and energy consumption. With four inputs but six degrees of freedom, the quadrotor is a classic under-actuated system. It is impossible to move in a translational direction without changing the attitude of the vehicle. These fast and under-actuated dynamics make it an ideal platform for controls research. The Ascending Technologies *Hummingbird* quadrotor is a research level platform which can be seen in Figure 1-3.



Figure 1-3: The Ascending Technologies *Hummingbird* Quadrotor in Flight

## **Motivation**

This thesis describes the development of a small, quiet, and computationally lean robotic flying camera system that can be used to automate data collection in support of environmental studies. Our key application domain is surveillance at sea—observing objects such as whales and boats where the target stands out against a relatively uniform background. This problem has two aspects we address using computer vision:



1. Using the vehicle's vision system to detect and track objects at sea.
2. Autonomously controlling the robot using visual position estimates.

Observing whales is important for many marine biology tasks including taking census, determining family lineage, and general behavioral observations. Currently, whales are observed manually using binoculars and cameras from the shore or boats. Notes of their behavior are typically made using pencil and paper. The process is error prone, non-quantitative and very labor intensive. Human-operated planes and helicopters are also used, but the data gathered this way is limited. Planes fly at high altitude, can not hover, and the data collected is limited in duration and precision. Traditional helicopters can hover and fly closer to the sea surface, but they are noisy and drive the whales away.

In our first field experiments we used a small hovering UAV to automate the data collection of whales. Figure 1-4 shows a typical frame from this video data set. We found that the robot is quiet enough to fly close above the water surface and not disturb the whales. The robot can hover and adjust its velocity to track the whales and capture their natural behavior with images of unprecedented detail. These field experiments were done between August 20-25 2009 when a joint MIT and Argentinean Whale Conservation team deployed a remote control Ascending Technologies Falcon 8 robot over the sea at Peninsula Valdez, Argentina to collect data on Southern Right whales. The team completed several successful missions of approximately fifteen minutes each, during which the robot was manually piloted over groups of whales and video was recorded. Using this data, the goal in this thesis is to create a computationally impoverished flying robot system that relies on vision-based position estimates and only on-board processing to navigate, detect whales, and track whales in order to automate these types of field experiments. This is different than existing systems that use additional sensors such as GPS or laser scanners for position estimates or perform off-board processing of the video stream.



Figure 1-4: Sample Frame From Video Footage Recorded in Argentina

### **Method of Approach**

The whale tracking algorithm performs object recognition using a pixel-level classifier and domain knowledge. Image hue and saturation values are suitable invariants to segment whales from other elements in the scene. A two-dimensional mathematical model is created to describe the target. In subsequent frames, individual pixels are evaluated and assigned a probability of being a target pixel based on their hue and saturation values as compared to the target model. Groups of high probability pixels are identified as objects and presented to the user.

The algorithm was evaluated and extensive results obtained over 13,000 frames representing Southern Right whales, Blue whales, Grey whales, and Humpback whales. The data was collected from a variety of angles under varying lighting conditions. The results of these tests showed 98.99% recall for 3,537 frames of Southern Right whale footage collected in Argentina. To show adaptability to varying target types, over 5,823 frames of video footage containing boats was evaluated as well. This footage also displayed the ability of the vision system to track multiple targets with varying characteristics simultaneously.



The visual servoing control system is built upon the object identification and tracking algorithms. This ability to position the quadrotor above a moving target is demonstrated through several indoor experiments. The vision system serves as a high level path planner, creating a desired trajectory to position the quadrotor over the target. A traditional quadrotor model is adapted to include the dynamics of an on-board attitude controller and the model parameters are learned through a system identification process. This model is used to formulate Linear Quadratic Regulator (LQR) position controllers in order to maneuver the quadrotor to a desired state. The additional payload of the on-board computer and camera makes control difficult. The weight is essentially a non-aligned constant disturbance which has to be accounted for with constant offsets and integrator control. Velocity state control is also necessary for precise position control. This requires accurate velocity signals from either motion capture data or an Extended Kalman Filter (EKF). The camera is operated at sixty frames per second, but due to the on-board computational limitations, the vision system is run only at ten to twenty frames per second. The control commands are computed at forty hertz.

Our methodology for designing these algorithms begins with developing a base-level controller using a motion capture system that provides highly accurate vehicle state feedback at a high frequency. The vision system is used to track a designated target. The target for the hardware experiments is the iCreate robot as shown in Figure 1-1. For proof of concept of outdoor tracking, the whale footage was projected onto the floor and the quadrotor identified the simulated whales in the image. A sample image from this experiment is shown in Figure 1-5. Additionally, we remove the external localization system and replace it with visual position estimates for controlling the robot. The vision system position estimates are fused with on-board Inertial Measurement Unit (IMU) data in an EKF to estimate the state of the vehicle. The robot was able to reliably track the moving surface target independent of any external localization source. This ability would be useful for urban environments or indoor settings when external localization such as GPS temporarily drops out or is completely unavailable.

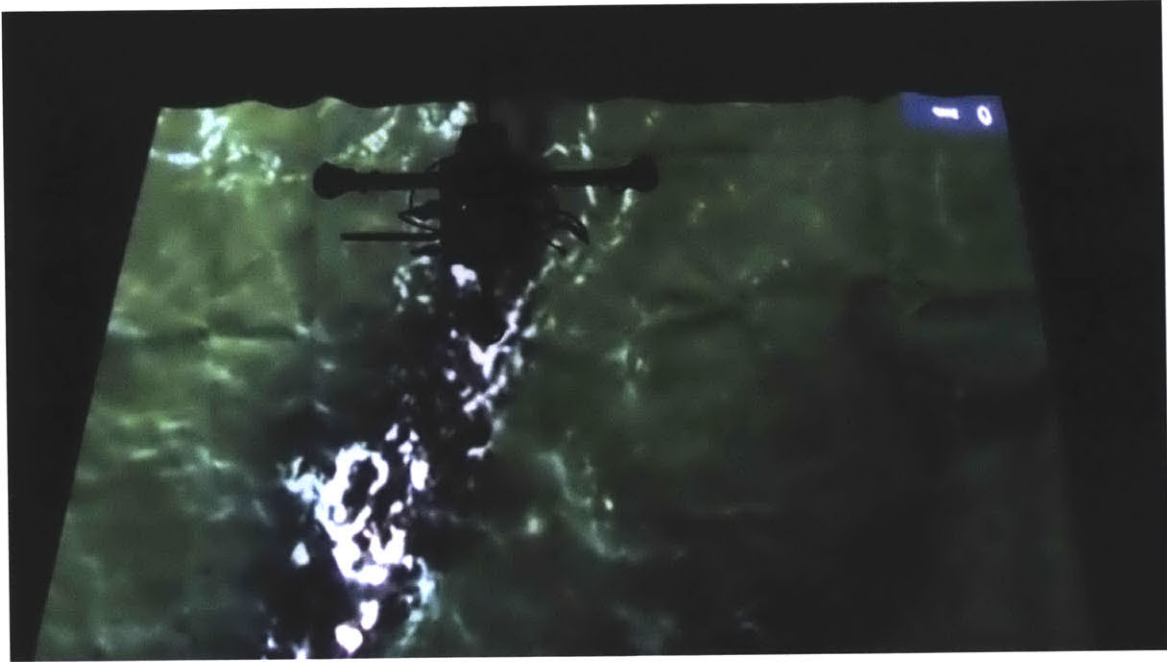


Figure 1-5: Tracking of Projected Whale Video Footage

### Technical Challenges

A key challenge in this thesis is to avoid the reliance on an external processing source in order to develop a system robust to communication dropouts. A communication link is often used to transfer images and control commands between the robot and a ground station. Wireless bandwidth may be limited in areas and not provide the means to transmit images at a servable frequency when processed off-board. While image compression techniques could increase transmission rate, it could also create artifacts which may negatively impact the performance of computer vision algorithms. Any delays in the system could result in unstable flight trajectories due to the fast and under-actuated dynamics of the quadrotor. The system resulting from this research is therefore capable of operating at longer ranges and is generally safer than one relying on external processing.

Computational efficiency is a key issue in this work and considerable time was taken to develop a vision algorithm that was not only accurate but also computationally fast. Initially, image gradient-based feature tracking was used to track the intended target, however this was abandoned for the proposed method using hue

and saturation values. The proposed method is highly user intuitive and produces accurate results for varying video footage quality.

The fast dynamics of the quadrotor require a fast controller for stabilization. A classical PID controller was implemented initially but was replaced by a set of LQR controllers derived from a validated system model. The quadrotor's fast dynamics also amplify any delays in the system and creates instability. The system architecture was modified to utilize a global system timestamp to more accurately process sensor measurements which improved the state output of the filter. Hardware pieces were designed and redesigned to reduce weight as much as possible. In the end, these challenges were overcome to produce an autonomous visual servoing system utilizing only on-board computation.

## 1.1 Contributions

The ability for MAVs to autonomously navigate in an outdoor environment independent of external processing and localization is currently a highly desirable yet elusive capability. Current independent outdoor navigation systems rely on a combination of GPS and Inertial Navigation System (INS) measurements processed on-board a microcontroller. Alternatively, systems using cameras or laser scanners for localization and control rely on off-board processing. There is a need for systems that are able to navigate in unknown and/or GPS denied environments. In turn, autonomous reliable navigation will enable solutions to higher-level tasks such as exploration and multi-vehicle trajectory planning. While large UAV systems have utilized on-board computation for image processing and localization, this capability is currently unavailable for smaller MAV systems. An autonomous MAV is essential for indoor operation or environments where agile movement is critical.

Towards these goals, our contributions include:

- *Fast Object Representation and Identification Algorithms for Computationally Impoverished Platforms* - An image segmentation algorithm is presented which proved to be of high accuracy while being computationally efficient enough to run in real time. This algorithm was tested on varying object types, can be used

to track multiple objects of differing characteristics, and is capable of creating a trajectory to servo a vehicle.

- *Model Adaptation, Simulation, and Controller Synthesis* - The classical quadrotor model is adapted to take advantage of the on-board attitude stabilization system. The model is developed in MATLAB's Simulink toolbox with parameters learned from real test flight data. This infrastructure can be easily adapted for other quadrotor types and be used to simulate and validate various types of position controllers. Based on the validated model, LQR position controllers are designed and implemented.
- *On-board IMU and Monocular Vision Based Control for a Computationally Impoverished Platform* - A control scheme is developed for completely on-board visual servoing. Both the control and image processing algorithms are executed on-board the quadrotor platform. The visual position estimates are used to create a desired trajectory along with motion capture state feedback. Further, the visual position estimates and on-board IMU data are used as inputs to an EKF to provide accurate state estimation with minimal delay.
- *Experimental Verification of the Visual Servo Control Schemes* - In physical hardware experiments, the performance of the vision and control systems were evaluated. The iCreate target was programmed to move in a varied trajectory at a constant speed. Utilizing the motion capture system for state feedback, the quadrotor tracked a surface target using position estimates from the vision system. Estimated state feedback computed on-board was then used to maneuver the quadrotor as it reliably tracked the iCreate.

## 1.2 Thesis Overview

This thesis is divided into six chapters. Chapter 2 presents related work in the areas of quadrotor vehicles, image segmentation and identification, and visual servoing. Chapter 3 describes the experimental set-up to motivate the development of the object identification algorithms as well as the visual servoing control systems. The

camera modeling and calibration process is discussed as well as the quadrotor modeling and system identification process. This chapter also explains the development of the EKF used for state estimation. Chapter 4 discusses the development of the image segmentation algorithm as well as the target tracking algorithm. Chapter 5 describes the derivation of the low level controller for stable quadrotor flight. Chapter 6 presents an evaluation of the proposed vision algorithms utilizing whale video footage. The results of several experiments utilizing the vision algorithms to track a moving surface target are also presented. Chapter 7 concludes the thesis and presents lessons learned as well as several areas of future research.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 2

## Related Work

This thesis builds upon a large body of work in:

- Developing MAV systems.
- Developing algorithms for object recognition and tracking.
- Developing visual servoing control systems.

### 2.1 General Background

The quadrotor is not the most well known form of a rotary-wing vehicle, yet its unique design dates back to August 24, 1907 when the Breguet-Richet *Gyroplane No. 1* lifted a few feet off the ground in France [58]. This model used a single engine to drive all the rotors and was therefore underpowered and lacked a proper means of control. It was not until 1956 that D. H. Kaplan demonstrated the ability to control the aircraft attitude using differential thrust [33], but this design modification has stayed consistent through current production and research models.

Popular hobby models today include the MiKroKopter [68] and the Draganflyer [46]. Recent advances in on-board microprocessors, micro-electro-mechanical-systems (MEMS) inertial sensors, high density power storage, and integrated miniature actuators have enabled these quadrotor vehicles to be outfitted with on-board attitude stability. This drastically increases the ease of use of the platform and has made the quadrotor more accessible to amateur pilots and researchers. The Parrot

AR Drone has two on-board cameras and is able to be flown via an iPhone application [26]. The microdrone md4-200 is also a commercially available quadrotor platform equipped with an on-board camera [63].

On-board cameras are becoming one of the most common sensors on-board both commercial and research quadrotor platforms. Typically the cameras are used for surveillance and inspection operations. Computer vision techniques are currently being applied to identify and track marine animals to aid marine biologists [53,54,79] and even to help mitigate damage to whales during US Navy sonar operations [83]. Currently, research is being done in the area of visual odometry which allows the quadrotor to stabilize based on feedback from the vision system [2]. When Global Positioning Satellite (GPS) reception is available, outdoor systems can utilize target tracking algorithms to follow a target by creating a series of desired GPS waypoints [52].

## 2.2 Quadrotor Development and Applications

Current research in the control of quadrotors has its origins in research by Hauser et al. who used a minimum phase approximation with input-output linearization on a Harrier jump jet aircraft model [36]. This method was later improved beyond slightly nonminimum phase systems to strongly nonminimum phase systems by Martin et al. [61]. Shim et al. evaluated the effectiveness of linear multi-variable control, fuzzy logic control, and nonlinear tracking control on a traditional helicopter model [84]. Using a similar traditional helicopter model, Frazolli et al. presented a tracking controller which avoided the artificial singularities introduced from attitude parameterization [31]. Young et al. developed a quadrotor tail-sitter vehicle which combined a flying-wing design with quadrotor propulsion. This unique design resulted in a vehicle with both hover capability as well as high-speed aerodynamic efficiency [92].

Extensive modeling of the quadrotor platform in the early 2000's as well as the development of advanced control techniques for attitude stabilization marked the beginning of considerable research interest in this platform. Hamel et al. developed a generic quadrotor model which included aerodynamic and gyroscopic effects in



addition to the airframe and motor dynamics [35]. A joint collaboration between Commonwealth Scientific and Industrial Research Organization (CSIRO) and the Australian National University lead to the development of the “X-4 Flyer” quadrotor platform [75]. The second generation was presented in 2004 [78] which improved the mechanical design based on simulation results and extensive modeling of roll and pitch rotor damping as well as blade flapping [22]. The authors were able to achieve stable attitude control in the roll and pitch axis in tethered flight [22] and even outdoor autonomous flight [77]. Early work using vision feedback as a primary sensor was done by Altug et al. [5]. A ground camera was used to estimate the pose of the helicopter and a feedback linearizing controller as well as a backstepping-like controller were presented.

With over a decade of research into the modeling and control of the quadrotor platform, the vehicle has become popular in aerial vehicle research around the world. cole Polytechnique Fdrale de Lausanne (EPFL) has developed the “OS4” quadrotor [11,12], the French Alternative Energies and Atomic Energy Commission’s (CEA) unique quadrotor used four blades per motor [70], and Cornell’s Autonomous Flying Vehicle was built using commercially available parts [72]. Stanford developed a quadrotor known as STARMAC which is used to implement decentralized control algorithms [40]. Utilizing the STARMAC platform, controllers have been designed which compensate for blade flapping and thrust variation while performing a complicated stall turn maneuver [41, 45]. MIT’s Aerospace Controls Laboratory (ACL) developed the Real-time indoor Autonomous Vehicle test Environment (RAVEN) system to evaluate algorithms for multi-agent missions as well as low-level controller performance [42].

Research has now progressed beyond design, construction, and basic attitude stabilization of the quadrotor platform and into investigations of control strategies for advanced trajectory following. The Ascending Technologies quadrotors have enabled researchers to focus on position control by providing platforms with robust on-board attitude controllers [34]. MIT’s ACL developed simple LQR controllers based on linearized quadrotor dynamics which allow multiple aerial vehicles to follow unique tra-

jectories simultaneously [89]. Salazar-Cruz et al. utilized nonlinear control laws with nested saturations to stabilize the quadrotor around simple planar trajectories [81]. Lupashin et al. used a learning strategy to perform high-speed simultaneous flips with the quadrotor [60]. At the University of Pennsylvania, Mellinger et al. developed aggressive trajectories which allowed quadrotors to maneuver through small windows and perch on nearly vertical surfaces [66].

## 2.3 Object Recognition

The field of computer vision has an extremely large literature in image segmentation, object representation, and object tracking. Several novel image processing methods have been introduced and improved. This thesis uses image thresholding techniques based on the Hue-Saturation-Value (HSV) color space. It was shown that object identification has distinct advantages when using the HSV color space compared to the classical RGB color space in [87].

Traditional segmenting methods rely on image thresholding by assigning pixels to a user defined number of classes. Otsu’s method is an “optimal” thresholding algorithm which minimizes the variances between pixels in a class while maximizing the variance between classes [74]. Similarly, the Niblack algorithm [71] uses information about neighborhoods of pixels but requires extra tuning parameters which would make the resulting system vulnerable to lighting or environmental changes in the image. Techniques such as graph cuts and Maximally Stable External Region (MSER) work well on real world scenes but are computationally expensive. In graph cuts, each pixel is a vertex and edges are computed as the absolute value of the difference in color between pixels on the edge. After each iteration, edge weights below a certain threshold are removed and the adjoining pixels are classified as a single vertex [15,29,80]. MSER considers the set of all possible thresholding values of an intensity image and groups pixels into regions whose size remains relatively unchanged across varying threshold levels [62].

The methods described are unique in that they attempt to automatically segment the image into different regions. An alternative approach is to describe the intended

target by isolating certain characteristics of the target such as color or texture. This results in creating histograms which describe the entire image in terms of a specific characteristic. Hopefully, the intended target can be identified by a distinct peak in the histogram. Algorithms such as k-means clustering attempt to optimally isolate the strongest peaks in the data [18,49,51]. The Meanshift algorithm [32] differs from k-means by removing assumptions about the number and shape of the clusters. This algorithm performs gradient ascent on the histogram to find relative maxima in the data. All bins associated with the same peak are clustered together and can be used to identify and track a target [21]. The Camshift algorithm is an extension of the Meanshift algorithm which adapts the peak searching window size and has been used to segment moving targets [4]. Integral histograms developed by Porikli claim to be more accurate and computationally efficient than the Meanshift algorithm [76].

Additional methods are based on a known model of the intended target. If the cameras are fixed and the background is constant, foreground extraction techniques can be used as well as motion detection to identify targets moving in the image [16, 18, 91]. If labeled information is available, classifiers can be trained on sample data to learn a model of the object in a technique known as template matching [47, 57]. Also, particle filter based tracking algorithms have been applied to the tracking problem resulting in the Condensation algorithm [56, 90].

Instead of analyzing the relationship between the camera's position and a target in the image frame, visual odometry attempts to extrapolate the camera's position and orientation using features in the image. Popular feature detectors today include SURF [8] and SIFT [59] which use the RANSAC algorithm [30] for outlier detection. By detecting the movement of these features through several frames, the motion of the camera can be estimated [73]. Monocular SLAM techniques have also been developed to localize the camera system based purely on visual information [24].

## 2.4 Visual Control of Aerial Vehicles

A good review of vision based control of aerial vehicles is found in [55]. The goal of this thesis was to create a completely on-board, independent navigation and tracking

system. Typically aerial navigation systems involve the use of a GPS sensor [6,50,52] to provide accurate target tracking capabilities.

Cameras are quickly becoming an important sensor for autonomous navigation because they are small, passive, and consume low amounts of power. Other quadrotor systems use additional sensors such as laser scanners to provide accurate pose estimation but these can be heavy and a power burden [2,38]. Laser scanners often utilize SLAM techniques to estimate position which is not only highly computationally expensive, but also may require prior knowledge of the environment. Soundararaj et al. and Ahrens et al. performed visual SLAM to navigate a quadrotor in an indoor environment relying on off-board computation [3,86]. Stereo cameras are also popular but this technique loses accuracy as the distance from the target increases [43].

There has been a specific interest in using visual position estimates for autonomous landings, but these systems often use special landing pads that allow accurate position and attitude estimation [48,67,82]. The reliance on a predefined target is not practical when the system needs to track targets of varying and unknown characteristics.

Many vision-only systems require off-board computation since the platform is too small to carry a computer capable of executing the necessary algorithms at a fast enough frequency [19,20,27,88]. Additionally, many of these vision-only systems have no means of estimating altitude [19,69]. Instead, an additional sensor is used or the altitude is controlled manually.

Finally, the vision system developed in this thesis is based on the estimated centroid of the intended target. Many vision systems use feature finding techniques such as optical flow [19,39,65,86]. Soatto et al. [85] developed forms of the implicit Extended Kalman Filter (EKF) which incorporated vision based measurements into the EKF by making use of the epipolar constraint. Roy et al. utilized a machine learning classifier to distinguish targets from the background and perform geo-location of the identified target [37]. The estimated target position was then combined with a GPS position to autonomously track a moving vehicle.

At the time of this thesis, current work at Eidgenössische Technische Hochschule Zurich (ETH) is progressing towards the development of systems using visual feedback

fused with IMU data processed on-board a quadrotor platform. Pollefeys et al. [64] developed the PIXHAWK platform which utilizes four cameras for localization, pattern recognition, and obstacle avoidance. Computation occurs on-board two separate dual core processors and an artificial marker localization scheme is used. Siegwart et al. [10] use visual SLAM techniques to control the position of a tethered quadrotor vehicle. In untethered experiments, a specific circular landmark was used to extract the vehicle state from image information [28].

## 2.5 Context for This Research

This research builds upon the previous literature in several distinct areas. The basic nonlinear model of the quadrotor was taken from the literature and modified by altering the control inputs as well as estimating the dynamics of the on-board attitude controller. Many models use desired motor speeds as control inputs. However, the quadrotor used in this thesis had an on-board attitude controller to compute the motor speeds. Since we did not need to perform any aggressive maneuvers with the vehicle, aerodynamic effects such as blade flapping were ignored.

In the discipline of computer vision, this research diverges from the computationally expensive image segmentation methods such as graph cuts or k-means clustering. Otsu’s method, for example, only works for two distinct pixel classes and would be unsuitable for this application where the target and background could have similar characteristics. Instead, we use the HSV color space as suggested by previous work to create a simple yet accurate two-dimensional model of the target. This method is computationally fast, easily portable, and open to user interaction.

The control schemes we present in this research build upon previous work combining computer vision-based feedback and control. Feature based localization is useful for navigating static indoor environments but can provide noisy and inaccurate velocity estimates when applied to a static and slightly uniform background such as the ocean. Some vision-based localization methods rely on off-board camera systems which are not practical for our proposed scenario. Reliance on off-board computation requires images and control commands to be transmitted wirelessly, leaving room for

interference with the signals. By performing the computation on-board, this system avoids these complications as well as potential time delays. The method of altitude estimation in this research is accurate at low altitudes which is advantageous when compared to pressure sensors which have significant drift errors. The combination of feedback from a vision system and IMU measurements has been successfully implemented in the previous work of others. We take this feedback strategy and implement the state estimation and image segmentation completely on-board the vehicle. The result is a visual servoing system which utilizes vision-based position estimates and IMU measurements for control without relying on external localization or processing. This makes our system capable of operating without GPS updates for extended periods of time.

# Chapter 3

## Experimental Set-up

We address the problem of tracking objects at sea by using an autonomous quadrotor robot equipped with a downward-pointing camera and an on-board computer. In this section we detail the hardware and software architecture of this system and highlight the intrinsic constraints that have motivated our solution to visual object recognition and visual servoing. We also describe the specifics of the experiments used to validate the proposed control strategy in hardware. The quadrotor was tasked with identifying and tracking a programmable surface target.



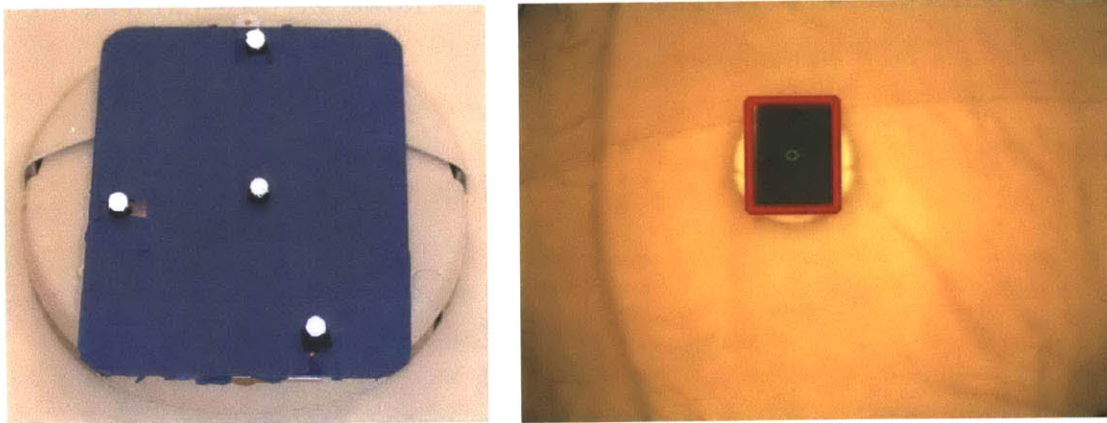
Figure 3-1: Quadrotor Outfitted With fit-PC2 and Firefly MV2



### 3.1 Laboratory System Architecture

We utilize an Ascending Technologies *Hummingbird* [34] quadrotor extended with a CompuLab fit-PC2 computer and a Point Grey Firefly MV2 camera. The quadrotor is controlled using the vision-based control strategy we describe in Chapter 5 with position estimates from the object identification and tracking algorithms we present in Chapter 4. A Vicon motion capture system is used to measure the ground truth pose of the quadrotor and the target.

The network topology for the hardware experiments is summarized graphically in Figure 3-3. The solid lines represent wired Ethernet connections and the dashed lines represent wireless connections. This figure includes the computers necessary to use the Vicon motion capture system. This system outputs the position and attitude of the quadrotor and target at 120 Hz with sub-millimeter accuracy and minimal delay. The motion capture data is forwarded to the fit-PC2 via an 802.11 UDP connection. A laptop is used to remotely run the commands on the fit-PC2 via a wireless SSH connection but no computation is performed on the laptop during the experiments.



(a) Target Used for Experiments

(b) Vision System Output of Target

Figure 3-2: Views of the iCreate Target Used for Hardware Experiments

The target for the robot tracking experiments was a  $21.5 \times 28.0$  cm blue clipboard mounted onto an iRobot iCreate as shown in Figure 3-2(a). The iCreate was programmed to follow a specific trajectory at a constant speed and direction. For these experiments, the iCreate moved at a desired speed of 2.5 cm/s. Figure 3-2(b) shows



a sample output frame of the vision system tracking the iCreate.

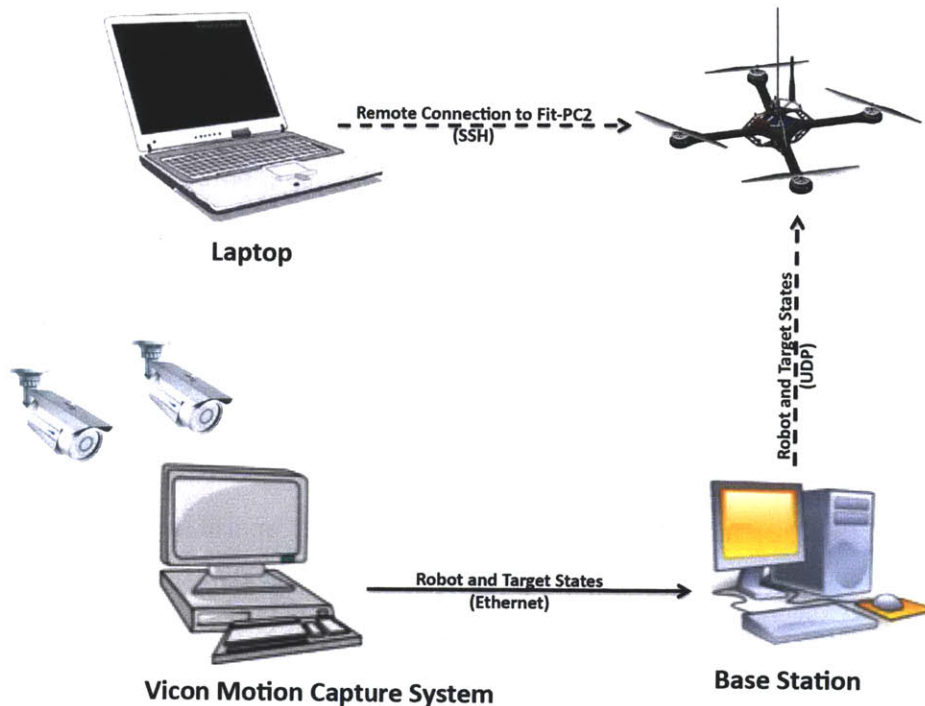


Figure 3-3: Experimental Network Topology

### 3.1.1 Experimental Hardware

The vision system is used to provide high-level position control of an Ascending Technologies *Hummingbird* quadrotor vehicle. The *Hummingbird* is  $40 \times 40 \times 10$  mm in size weighing 550 g out of the box. The maximum recommended payload is 200 g for a flight time of twelve minutes running off a single 11.1 V Lithium Polymer (LiPo) battery. On-board, the *Hummingbird* is equipped with a Global Positioning Satellite (GPS) receiver, compass, and a Micro Electro-Mechanical Systems (MEMS) gyro. Autonomous position control is executed by sending low-level attitude commands to the on-board attitude stabilization control via a serial interface. It is possible to modify the on-board pre-programmed attitude control system, but this is typically only done when the quadrotor needs to execute harsh maneuvers. For this research, the quadrotor was intended to hover stably and the on-board attitude controller was sufficient.

The standard *Hummingbird* quadrotor is outfitted with a fit-PC2 computer and Firefly MV2 camera system. A bracket was 3-D printed to mount the camera and fit-PC2 to the bottom of the quadrotor. A USB to serial cable was used to interface with the quadrotor AutoPilot board and a power cable split the on-board battery power between the quadrotor and fit-PC2. A significant experimental issue was powering the equipment. The full system combines for a total payload of 535 g, well above the recommended maximum payload of 200 g for this platform. The additional weight combined with the added power requirements of the computer and camera results in a flight time of about six minutes, roughly half the normal flight time. The final experimental configuration of the quadrotor is shown in Figure 3-1.

All computation is executed on-board a CompuLab fit-PC2. This computer is only  $101 \times 115 \times 27$  mm in size and weighs 300 g. It operates off the same 11.1 V battery as the quadrotor and consumed 8 W at full CPU load. The fit-PC2 contains an Intel Atom Z550 2 GHz processor with 2 Gb of RAM. It is equipped with 6 USB 2.0 ports, an 802.11n WLAN card, and is booted from a 16 Gb SD card. The solid state drive is more robust to crashes than the common disk drive and is also very lightweight.

Currently, popular on-board computation systems such as the Gumstix rely on the ARM processor. However, the Atom processor has several unique advantages when compared to other processors. Primarily, since it uses the x86 architecture, it enables the user to install standard operating systems. This processor also allows for the use of standard drivers for devices such as the wireless network card and Firefly camera. This results in a system that can be easily reproduced and upgraded. Additionally, the Intel Performance Primitives (IPP) are available for the Atom processor. This performance library can be integrated with the open source computer vision libraries (OpenCV) [17] to boost performance of common image processing and computer vision functions.

Images are transferred over a USB 2.0 connection from the Firefly MV2 to the fit-PC2. The images are downsampled to  $320 \times 240$  pixels. For the visual servoing experiments, the intended target is relatively large in the image. Therefore, down-

sampling does not result in a loss of a large amount of information about the target. This improves computational efficiency without a large sacrifice in accuracy. The camera is capable of  $640 \times 480$  pixels at sixty frames per second but in practice the on-board computation limitations results in a frame rate around twenty frames per second. Off-board processing of the image would require wireless transmission of image data. This additional processing and potential for interference would further reduce the frame rate.

### 3.1.2 Software Architecture

The software architecture is described in Figure 3-4. Several modules are run simultaneously on the fit-PC2. Message routing between modules as well as data logging is handled by the Lightweight Communications and Marshaling (LCM) library [44]. Developed by the MIT DARPA Urban Challenge Team, LCM was designed for tightly-coupled systems connected via a local-area network. These libraries are targeted at real-time systems where high-bandwidth and low latency are desired.

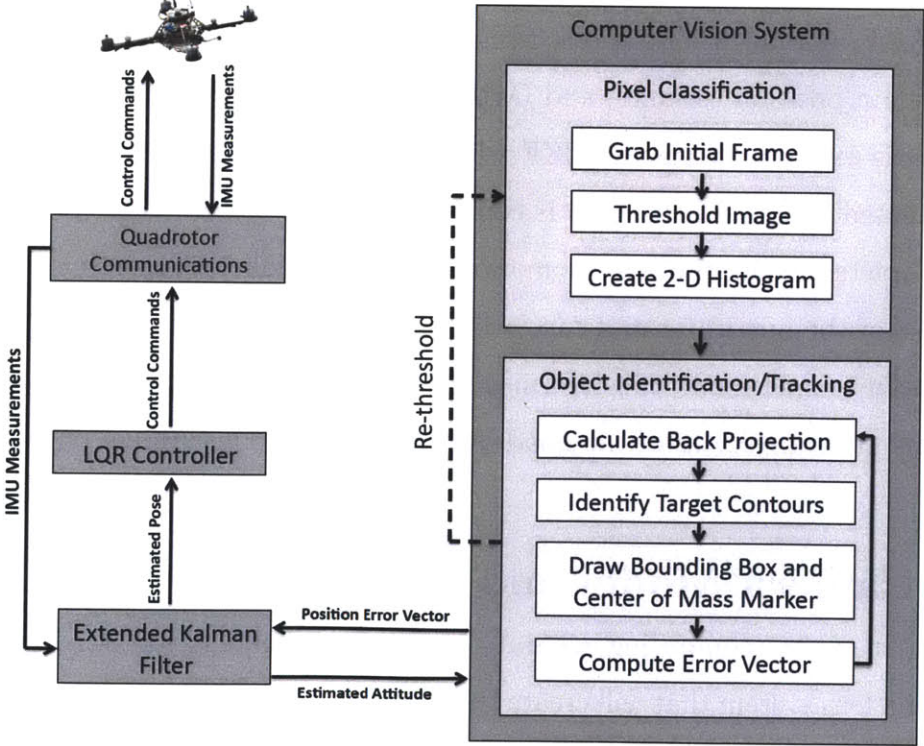


Figure 3-4: Experimental Software Architecture

The vision system module uses the algorithms described in Chapter 4 to identify the target in the image frame. Additionally, Equations 3.19, 3.20, and 3.21 are used to compensate for the vehicle's attitude, convert the vision system position estimates from image plane coordinates in pixels to a body coordinate frame in millimeters, and estimate the vehicles altitude. This module runs at ten Hz due to the additional processing of the vision algorithms.

The control module utilizes four independent LQR controllers described in Chapter 5 to compute the roll, pitch, yaw and thrust commands. It receives the full state pose of the robot either from the motion capture module or from the EKF module depending on the experiment being run. The on-board AutoPilot software computes the individual motor commands. The position control commands were calculated at forty Hz.

The quadrotor communication module is used to interface with the quadrotor AutoPilot board. This module receives Inertial Measurement Unit (IMU) measurements and transmits control commands via a USB to serial connection. This module operates at thirty Hz.

The EKF module, which is discussed in more detail in Section 3.4, receives the vision system position estimate and IMU measurements. The estimated pose of the quadrotor is extracted from the EKF which is used to calculate control commands. The estimated pose from the filter is computed at 110 Hz.

Not depicted in the figure is the motion capture module which receives the ground truth pose of the quadrotor and target from the motion capture system. Depending on the specific experiment, this information is used to control the robot as well as log the ground truth positions of the quadrotor and target. This system operates at 120 Hz.

## 3.2 Camera Model and Calibration

The first step to computing accurate computer vision algorithms is to model the camera and lens system that will create the images necessary for the vision algorithms. Cameras of different qualities produce different images of the same scene. While

more expensive cameras are designed to minimize these errors, the small, portable, and inexpensive cameras desired for this research are not constructed to such high specifications. The primary errors in camera images are caused by radial and tangential distortion. In order to calibrate the camera, it is necessary to compute both the intrinsic and extrinsic properties of the camera. Fortunately, there are methods to classify the camera and lens parameters to rectify the images of these errors which we discuss in Section 3.2.4.

The vision algorithms we developed rely on accurate color space image segmentation. The more varied the color characteristics of the target when compared to the background, the better the algorithms perform. To improve the color capture and representation it is necessary to properly calibrate the camera. Once the target has been successfully identified we need to accurately know the target's location. The output of the vision system is the estimated target centroid position in meters. For this estimate to be accurate, all camera distortions need to be taken into account. The control strategy for positioning the quadrotor relies heavily on an accurate transformation of the target centroid from pixel coordinates to robot coordinates. Removing the effects of these distortions has a direct impact on the accuracy of the visual servoing control system.

### 3.2.1 Camera and Lens Selection

We chose the Point Grey Firefly MV2 which has a 1/3" progressive scan Complementary MetalOxideSemiconductor (CMOS )(Aptina MT9V022) image sensor. The camera weighs only 30 g and has a maximum resolution of  $752 \times 480$  pixels at sixty frames per second. This camera is commonly used in robotics due to its lightweight and compact form, strong API, and high resolution at fast frame rates. Images are pulled from the camera via a USB 2.0 High-Speed interface. While the camera is similar in size to a webcam, it produces images of machine vision grade quality. Since the camera uses a global shutter, it avoids the edge artifacts caused by the rolling shutter in typical CMOS cameras.

A Tamron 1/3" lens is chosen as well. With a 2.2 mm focal length, the field of view is approximately  $119^\circ \times 90^\circ$ . This wide angle provides a larger view of the world

and therefore the target. The lens weighs 41.3 g and a ring lock prevents accidental adjustment after the lens has been focused. The lens also contains IR elements which improve the brightness of the images regardless of lighting conditions. The wide angle lens causes significant image distortion which is accounted for in the camera calibration process.

### 3.2.2 Pinhole Camera Model

The simplest model of the camera is known as the pinhole camera model which is depicted in Figure 3-5. The focal length relates the size of the image on the image

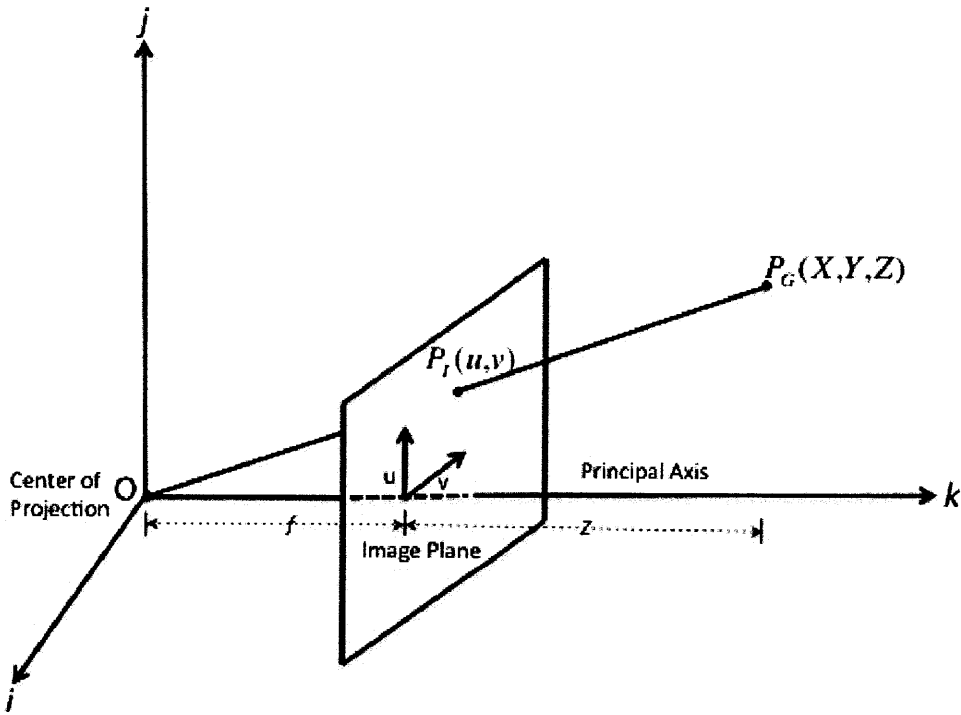


Figure 3-5: Pinhole Camera Model

plane to the size of the image in real life. In the idealized pinhole model, the focal length,  $f$ , is the fixed distance from the center of projection to the image plane. The pinhole model is characterized by Equation 3.1.

$$\begin{aligned} u &= F_x \left( \frac{X}{Z} \right) + c_x \\ v &= F_y \left( \frac{Y}{Z} \right) + c_y \end{aligned} \tag{3.1}$$

Equation 3.1 maps a point  $P_G$  in the physical world with coordinates  $(X, Y, Z)$  to the point  $P_I$  on the image plane with coordinates  $(u, v)$  and is known as a projective transform. The values  $F_x$  and  $F_y$  represent the physical focal length in distance units such as millimeters. The principal point  $(c_x, c_y)$  is the estimated center of the image plane. Since the image sensor cannot be perfectly placed, the principal point represents a translation from the optic axis to the center of the image coordinate frame.

In order to compute the values of the point  $(u, v)$  in pixels, it is necessary to calculate the focal lengths in pixel units. This is done with Equation 3.2.

$$\begin{aligned} f_x &= F_x * s_x \\ f_y &= F_y * s_y \end{aligned} \tag{3.2}$$

Here  $f_x$  and  $f_y$  are the focal lengths of the lens in pixel units. This value is computed by taking the product of the physical focal length in millimeters  $(F_x, F_y)$  and the size of the imager elements  $(s_x, s_y)$ . This model assumes rectangular pixels which causes  $s_x$  and  $s_y$  to be distinct values with units of pixels per millimeter. While neither the focal lengths  $(F_x$  and  $F_y)$  or the scaling factors  $(s_x$  and  $s_y)$  can be measured directly without taking apart the camera, their products  $(f_x$  and  $f_y)$  can be estimated through a camera calibration process.

### 3.2.3 Lens Distortions

Due to difficulties in manufacturing a perfect lens as well as exactly aligning the optical axis of the lens with the imager center, image distortions occur. The two main types of distortion are known as radial distortion and tangential distortion.

Radial distortion is evident when the lens distorts the pixels near the edge of the image. This type of distortion is commonly known as the “fish-eye” effect. Radial distortion occurs when the light rays enter the lens on the outer edges and are refracted more strongly than if they had entered at the center of the lens. While expensive lens systems and high-end cameras can take measures to reduce this distortion, radial distortion is pronounced on lower end camera systems such as the one used for this

research.

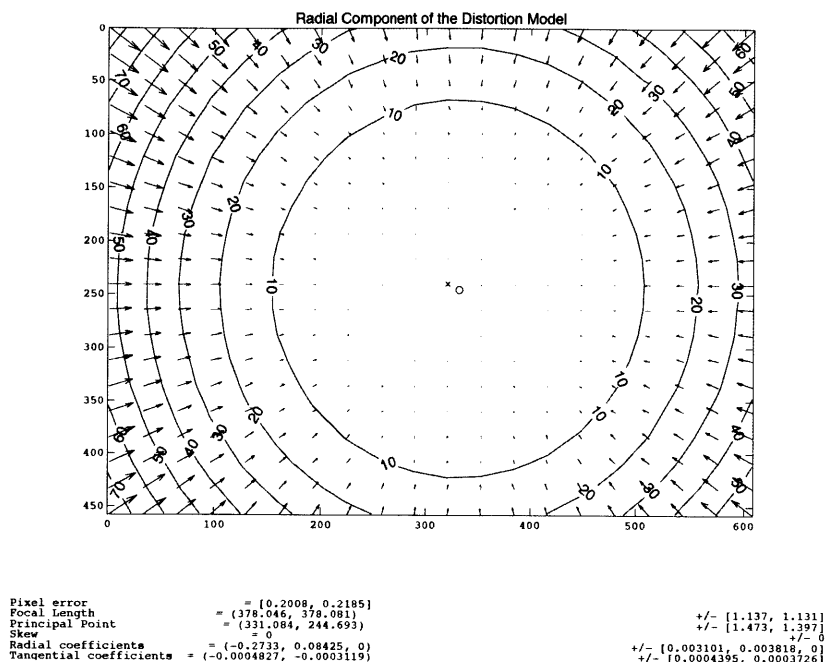


Figure 3-6: Effects of Radial Distortion

Radial distortion is estimated with a polynomial function which is a Taylor series expansion around  $r = 0$ . Here  $r = \sqrt{u^2 + v^2}$  which implies that  $r = 0$  is the center of the image plane. The value of  $r$  is symmetrical which removes the even coefficients in the expansion. Additionally, since  $f(r) = 0$  at  $r = 0$ , the first term in the expansion is also removed. The remaining structure is  $f(r) = k_1r^2 + k_2r^4 + k_3r^6 + \dots$ . However only the first two terms are generally needed in the calibration process. Equation 3.3 summarizes the equations used to estimate the radial distortion in an image.

$$\begin{aligned} u_c &= u(1 + k_1r^2 + k_2r^4) \\ v_c &= v(1 + k_1r^2 + k_2r^4) \end{aligned} \quad (3.3)$$

In this equation,  $(u, v)$  is the original location of the distorted point,  $(u_c, v_c)$  is the corrected location, and  $k_i$  are the estimated radial distortion coefficients. Figure 3-6 shows the effects of radial distortion on the Point Grey Firefly MV2.

Tangential distortion is caused by the misalignment of the imager surface and the lens. In cheaper cameras, these are not always exactly positioned. Tangential



distortion is estimated with two additional parameters  $p_1$  and  $p_2$  as shown in Equation 3.4.

$$\begin{aligned} u_c &= u + [2p_1xy + p_2(r^2 + 2x^2)] \\ v_c &= v + ([2p_2xy + p_1(r^2 + 2y^2)] \end{aligned} \tag{3.4}$$

Figure 3-7 depicts the effects of tangential distortion on the Point Grey Firefly MV2.

Depending on the magnitude of the radial distortion, an additional  $k_3$  parameter can be used. In practice, the four distortion coefficients  $k_1, k_2, p_1, p_2$  are sufficient. The camera model used in this research is a combination of the pinhole camera model with the additional equations to account for radial and tangential distortion. The corrected pixel location is computed as a linear combination of the effects of radial and tangential distortion.

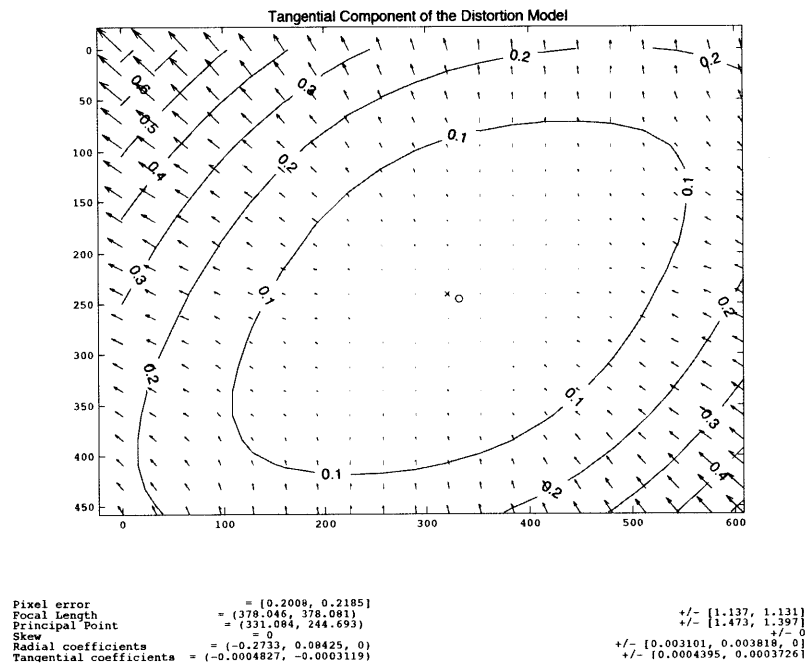


Figure 3-7: Effects of Tangential Distortion

### 3.2.4 Camera Calibration

The goal of the camera calibration process is to calculate the intrinsic camera values as well as the distortion coefficients. By using a known structure, specifically a chessboard, it is possible to estimate these values. This is done effectively by viewing

a known structure from multiple locations and orientations.

The points  $P_I$  and  $P_G$  are represented as homogenous coordinates which results in  $P'_I = (u, v, 1)$  and  $P'_G = (X, Y, Z, 1)$ . Equation 3.1 is represented as the  $3 \times 3$  matrix  $A$  known as the intrinsic camera matrix and is shown below. Once estimated, the values in matrix  $A$  can be reused without recalibrating the camera. These values only depend on the camera and lens used at the time of calibration and are independent of the scene viewed.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

In order to estimate the pose of the calibration object, the matrix  $[R|T]$  is used. The matrix  $[R|T]$  is known as the extrinsic camera matrix. This matrix is comprised of a rotation matrix ( $R$ ) appended with a translation vector ( $T$ ). Given the point  $P_G$ , this matrix is used to compute the corresponding point  $P_I$  in the imager coordinate frame. This physical transformation relates the plane in the real world to the imager plane. In computer vision literature, the projective mapping from one plane to another is defined as planar homography.

$$[R|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

The primary equation utilized for calibrating the camera is Equation 3.5.

$$P'_I = A[R|T]P'_G \tag{3.5}$$

The matrix  $A[R|T]$  is a  $3 \times 4$  matrix that can be simplified to the matrix  $C = [c_1, c_2, c_3]'$  where  $c_i$  is a  $1 \times 4$  row matrix. There are twelve unknowns which must be estimated. For each image taken, two equations are computed as shown in Equation 3.6. This results in only six images being needed to solve for the unknown parameters. In

practice, more than six images are taken which leads to an over determined set of equations solved using least squares.

$$\begin{aligned}u(c_3Z) - c_1X &= 0 \\v(c_3Z) - c_2Y &= 0\end{aligned}\tag{3.6}$$

### 3.2.5 Calibration Results

Extensive research has been done on methods to estimate the intrinsic as well as extrinsic camera properties along with the distortion coefficients. Specifically, the MATLAB camera calibration toolbox [14] and OpenCV libraries provide simple functions to analyze images and compute the desired variables.

A small chessboard pattern was the target used to create input images for the calibration process. These images can be seen in Figure 3-8. In total, nineteen

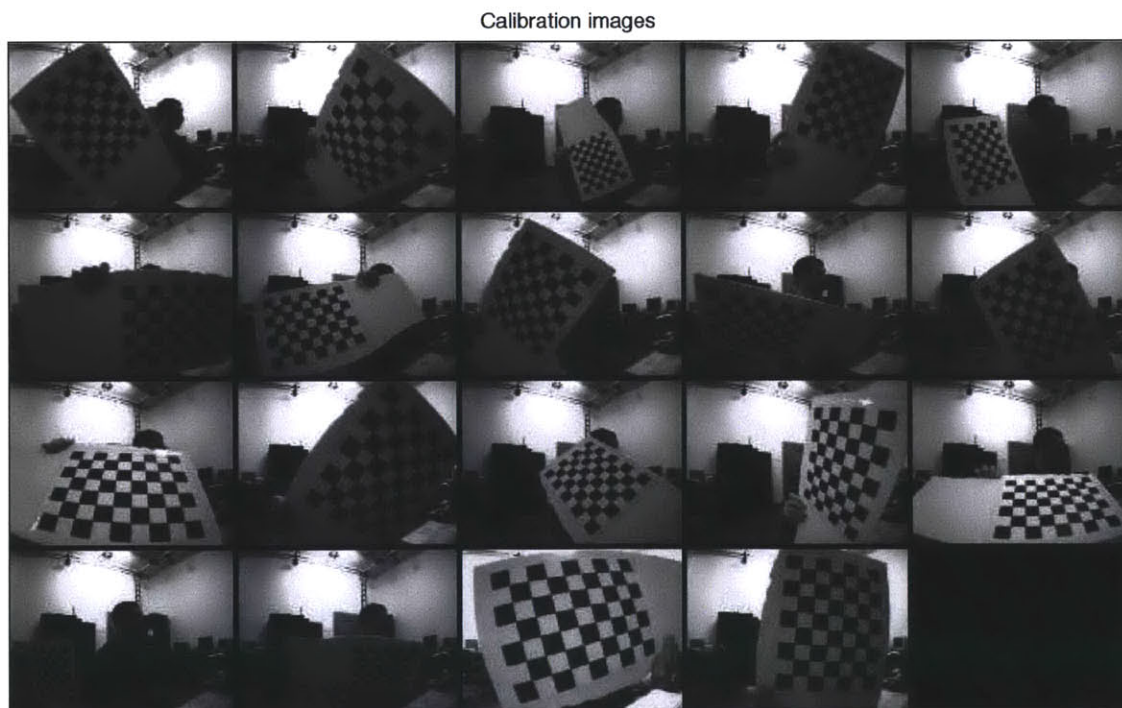


Figure 3-8: Chekerboard Images Used for Calibration

images were used for the calibration process. Both the MATLAB camera calibration

toolbox and OpenCV libraries were used to estimate the camera model parameters for comparison purposes as well as accuracy. Both methods identify the corners of the interior squares of the chessboard. They then utilize optimization techniques to minimize the reprojection errors. Sub-pixel accuracy is possible in both methods and in this calibration, the reprojection error is centered around zero with all errors less than one pixel.

As mentioned previously, it is possible to calculate the rotation and translation matrix ( $[R|T]$ ) relating the object plane to the image plane. Figure 3-9 visualizes these estimates by reprojecting the location of the target relative to the fixed camera.

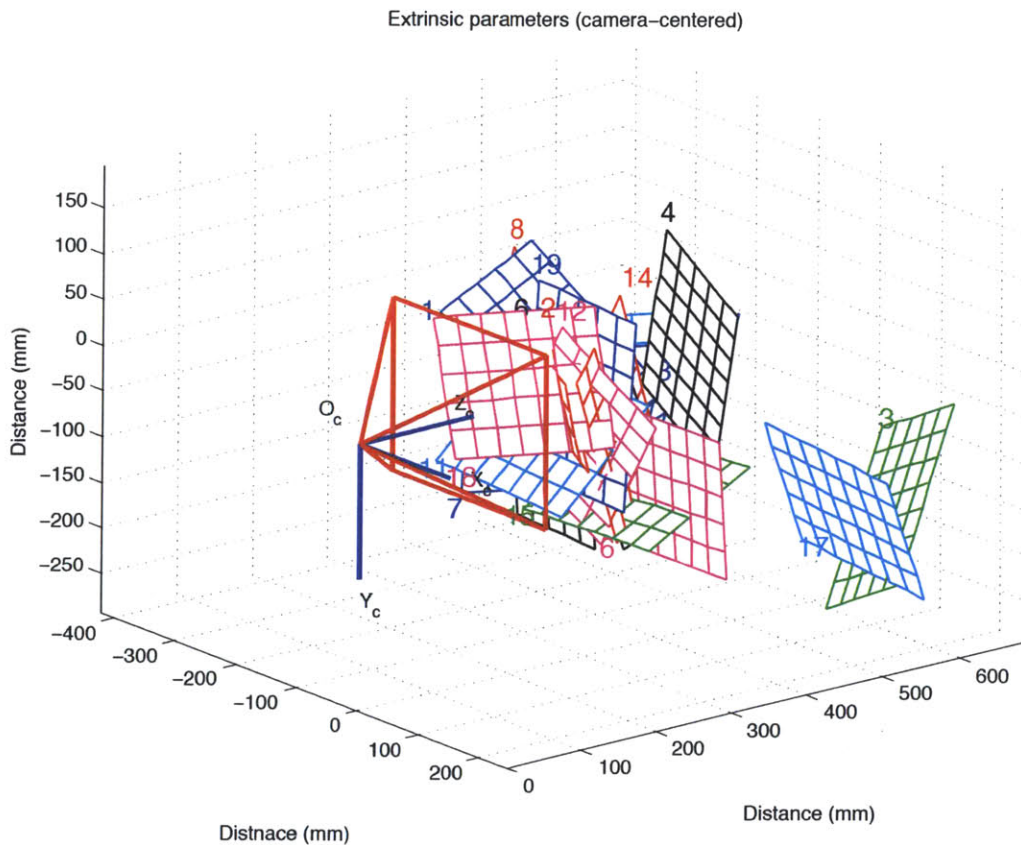


Figure 3-9: Camera Centered View of Estimated Checkerboard Locations

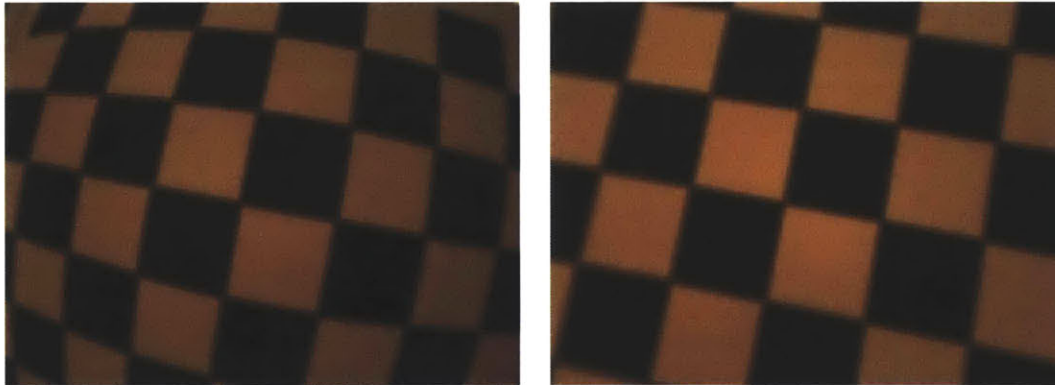
Table 3.1 displays the outputs of both calibration methods. The intrinsic parameters match very closely while the distortion parameters have a slight variation. For this research, the OpenCV values were used due to the ease of integrating the calibration functions into the existing codebase. It is also possible to validate some of the

Table 3.1: Comparison of Camera Calibration Methods

Parameter	MATLAB Toolbox	OpenCV Library
$(f_x, f_y)$	(378.05,378.08)	(378.16,378.09)
$(c_x, c_y)$	(331.08,244.70)	(329.47,245.12)
$[k_1, k_2]$	[-0.273, 0.084]	[-0.260,0.063]
$[p_1, p_2]$	$[-4.827e^{-4}, -3.112e^{-4}]$	$[-3.332e^{-4}, 9.67e^{-5}]$

assumptions made during the development of the model. First, it was assumed that the pixel axes were orthogonal. If a skew parameter is estimated, the resulting angle between the image axes is  $90.01^\circ$ . Additionally, if an extra term in the Taylor series expansion for radial distortion is estimated, the value ( $k_3$ ) is only  $-4.5e^{-4}$ . These parameters are ignored during the calibration process and assumed to be zero.

Figure 3-10(a) is a sample image of the checkerboard calibration pattern. The radial distortion is clear near the image edges while the tangential distortion is less apparent. These distortions can be removed through the calibration process which produces the image in Figure 3-10(b). The checkerboard pattern here contains orthogonal lines which are no longer curved due to distortions. This results in a more accurate mapping from pixel coordinate frame to world coordinate frame.



(a) Raw Camera Image

(b) Corrected Camera Image

Figure 3-10: Raw Image Before and After Distortion Correction

### 3.2.6 Camera Settings Configuration

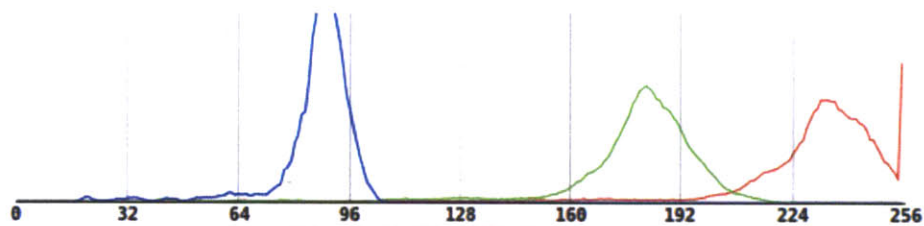
Modeling the physical properties of the camera helps account for the distortions caused by the physical parameters of the camera and lens. However, there are additional software adjustments that can improve the quality of image that is used for



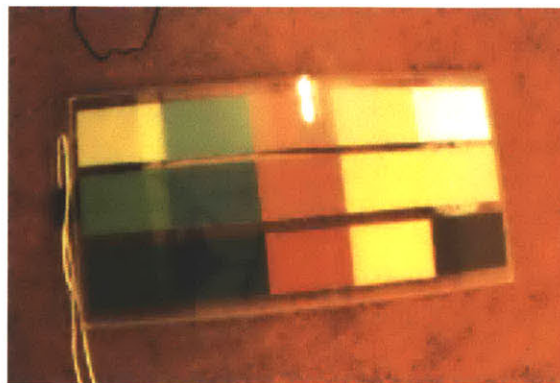
processing. Objects appear differently under daylight compared to a room with halogen lighting. White balancing is the process of manipulating the gains of the color channel through the diagonal matrix  $\mathbf{J}$  to account for these changes.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} J_R & 0 & 0 \\ 0 & J_G & 0 \\ 0 & 0 & J_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Some cameras come with preset modes to adjust the  $\mathbf{J}$  matrix. Others automatically adjust the  $\mathbf{J}$  matrix by focusing the camera on a white object. The goal of white balancing is to adjust the  $\mathbf{J}$  matrix so that tristimulus values are equal ( $R' = G' = B'$ ) when looking at a white object. The Firefly MV2 allows the user to adjust the Blue and Red gains each time the camera is initialized. The default values produce the histogram shown in Figure 3-11(a) and the image shown in Figure 3-11(b).



(a) Default Tristimulus Histogram

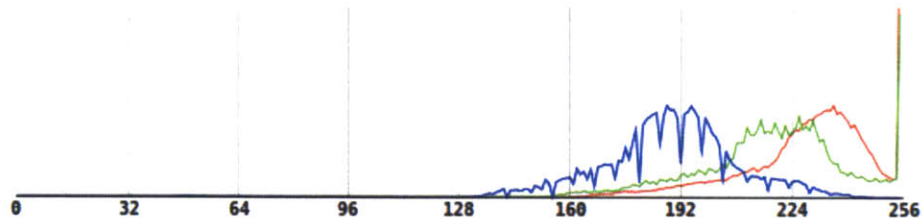


(b) Default Camera Image

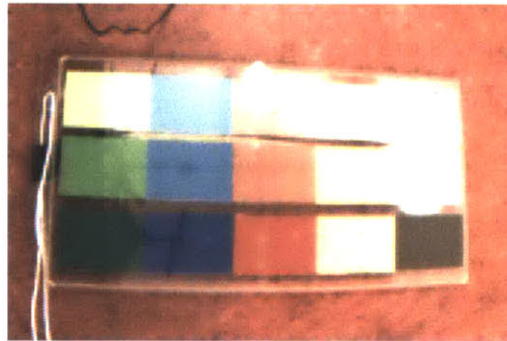
Figure 3-11: Effects of Default White Balance Values

To compensate for the lighting indoors, the  $\mathbf{J}_R$  value was increased and the  $\mathbf{J}_B$

value was decreased producing the histogram in Figure 3-12(a) and the image in Figure 3-12(b). These histogram figures represent the tristimulus values looking at a blank sheet of white paper. After the change in white balance values, the peaks of the tristimulus histograms are of similar magnitudes and the resulting image has the desired white chromaticity.



(a) Corrected Tristimulus Histogram



(b) Corrected Camera Image

Figure 3-12: Effects of Corrected White Balance Values

Monitors produce luminance values as a nonlinear function of the control voltage where  $L$  is the luminance,  $V$  is the control voltage and  $\gamma \approx 2.2$ .

$$L = V^\gamma$$

To correct for this, cameras use the inverse nonlinear operation known as gamma encoding. This produces a linear system from image capture to display. Fortunately, the Point Grey camera has the ability to enable or disable the gamma encoding. In order to get the best results possible, gamma encoding is disabled such that the tristimulus values in the image are the raw unmodified values. Finally, the camera supports  $640 \times 480$  resolution at sixty frames per second. This frame rate is set each time the camera is initialized to ensure the most recent image is being processed.

### 3.3 Quadrotor Dynamic Model

The quadrotor vehicle operates on the concept of variable torques and thrusts. The motors are arranged in pairs along the horizontal ( ${}^b y$ ) and vertical ( ${}^b x$ ) axes with the forward pair rotating clockwise and the horizontal pair rotating counter-clockwise. This design results in the reaction torques from the pairs of motors being exactly opposed by each other. The elimination of the rotating moment allows the vehicle to maintain a constant heading while hovering. Yaw is controlled by varying the speeds of the pairs of motors to create a non-zero net counter torque. Altitude is controlled by varying the thrust from each motor by equal amounts to provide a net thrust vector and no rotational moment. To move in the lateral  ${}^b x$  and  ${}^b y$  directions, the relative speeds of each motor in the pair are varied to create a desired lateral thrust offset. The simple design results in small platforms which are battery operated, able to perform a stable hover, and safe to use in indoor environments.

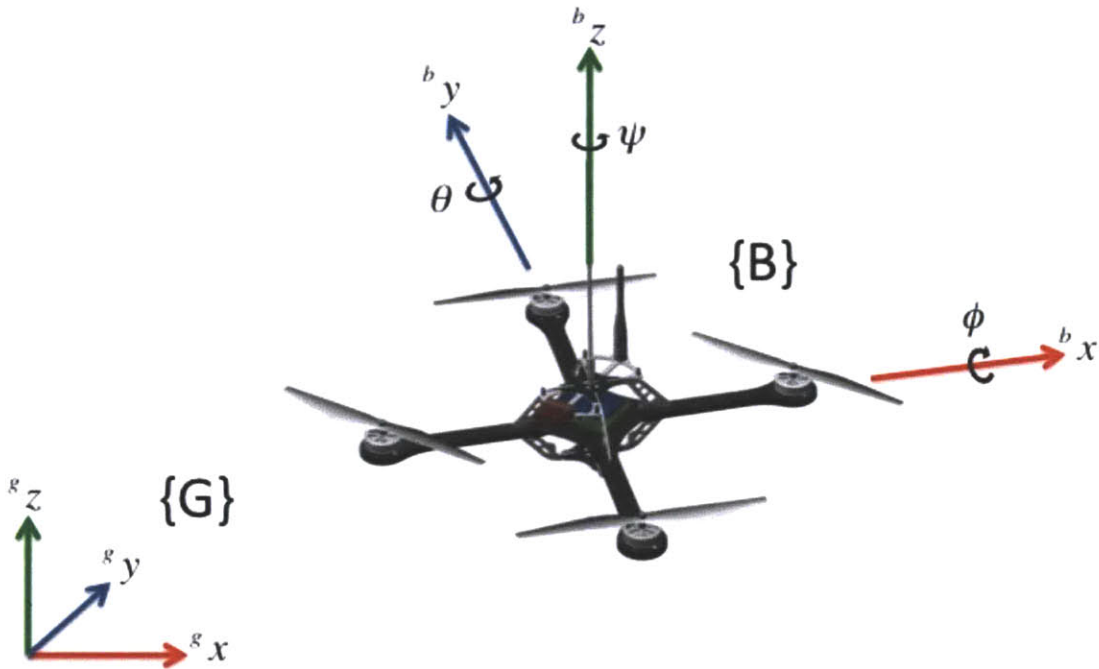


Figure 3-13: Quadrotor Structural Diagram

The basic quadrotor structure used for the model development is shown in Figure 3-13 depicting the Euler angles of roll ( $\phi$ ), pitch ( $\theta$ ), and yaw ( $\psi$ ), a body coordinate



frame  $[{}^b x, {}^b y, {}^b z]$ , and the global coordinate frame  $[{}^g x, {}^g y, {}^g z]$ . This model relies on several assumptions [13]. First, the quadrotor structure is rigid and symmetrical with a center of mass aligned with the center of the body frame of the vehicle. It is also assumed that the force of each motor is proportional to the square of the motor velocity. Additionally, the propellers are considered to be rigid. An estimated model is compared with ground truth accelerations in Section 3.3.1 to validate the model as well as the assumptions.

First, the coordinate system is divided into a global reference frame ( $G$ ) and a body reference frame ( $B$ ) as shown in Figure 3-13.

$$\begin{aligned} m {}^g \ddot{\mathbf{X}} &= \mathbf{F}_f + \mathbf{F}_t + \mathbf{F}_g \\ \mathbf{J} \ddot{\boldsymbol{\Omega}} &= \boldsymbol{\Gamma}_f + \boldsymbol{\Gamma}_a \end{aligned} \quad (3.7)$$

Here,  $m$  is the mass of the quadrotor and  ${}^g \mathbf{X}$  is the position of the vehicle's center of mass in relation to the body reference frame. Additionally  $\mathbf{J}$  is a constant, symmetric, and positive definite rotational inertia matrix with respect to the body frame of the quadrotor defined by Equation 3.8 and  $\boldsymbol{\Omega}$  is the angular acceleration of the quadrotor in the body fixed frame.

$$\mathbf{J} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (3.8)$$

The rotational transformation matrix between the global reference frame and the body reference frame is defined by matrix  $\mathbf{R}_{\mathbf{G}}^{\mathbf{B}}$  where  $C$  and  $S$  represent the cosine and sine trigonometric functions and the angles are defined in the body coordinate frame.

$$\mathbf{R}_G^B = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\psi S_\phi \\ C_\theta S_\psi & S_\psi S_\theta S_\phi - C_\phi C_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}$$

$\mathbf{F}_f$  is the resultant force generated by the four motors in the global coordinate frame defined below.

$$\mathbf{F}_f = \mathbf{R}_G^B \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 F_i \end{bmatrix}$$

The force of each motor,  $F_i$ , is defined in equation 3.9 as the product of the lift coefficient  $K_p$  and the square of the motor's angular velocity  $\omega$ . This depicts the relationship between motor speed and resultant force, which is the basis for stable control of the vehicle.

$$F_i = K_p \omega^2 \quad (3.9)$$

Additionally,  $\mathbf{F}_t$  is the resultant global drag force along each translational axis defined below where  $K_{ftx}$ ,  $K_{fty}$ , and  $K_{ftz}$  are drag coefficients.

$$\mathbf{F}_t = \begin{bmatrix} -K_{ftx} & 0 & 0 \\ 0 & -K_{fty} & 0 \\ 0 & 0 & -K_{ftz} \end{bmatrix} {}^g \dot{\mathbf{x}}$$

$\mathbf{F}_g$  is the gravity force vector.

$$\mathbf{F}_g = \begin{bmatrix} 0 & 0 & -mg \end{bmatrix}^T$$

$\mathbf{\Gamma}_t$  is the moment developed by the quadrotor in the body frame expressed below. The distance from the vehicle's center of mass to the motor is defined by  $l$  and  $K_d$  is a rotational drag coefficient.

$$\mathbf{\Gamma}_t = \begin{bmatrix} l(F_3 - F_1) \\ l(F_4 - F_2) \\ K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}$$

Finally,  $\mathbf{\Gamma}_a$  is the resultant torques of aerodynamic friction where  $K_{fax}$ ,  $K_{fay}$  and  $K_{faz}$  are rotational drag coefficients.

$$\mathbf{\Gamma}_a = \begin{bmatrix} -K_{fax} & 0 & 0 \\ 0 & -K_{fay} & 0 \\ 0 & 0 & -K_{faz} \end{bmatrix} \dot{\theta}$$

The inputs to the traditional model derived above are the angular velocities of the four motors as shown in Equation 3.10.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K_p & K_p & K_p & K_p \\ -K_p & 0 & K_p & 0 \\ 0 & -K_p & 0 & K_p \\ K_d & -K_d & K_d & -K_d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (3.10)$$

This results in the following equations of motion described in Equation 3.11.

$$\begin{aligned} g\ddot{x} &= \frac{(C_\phi S_\theta C_\psi + S_\phi S_\psi)u_1 - K_{ftx}{}^g \dot{x}}{m} \\ g\ddot{y} &= \frac{(C_\phi S_\theta C_\psi - S_\phi C_\psi)u_1 - K_{f ty}{}^g \dot{y}}{m} \\ g\ddot{z} &= \frac{(C_\phi C_\theta)u_1 - K_{ftz}{}^g \dot{z}}{m} - g \\ \ddot{\phi} &= \frac{l u_2 - K_{fax} \dot{\phi}}{I_x} \\ \ddot{\theta} &= \frac{l u_3 - K_{fay} \dot{\theta}}{I_y} \\ \ddot{\psi} &= \frac{u_4 - K_{faz} \dot{\psi}}{I_z} \end{aligned} \quad (3.11)$$

### 3.3.1 Model Adaption

Utilizing the Ascending Technologies platform has several distinct advantages. Primarily, we can rely on the on-board AutoPilot system to control the attitude of the quadrotor. The AutoPilot system computes the desired motor angular velocities  $([u_1, u_2, u_3, u_4]^T)$  as defined in Equation 3.10. Motor voltages are computed by the on-board motor controllers. This allows the user to focus on position control. The AutoPilot inputs are a desired roll and pitch angle, desired yaw rate, and a generalized thrust command. In order to create an effective position controller, it is necessary to modify the mathematical model of the quadrotor to reflect the true nature of our system. This model is developed using MATLAB's Simulink toolbox.

The rotational dynamics are modeled first. The roll and pitch dynamics are modeled as a first order system to take into account the effects of the on-board attitude controller. The equations of motion are shown in Equation 3.12.

$$\begin{aligned}\dot{\phi} &= \frac{K_\phi}{a_\phi} U_{\phi_d} - \frac{1}{a_\phi} \phi + C_\phi \\ \dot{\theta} &= \frac{K_\theta}{a_\theta} U_{\theta_d} - \frac{1}{a_\theta} \theta + C_\theta\end{aligned}\tag{3.12}$$

For these dynamics, the roll and pitch angles ( $\phi$  and  $\theta$ ) are functions of the controller inputs ( $U_{\phi_d}$  and  $U_{\theta_d}$ ), drag coefficients ( $K_\phi$  and  $K_\theta$ ), and linear offsets ( $C_\phi$  and  $C_\theta$ ). When flown manually, the roll and pitch inputs come from an RC receiver and can range from  $\pm 2047$ . These values must be converted to a desired roll and pitch angle for the on-board attitude controller. In Equation 3.12, this transformation is calculated using the values  $\frac{K_\phi}{a_\phi}$  and  $\frac{K_\theta}{a_\theta}$ . By taking this transformation into account, our controller computes inputs in the native form of the on-board attitude controller.

The on-board attitude controller takes a desired yaw rate as an input. The yaw dynamics are modeled by Equation 3.13. Again,  $K_\psi$  is a gain and  $C_\psi$  a linear offset.

$$\dot{\psi} = K_\psi U_{\psi_d} + C_\psi\tag{3.13}$$

The Simulink block diagram representation of the rotational dynamics model is shown in Figure 3-14. The angles were saturated at forty-five degrees.

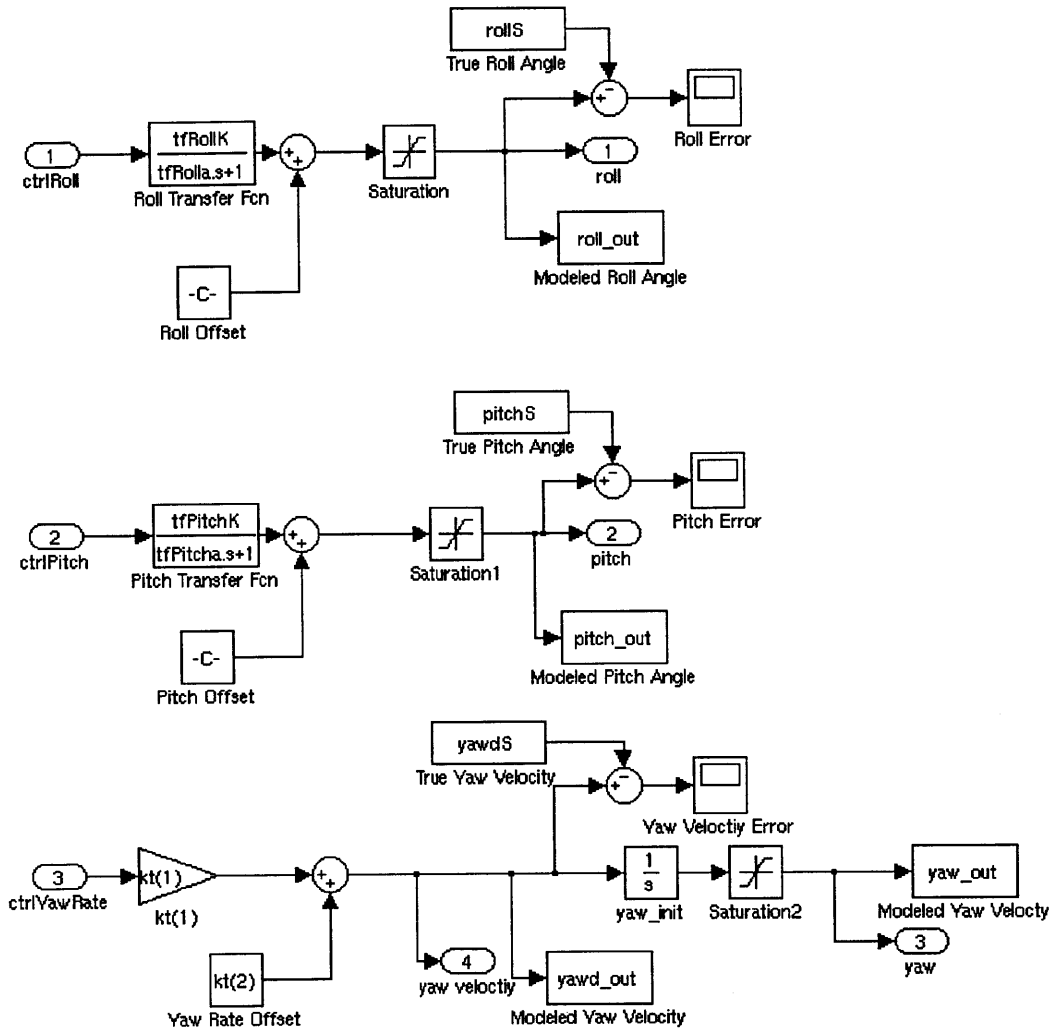


Figure 3-14: Rotational Dynamics Block Diagram

The translational dynamics are shown in Equation 3.14.

$$\begin{aligned}
 {}^b\ddot{x} &= K_p\theta - K_{\dot{x}}{}^b\dot{x} + C_x \\
 {}^b\ddot{y} &= K_r\phi - K_{\dot{y}}{}^b\dot{y} + C_y \\
 {}^b\ddot{z} &= \frac{1}{m}(K_zU_z - K_{\dot{z}}{}^b\dot{z} + C_z) - g
 \end{aligned}
 \tag{3.14}$$

The inputs for  ${}^b\dot{x}$  and  ${}^b\dot{y}$  translational motion are the roll and pitch angles computed

by the rotational dynamics. For the  $b_z$  axis, the input is from the RC receiver or position controller. These equations all contain a gain on the input, drag coefficient, and offset. The mass of the vehicle is equal to one kg and represented by the variable  $m$ . The variable  $g$  is the acceleration due to gravity. The block diagram representation of the quadrotor's translational dynamics are shown in Figure 3-15.

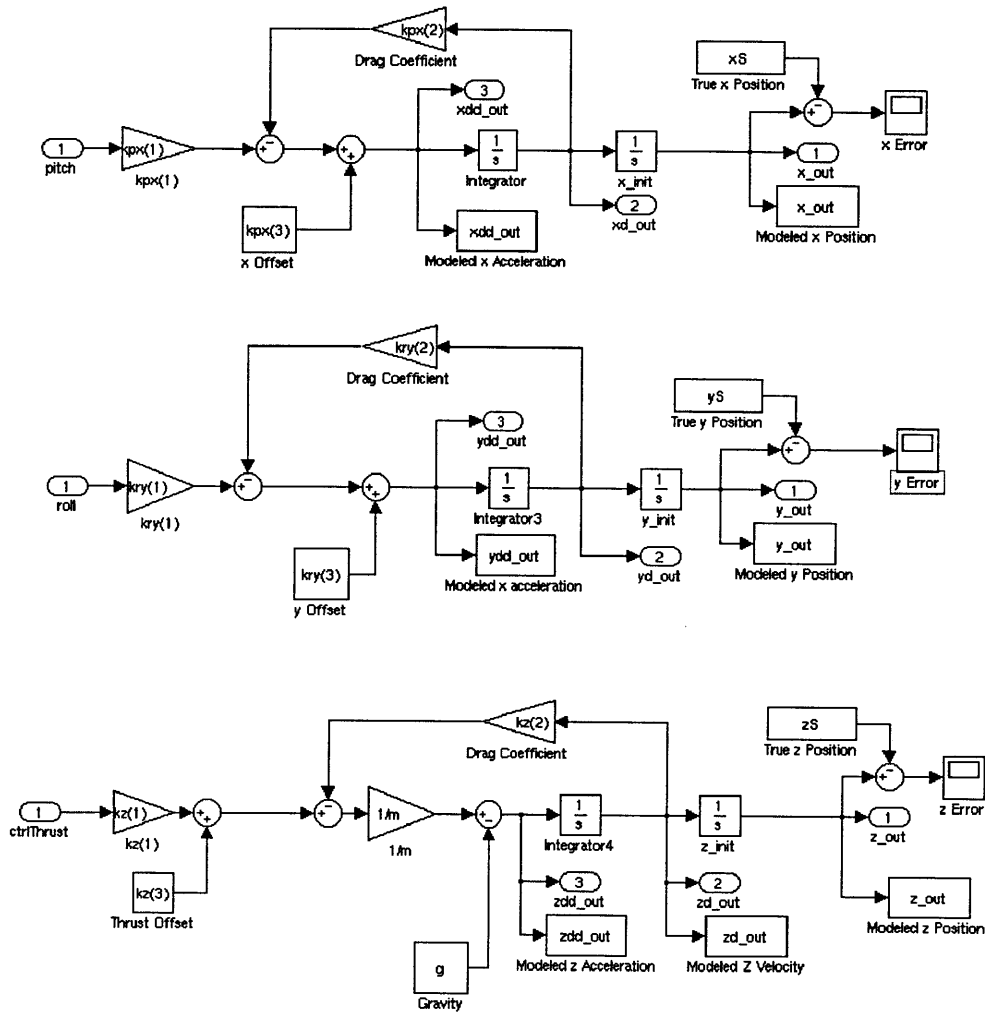


Figure 3-15: Translational Dynamics Block Diagram

### 3.3.2 Parameter Estimation and Model Verification

Using the model developed above in Simulink, it was possible to utilize experimental data to estimate the model gains, drag coefficients, and linear offsets. Simulink provides a Parameter Estimation toolbox which uses least squares optimization to

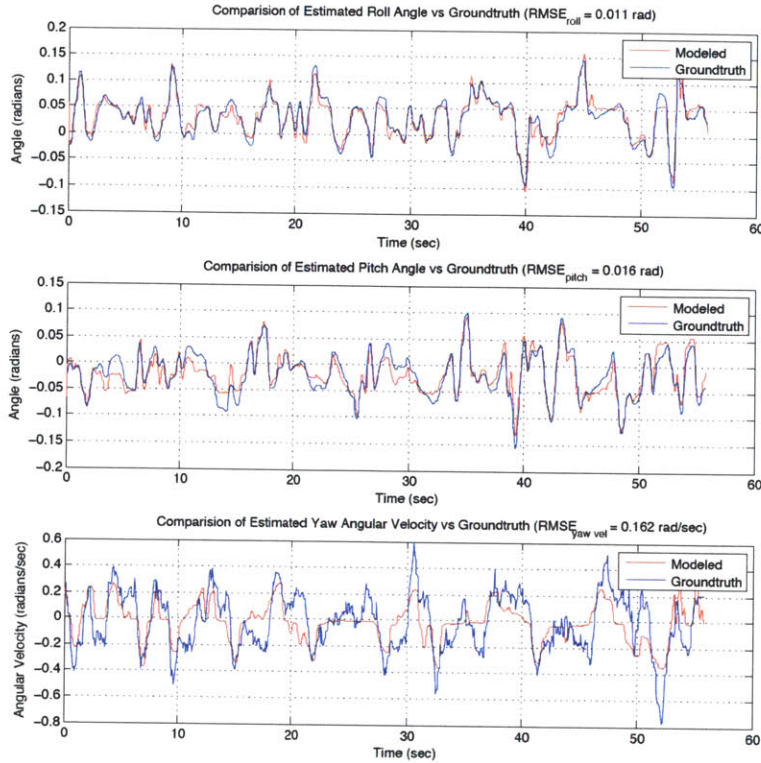


Figure 3-16: Comparison of Simulated Rotational Dynamics and Ground Truth

approximate user defined parameters. Additionally, the System Identification toolbox was used to calculate the transfer functions which estimated the on-board roll and pitch controllers. The quadrotor was flown manually and the ground truth position and attitude was recorded by a Vicon motion capture system. Additionally, the RC inputs were recorded. Using RC data as inputs and the motion capture data as ground truth output, the model parameters are estimated. A comparison of the modeled roll and pitch angles as well as the yaw velocity is shown in Figure 3-16 as well as the translational accelerations in Figure 3-17. In practice, this model was sufficient for controller design, tuning, and testing despite these assumptions.

Several factors affected the performance of the model estimation process. The quadrotor could only be flown inside the footprint of the motion capture system which limited the maneuvers of the quadrotor. The yaw angle was manually controlled to be near zero degrees in order to make the quadrotor easier to fly. It was also kept at a fairly stable height for safety purposes. Combined, this resulted in the possibility

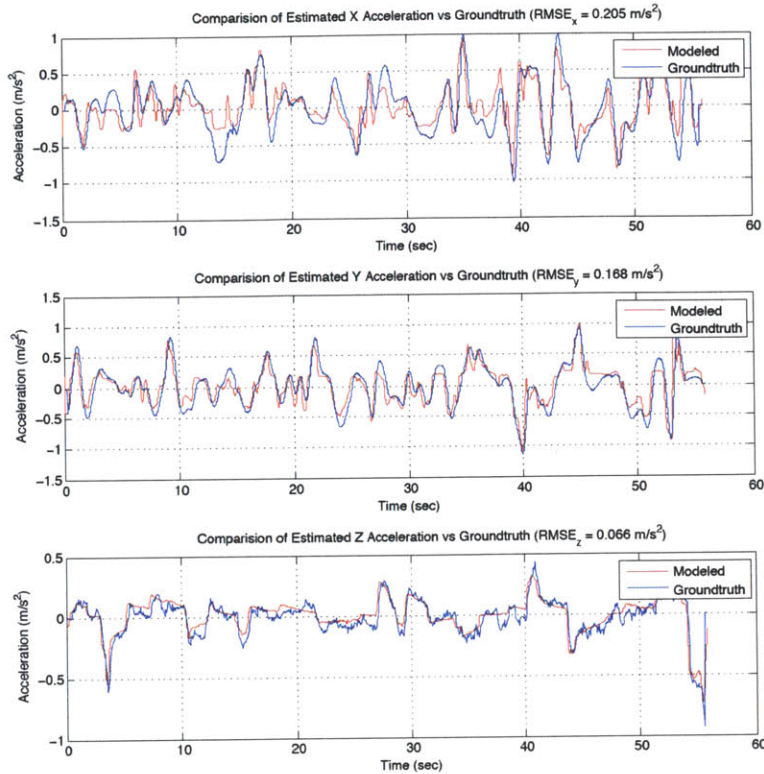


Figure 3-17: Comparison of Simulated Translational Dynamics and Ground Truth

that the test run did not excite the full range of dynamics of the system. With a larger footprint and better pilot, a more rigorous path could be recorded to improve the model.

Additionally, the assumptions made in the model derivation could have a negative impact on the model's performance. The quadrotor alone is very symmetrical but with the additional payload of the camera and computer, this symmetry is adversely affected. The additional payload also shifts the center of mass which must be approximated. This assumption is the most likely source of error in the model. Aerodynamic effects due to blade flapping could be added, but we believe these effects would be small in comparison when the quadrotor is operated around a stable hover. Finally, any error in the roll or pitch angles are reflected as errors in  ${}^b x$  and  ${}^b y$  accelerations since the roll and pitch angles are inputs to these systems.



### 3.4 Extended Kalman Filter

While it is possible to estimate relative position of the quadrotor through double-integrating the linear acceleration measurements, the on-board MEMS IMU is of too poor quality for this method to be effective. The IMU is subject to biases which result in drifts in velocity and position as the biases are integrated. This is shown in Figure 3-18 where after ten seconds, the true and estimated velocities have drifted by 0.12 m/s and position estimates would have drifted even more.

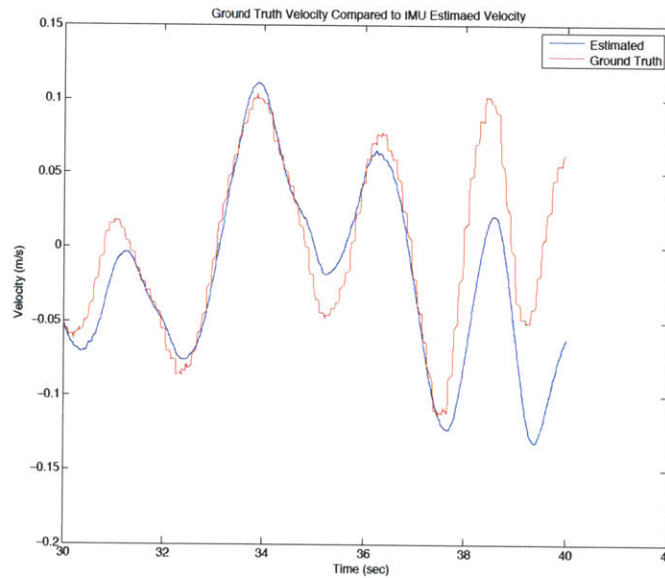


Figure 3-18: Integration Errors Due to IMU Drift

This motivates the necessity for an exteroceptive sensor, specifically a camera for this thesis, in order to estimate the relative position of the quadrotor accurately. An Extended Kalman Filter (EKF) is used to correct for errors in integrated position estimates from the IMU with position estimates from the vision system. Between vision updates, the filter utilizes the higher frequency IMU readings to provide a state estimate at a servoable rate. Most importantly, the filter needs to provide an accurate velocity signal. The under-damped dynamics of the quadrotor implies that proportional-derivative-like control must be used to hover the quadrotor stably and prevent oscillations. For the derivative-like term to be effective, a smooth, accurate and minimally delayed velocity estimate must be computed. Measurement delays

quickly destabilize the fast dynamics of the quadrotor.

The EKF is adapted extensively and directly from [1, 7] which was implemented using the KFilter library [93]. This filter combines position estimates from the vision system algorithms as well as attitude and acceleration measurements from the IMU. While the IMU measurements are received at a frequency of thirty Hz, the vision system outputs position estimates at ten Hz. The filter has to deal with these asynchronous measurements and the inherent latencies in these measurements. While the IMU measurements are noisy and include a bias that drifts over time, the slower position estimates from the vision system are more accurate and without a varying bias. The covariance matrices are weighted to favor the vision system position estimates and utilize the IMU measurements to estimate the quadrotor’s position between vision system updates. This improves the quality of the velocity estimates which are necessary and sufficient for precise position control of the vehicle.

### 3.4.1 Process Model

The filter is used to estimate the quadrotor’s positions, velocities, accelerations, attitude, and the biases in the IMU. The state vector is

$$\mathbf{X} = [{}^g x, {}^g y, {}^g z, \phi, \theta, \psi, {}^b \dot{x}, {}^b \dot{y}, {}^b \dot{z}, \dot{\psi}, {}^b \ddot{x}, {}^b \ddot{y}, {}^b \ddot{z}, \beta^{\dot{x}}, \beta^{\dot{y}}, \beta^{\dot{z}}, \beta^{\phi}, \beta^{\theta}]^T \quad (3.15)$$

The positions  $({}^g x, {}^g y, {}^g z)$  are represented in a global coordinate frame with the origin placed at the center of the target, the positive  $x$  axis pointing forward and the positive  $y$  axis pointing left of the target origin. Attitude  $(\phi, \theta, \psi)$  is represented by the Euler angles, roll, pitch, and yaw respectively. The rest of the state including the velocities and accelerations are represented in the body frame with the origin located at the center of mass of the quadrotor. The biases are represented by the variable  $\beta$  with a corresponding subscript.

The dynamic model used in the filter makes several assumptions. Primarily, that the vehicle will be kept level due to the attitude controller and we can therefore assume small attitude angles. This allows us to estimate the heading of the vehicle

separate from pitch and roll and also results in the  ${}^g z$  axis becoming independent of the  ${}^g x$  and  ${}^g y$  axes. The  ${}^g x$  and  ${}^g y$  axes are calculated utilizing the nonlinear functions

$$\begin{aligned} {}^g x_t &= {}^g x_{t-1} + \Delta t ({}^b \dot{x}_{t-1} \cos(\psi_{t-1}) - {}^b \dot{y}_{t-1} \sin(\psi_{t-1})) + w_x \\ {}^g y_t &= {}^g y_{t-1} + \Delta t ({}^b \dot{y}_{t-1} \sin(\psi_{t-1}) + {}^b \dot{x}_{t-1} \cos(\psi_{t-1})) + w_y \end{aligned} \quad (3.16)$$

where  $\Delta t$  is the filter update period and the terms  $w_x \sim N(0, \sigma_x)$  and  $w_y \sim N(0, \sigma_y)$  are zero-mean Gaussian noise variables.

The remaining position and velocity variables are updated by discrete integration where  $v = [{}^g z, \phi, \theta, \psi, {}^b \dot{x}, {}^b \dot{y}, {}^b \dot{z}]^T$  and  $w_v \sim N(0, \sigma_v)$ .

$$v_t = v_{t-1} + \dot{v}_{t-1} \Delta t + w_v \quad (3.17)$$

The linear accelerations are also computed as discrete integrations for several reasons. The quadrotor is a highly nonlinear system with fast dynamics and aerodynamic effects that are hard to characterize. A simple linear model is developed through a system identification process that relates attitude to translational accelerations. This model is slightly modified from the model used in Section 3.3.1 for the controller design. For the simplified EKF model, only the  ${}^b x$  and  ${}^b y$  accelerations are modeled. Also, the roll and pitch angles which serve as inputs to these equations are taken from the IMU. The model we present in Section 3.3.1 maps the position controller inputs through the on-board attitude controller to estimate the quadrotor attitude. Removing this error results in a slightly more accurate model of the quadrotor's translational accelerations.

In practice however, the acceleration estimates from the filter were as accurate as the modeled accelerations. Figure 3-19 shows the ground truth accelerations, as well as the accelerations from the EKF and from the simplified model. The Root Mean Square (RMS) errors in each axis are roughly equivalent. In the  $x$  axis, the EKF had a RMS error of 0.114 mm/s<sup>2</sup> and the model's RMS error was 0.112 mm/s<sup>2</sup>.

In the  $y$  axis, the EKF RMS error was  $0.113 \text{ mm/s}^2$  and the model RMS error was  $0.121 \text{ mm/s}^2$ . This validates the accuracy of the simplified model as well as the use of discrete dynamics in the EKF.

Lastly, the  ${}^b\ddot{z}$  acceleration term as well as the biases are modeled as a random walk where  $b = [{}^b\dot{z}, \beta^{\ddot{x}}, \beta^{\ddot{y}}, \beta^{\ddot{z}}, \beta^\phi, \beta^\theta]^T$ .

$$\begin{aligned} b_t &= b_{t-1} + w_b, \\ w_b &\sim N(0, \sigma_b) \end{aligned} \tag{3.18}$$

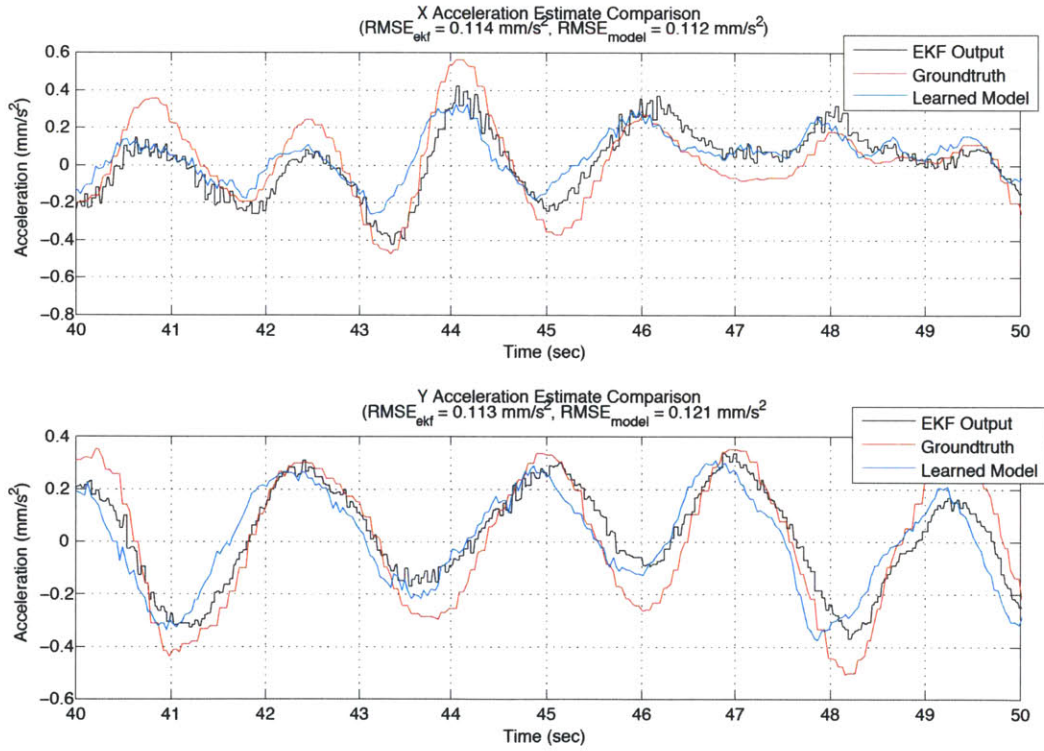


Figure 3-19: Comparison of Acceleration Estimates and Ground Truth Values

### 3.4.2 Measurement Model

Due to the asynchronous nature of the measurements, each sensor has its own measurement model. The gravity vector is subtracted from the IMU measurements before being sent to the filter. The biases are estimated online. The IMU measurement model includes the attitude angles and linear accelerations plus the biases and zero-mean

Gaussian noise. This is represented as the vector  $z_t^{IMU}$  where  $v_{IMU} \sim N(0, \sigma_{IMU})$ .

$$z_t^{IMU} = \begin{bmatrix} \ddot{x}_t^b + \beta_t^{\ddot{x}} \\ \ddot{y}_t^b + \beta_t^{\ddot{y}} \\ \ddot{z}_t^b + \beta_t^{\ddot{z}} \\ \phi_t + \beta_t^\phi \\ \theta_t + \beta_t^\theta \\ \psi_t \end{bmatrix} + v_{IMU}$$

The vision algorithm estimates the translational position to the target centroid as the vector  $z_t^{CV}$  where  $v_{CV} \sim N(0, \sigma_{CV})$ .

$$z_t^{CV} = \begin{bmatrix} {}^g x_t \\ {}^g y_t \\ {}^g z_t \end{bmatrix} + v_{CV}$$

Using only a single camera makes altitude estimation especially noisy and inaccurate without filtering. The height is calculated as a function of the size in pixels of the target as shown in Equation 3.19. This is calibrated by reading the target area in pixels at a known height. In the equation below,  $\bar{P}$  is the pixel area at the calibrated height,  $\hat{p}$  is the current pixel area of the target, and  $h_c$  is the calibration height.

$$\hat{z} = h_c \sqrt{\frac{\bar{P}}{\hat{p}}} \quad (3.19)$$

Figure 3-20 shows the vision system altitude estimate in light blue, the EKF altitude estimate in black, and the ground truth altitude in blue. The EKF output filters the noisy vision altitude approximations and returns an estimated altitude with a RMS error of 0.025 m.

Since the camera is rigidly attached to the quadrotor, changes in attitude cause

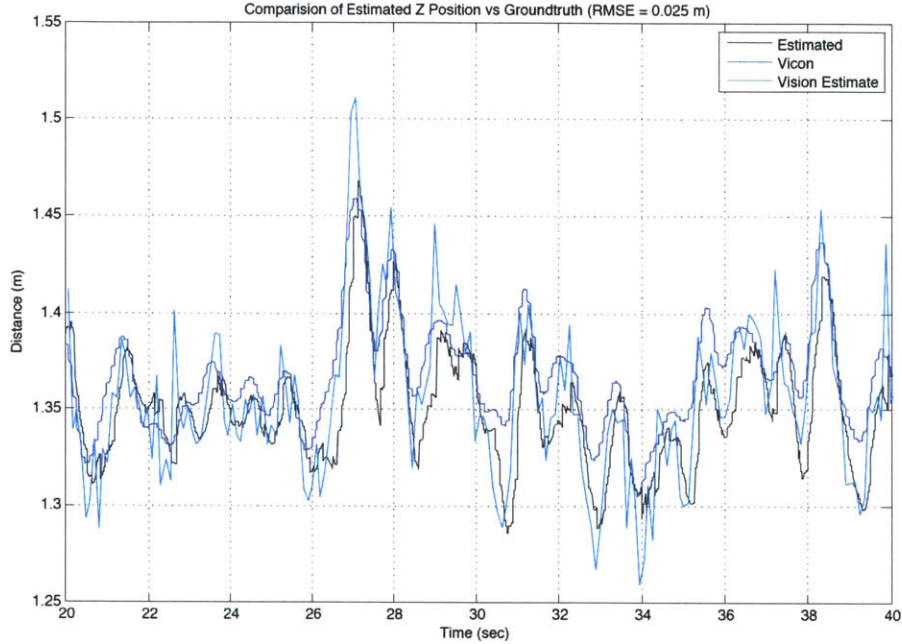


Figure 3-20: Comparison of Altitude Estimates and Ground Truth Values

the target to move in the image plane. This is compensated by Equation 3.20.

$$\begin{aligned} x_c &= x_e - \theta \left( \frac{I_W}{FOV_h} \right) \\ y_c &= y_e - \phi \left( \frac{I_H}{FOV_v} \right) \end{aligned} \quad (3.20)$$

This equation is a function of the uncompensated input position estimate in pixels  $(x_e, y_e)$ , the roll and pitch angles ( $\theta$  and  $\phi$ ), the dimensions of the input image in pixels ( $I_W$  and  $I_H$ ), and the horizontal and vertical fields of view of the camera ( $FOV_h$  and  $FOV_v$ ).

The camera is not mounted below the center of mass of the vehicle so a translation is applied to transform the image coordinate frame origin to the robot coordinate frame origin.

Finally, the position estimate is scaled from pixels to meters

$$\begin{aligned} x_m &= x_c \frac{\tan\left(\frac{FOV_v}{2}\right)}{\frac{I_H}{2}} \hat{z} \\ y_m &= y_c \frac{\tan\left(\frac{FOV_h}{2}\right)}{\frac{I_W}{2}} \hat{z} \end{aligned} \quad (3.21)$$

using the current height estimate  $\hat{z}$  and the previously defined variables as shown in Equation 3.21.

Diagonal process and measurement covariance matrices are chosen to limit the number of tuning parameters. The initial parameters were estimated by recording sensor measurements in a static environment. Using logged data, experimental runs were replayed through the filter and the estimated states were compared to the ground truth values. The covariance parameters were adjusted empirically until acceptable performance was observed over multiple sets of test data.

THIS PAGE INTENTIONALLY LEFT BLANK



# Chapter 4

## Object Identification and Tracking

This chapter presents the primary computer vision algorithms which first characterize the intended target in the hue (H) and saturation (S) planes and then identifies the intended target in subsequent frames based on this characterization. The chapter begins with an explanation of the decision to process the image in the HSV color space instead of the RGB color space. Next, our binary pixel-level classifier is presented which assigns each pixel a target or non-target label. The image segmentation algorithm is then described which identifies groups of target pixels in the image by computing the contours of the target pixel regions. Finally, the groups of pixels are described in terms their of size and position to provide high level control of the quadrotor. Using these algorithms, we produce a vision system that is able to track multiple images of varying hue (H) and saturation (S) characteristics at a servoable frame rate on a computationally limited platform. Figure 3-4 graphically depicts the complete vision system. In order take decrease the development time of the algorithms and to take advantage of the most sophisticated image processing techniques, the OpenCV libraries are used.

Robustness is key when dealing with image segmentation of real world objects. The ability to successfully identify the target over time has several underlying assumptions. Primarily, it is assumed that the intended target is homogenous with respect to the characteristic features chosen. This research segments pixels based on their H and S values. It is assumed that the H and S values that describe the target

remain constant, but in an outdoor environment this is rarely true as shadows and lighting changes affect the image. Recognizing the fragility of this assumption, additional processing steps are performed to remove misclassified pixels. The proposed algorithms also assume certain size characteristics in order to filter out errors in segmentation. Mainly, it is assumed that the intended target is large in the image frame. When tracking marine targets, it is possible for them to leave the surface of the water which would reduce their size in the image and make them difficult to successfully track.

All computation of these algorithms is executed on-board the quadrotor. However, unless the algorithms have been pre-configured to identify the target, user interaction is involved to tune the algorithms for the intended target. Once an image of the target has been captured, the user must alter the H and S threshold values to segment the target. The user is also able to modify several heuristic variables to improve the performance of the vision system. Once the system has been initialized, the algorithms are processed independent of the user.

## 4.1 Color Models

The mapping from RGB color space to HSV color space assumes that the RGB tristimulus values are proportional to scene luminance but in reality this is not often the case. RGB values are often offset, that is, for zero luminance the camera produces non-zero RGB values. The gain of the color channels can also be different, and this can vary continuously due to auto white balancing functions within the camera. Finally, a non-linear transformation is typically applied, the so-called gamma encoding described in Section 3.2.6. This is approximately a square root function, but the details vary from camera to camera. All of these variables contribute to sub-optimal HSV conversion which typically means that hue and saturation are not independent of overall luminance. For a particular camera these characteristics can be accounted for but for footage taken from the web this is outside our control. The hue of whales is close to white (grey is dark white) and this results in ambiguity with respect to white water from shore breaks or the wake of whales. The darkness of whales means that

the resolution of hue and saturation calculations is limited. The camera calibration process to account for these errors in our specific camera configuration is discussed in Section 3.2.4.

## 4.2 Pixel Classification

The primary task for identifying the intended object is to create a mathematical model of the target that can be applied in subsequent frames of the video file or camera output. This research requires a solution that is robust but also computationally efficient enough to run on the computationally impoverished on-board computer at servo rate. Algorithm 1 shows the process used to identify the object to be tracked.

The mathematical model used to represent whales is a two-dimensional histogram. First, an initial frame containing the target similar to Figure 4-1(a) is captured and converted from RGB color space to HSV color space. A two-dimensional histogram is computed which describes the probability distribution of the hue-saturation pair values within the image. Currently, the user is required to interactively select minimum

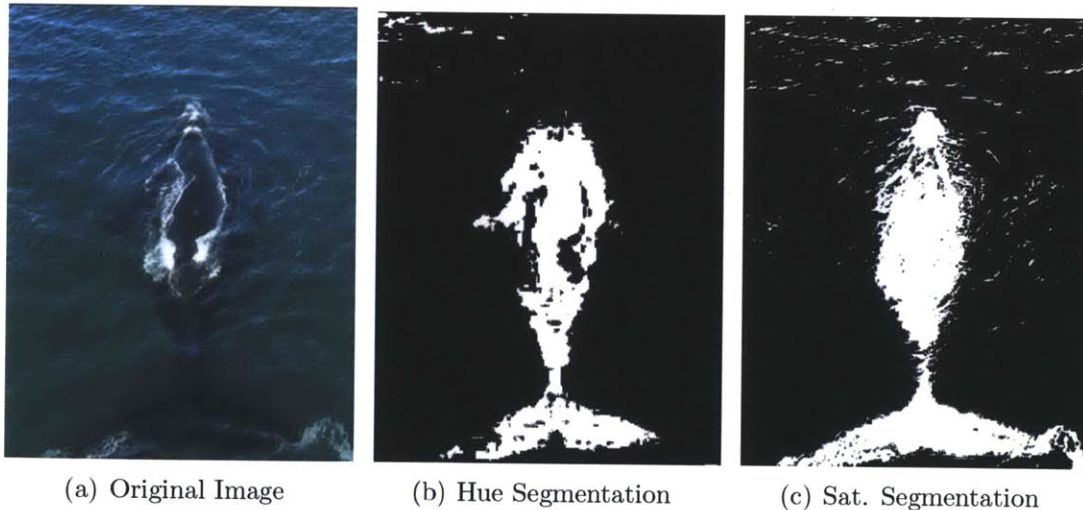


Figure 4-1: Image Thresholding in the Hue and Saturation Planes

and maximum threshold values for the H and S plane images using sliding track bars. These values are denoted as  $H_{min}$ ,  $H_{max}$ ,  $S_{min}$ , and  $S_{max}$ . The user can also input the number of bins in each histogram ( $H_{bins}$  and  $S_{bins}$ ) which determines the values of each bins' size ( $H_{width}$  and  $S_{width}$ ). A sample output of Algorithm 1 is shown for the H plane in Figure 4-1(b) and S plane in Figure 4-1(c) using the original image shown in

Figure 4-1(a). Both thresholded images exhibit a number of pixel misclassifications.

---

**Algorithm 1** Pixel Classification

---

**Input:** Image with target

**Output:** Thresholded H and S plane image and back-projection image

- 1: Convert Image to HSV
  - 2: Display original H and S plane images
  - 3: Calculate  $M$  from H and S image
  - 4: **while** Target not completely segmented **do**
  - 5:   Threshold H and S images to get  $H_{min}$ ,  $H_{max}$ ,  $S_{min}$ , and  $S_{max}$
  - 6:   **run** Algorithm 2
  - 7:   Calculate back-projection using  $M_t$
  - 8:   Display back-projection image
  - 9:   Display thresholded H and S plane images
  - 10: **end while**
- 

Using the threshold values selected by the user, the two-dimensional histogram  $M$  is modified so that the values of bins outside the threshold values are set to zero. The resulting histogram,  $M_t$  represents the target that the user has identified as a discrete probability distribution. This process is detailed in Algorithm 2.

The user is also presented with the back-projection image where the value of each pixel is the probability that the pixel belongs to the target. A sample back-projection image can be seen in Figure 4-2(a).

---

**Algorithm 2** Two-Dimensional Histogram Thresholding

---

**Input:** 2D histogram  $M$

**Output:** 2D target histogram  $M_t$

- 1: **for**  $h = 0$  to  $H_{bins}$  **do**
  - 2:   **for**  $s = 0$  to  $S_{bins}$  **do**
  - 3:     **if**  $h < H_{min}/H_{width}$  **or**  $h > H_{max}/H_{width}$  **or**  $s < S_{min}/S_{width}$  **or**  $s > S_{max}/S_{width}$  **then**
  - 4:        $bin_{H,S} = 0$
  - 5:     **end if**
  - 6:   **end for**
  - 7: **end for**
  - 8: **return**  $M_t$
- 

### 4.3 Image Segmentation and Object Description

Once the user has defined the threshold values acceptable for isolation of the target, the model histogram  $M_t$  is used to calculate back-projection images of future

frames. This process is shown in Algorithm 3. For each consecutive frame, the image is converted to HSV color space. The H and S value of each pixel is back projected through the histogram to form a whale probability image, for example Figure 4-2(a).

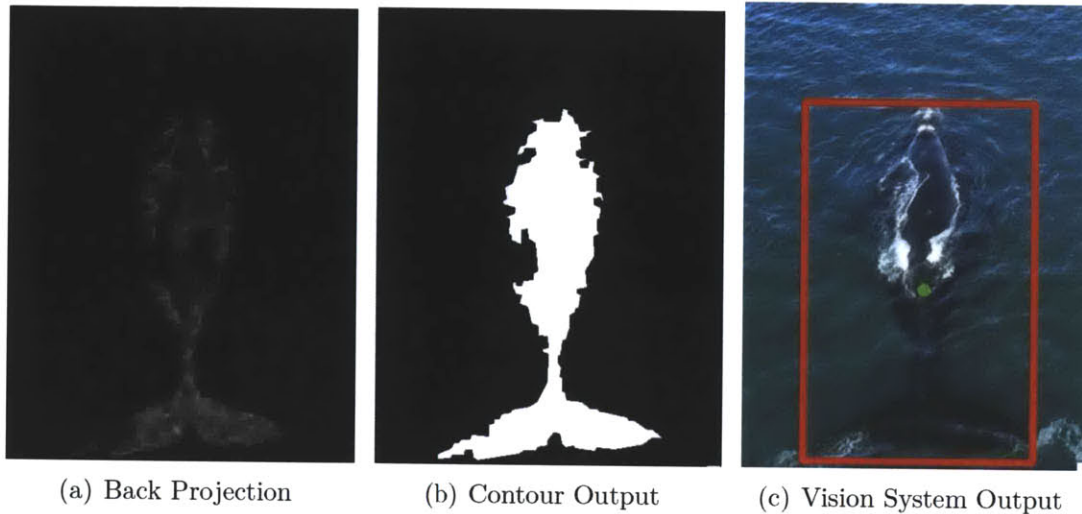


Figure 4-2: Target Recognition Through Contour Identification

Several heuristics are applied in order to improve the classification performance. Morphological heuristics are applied to reduce the low-level pixel misclassification while size-based heuristics are used at a high-level to eliminate entire regions of pixels.

First, a greyscale morphological opening operation with a  $3 \times 3$  kernel is applied to remove small false positive regions which are often noise in the back-projection image. Next, a greyscale morphological closing operation with a 3-by-3 square kernel is used to join together positive regions which are close in proximity to fill in small gaps.

After the back-projection image is filtered with morphological operations, the OpenCV function `cvFindContours` is used to represent contiguous groups of non-zero value pixels using Freeman chain code. This function is used due to its low overhead and simple structure which allows the groups of target pixels to be easily identified and manipulated. The function identifies the outer contours of target pixel groups.

Several descriptive features of the regions are used to further eliminate any misclassification. Not every method is required for successful object identification, but

these tools are available to the user. It is up to the user to determine which of the following heuristics are likely to improve the results of the vision system. These heuristics remove contours based on several size features. These features are presented to the user through sliding track bars. This allows the user to dynamically adjust the heuristic variables and observe the effect on the vision system's performance.

First, the perimeter of each contour is calculated and compared to a user defined minimum perimeter value. It is assumed that the whales are of the same relative size in the image so that contours with much smaller dimension sizes are considered noise. If the contour's perimeter is too small, the contour can be rejected.

It is also assumed that any occlusions are due to whales moving underneath another whale. In this case, the two objects would be combined into a single object. Therefore, the system can eliminate contours with dimensions smaller than a user defined percentage of the largest contour identified in the frame. These minimum dimensions are periodically updated to account for orientation changes of the largest contour due to camera manipulation or movement of the whale itself. This procedure is effective for instances when the target object is large in the frame which is one of our assumptions.

Finally, the user has the option to remove contours based on aspect ratios. Contours with extreme aspect ratios usually occur around the borders of the image and appear as noise. Assuming the that whale's natural shape is rectangular and relatively regular, it is possible to eliminate long and thin contours as noise.

If a contour passes the various heuristics, the `cvApproxPoly` function which uses the Douglas-Peucker algorithm [25] is applied to approximate the contour with a polygon having fewer vertices than the original contour. This operation helps eliminate noise at the edges of the contour so that the remaining polygon has smooth edges and a more distinct shape. Processing continues until there are no contours remaining or until a user defined maximum number of contours has been reached. A sample output of the completely processed contour image is shown in Figure 4-2(b).

The center of mass and an axis-aligned bounding box around the contour are plotted to identify the target to the user as shown in Figure 4-2(c). This process

continues on subsequent frames until the user ends the process, creates a new target histogram, or modifies the thresholds on a previous histogram.

---

**Algorithm 3** Image Segmentation and Object Description

---

**Input:** Image, target histogram  $M_t$

**Output:** Image with target(s) identified

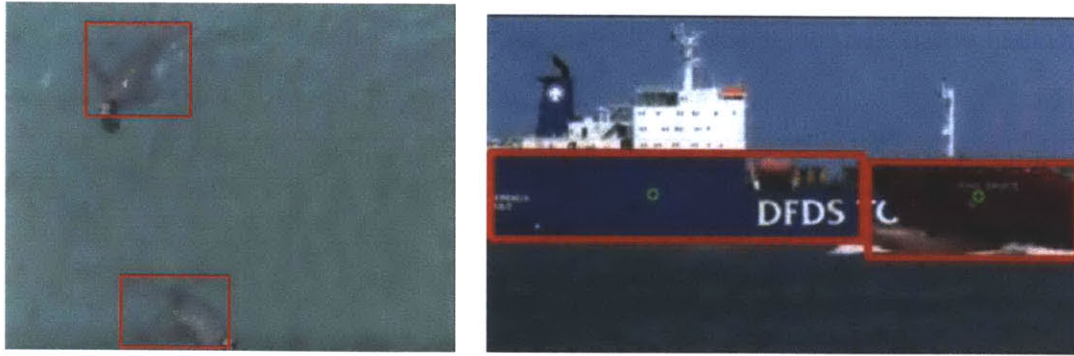
- 1: Convert image to HSV
  - 2: Calculate back-projection image using  $M_t$
  - 3: Perform morphological operations on back-projection
  - 4: Identify pixel groups as contours
  - 5: Identify largest contour height and width
  - 6: **if**  $Perimeter_{Contour} < Perimeter_{Max}$  **then**
  - 7:     Remove contour
  - 8: **end if**
  - 9: **if**  $Width_{Contour} < Width_{Max}$  **and**  $Height_{Contour} < Height_{Max}$  **then**
  - 10:     Remove contour
  - 11: **end if**
  - 12: Calculate center of mass
  - 13: Plot center and bounding box and display output image
- 

## 4.4 Tracking Multiple Objects

While this research focuses on developing a computer vision system to identify whales in an ocean environment, the pixel classification and image segmentation algorithms can be utilized for other applications. The main underlying assumption which makes the algorithms successful is the belief that the intended target will be described by roughly the same H and S values throughout the experiment and these values are different than the background's H and S values. There are some situations where the user may wish to track multiple objects at the same time. If the intended targets can be described by the same H and S characteristics, the vision system will identify them both simultaneously. As shown in Figure 4-3(a), the vision system is able to identify multiple seals using a single target histogram. This image is from footage recorded in Argentina.

However, by relying on one distinct target histogram, it is not possible to identify and track targets with significantly different H and S ranges. To overcome this obstacle, the vision system supports the creation of multiple target histograms. If a new object comes into view, the user has the ability to capture a frame and create a





(a) Tracking Multiple Targets of Similar H and S Properties      (b) Tracking Multiple Targets of Different H and S Properties

Figure 4-3: Simultaneous Tracking of Multiple Targets

separate target histogram for the new object using the process in Algorithm 1. Each time the user creates a new target histogram, the target histogram and corresponding threshold values are stored in a vector including the previous target histograms and their threshold values. Algorithm 3 is then executed for each target histogram separately. Each target is identified with a bounding box and center of mass marker. An example is shown in Figure 4-3(b) where one target histogram was created to track the blue ship while another was created to track the red ship.

## 4.5 Position Vector Calculation

Once the appropriate objects have been identified in the image, a vector that points to the average location of the objects is calculated and optionally displayed. For images with only one target, the vector points directly to the center of mass of the target. However, when there are multiple targets in the frame, this vector is computed in two separate ways. The first method takes each target's  $x$  and  $y$  center of mass coordinates and computes the average center of mass location of all the targets. This is shown in Equation 4.1 where  $(x_e, y_e)$  is a position vector representing the average location of the identified objects,  $x_i$  and  $y_i$  represent the center of mass coordinates for each object, and  $n$  is the total number of objects found in the image.

$$(x_e, y_e) = \left( \frac{1}{n} \sum_{i=1}^n x_i, \frac{1}{n} \sum_{i=1}^n y_i \right) \quad (4.1)$$



The second method computes a weighted average location based on the bounding box area of each target. Objects that appear larger in the frame are emphasized in this calculation. This calculation is shown in Equation 4.2 where each object's  $x$  and  $y$  center of mass coordinates are multiplied by the pixel area  $a_i$  of the bounding box enclosing the object with the constraints  $\sum_{i=1}^n a_i = 1$  and  $a_i > 0$ .

$$(x_e, y_e) = \left( \sum_{i=1}^n a_i x_i, \sum_{i=1}^n a_i y_i \right) \quad (4.2)$$

In Figure 4-4, the yellow arrow is the unweighted position vector while the black arrow is the weighted position vector. From this figure it is evident that the large size of the bottom group of whales has pulled the weighted position vector down compared to the unweighted position vector. If, for example, the user wanted to stay over the largest group of whales, the weighted position vector should be used. For the hardware experiments in this research, there was only one target so an unweighted position vector was used.



Figure 4-4: Weighted vs. Unweighted Position Vector

When tracking multiple objects using multiple target histograms, the position vector is computed slightly differently. When tracking multiple targets, a separate position vector is returned for each target histogram. For example, in Figure 4-3(b), an position vector would be calculated to point towards the blue ship's center of

mass and another position vector would be computed for the red ship's center of mass. In order to compute a final position vector, a cumulative, unweighted average is calculated. This is shown in Equation 4.3 where  $(x_f, y_f)$  is the final position vector,  $(x_e, y_e)$  is the most recent position vector calculated by Algorithm 3 for a new target histogram  $\bar{M}_t$ , and  $i = 1, 2, \dots, N$  where  $N$  is the total number of target histograms computed.

$$(x_f, y_f)_i = \left( \frac{(x_e, y_e)_i + (i - 1) \cdot (x_f, y_f)_{i-1}}{i} \right) \quad (4.3)$$

The position vector is ultimately used to provide high-level trajectory planning for the vehicle, so different applications could require different vector calculations. Enabling the user to calculate this vector in a variety of ways makes the complete vision system more adaptable to different situations. Nature is very unpredictable and we can not create algorithms appropriate for every situation, but we hope the tools integrated into this vision system make it useful for multiple applications.

# Chapter 5

## Controller Design

This chapter describes the development of the Linear Quadratic Regulator (LQR) controllers developed to control the robot position vector  $[{}^b x, {}^b y, {}^b z, \psi]^T$ . Figure 3-4 shows the overall control strategy to maneuver the quadrotor to a desired location determined by visual position estimates. This chapter begins with a derivation of the LQR gain matrix calculation to motivate the state space representation of the quadrotor's dynamics presented in Section 3.3. The LQR gain matrices are computed and shown to work in simulation. These gains are then implemented in hardware and the tuning process is explained. The resulting controllers are able to position the quadrotor with less than two centimeters of error utilizing motion capture state feedback.

### 5.1 LQR Derivation

The low-level position controller is developed as a set of LQR controllers. The LQR controller is a full state feedback controller that is provably optimal. Several assumptions are required to derive the optimality proof of the LQR controller. Since the LQR controller requires access to all states, the system must be fully observable. It must also be fully controllable for the inputs to impact every state. The plant is represented in the Linear Time Invariant (LTI) state space form

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

and the desired controller is of the form

$$\mathbf{u} = -\mathbf{K}(\mathbf{x} - \mathbf{x}_d)$$

where  $\mathbf{x}_d$  is the vector of desired states.

First, the Hamilton-Jacobi-Bellman (HJB) equation is derived which will be used as a sufficiency condition for optimality. Given an instantaneous cost function  $g(\mathbf{x}, \mathbf{u})$  and a cost-to-go function  $J = \int_0^\infty g(\mathbf{x}, \mathbf{u})$ , the optimal cost-to-go is the cost of the optimal trajectory remaining after time  $t$ . This is written as

$$J^*(\mathbf{x}, t) = \min_{\mathbf{u}(t)} \int_t^\infty g(\mathbf{x}, \mathbf{u})$$

Taking the limit as the time between control updates,  $dt$ , approaches zero results in:

$$\begin{aligned} J^*(\mathbf{x}, t) &= \lim_{dt \rightarrow 0} \min_u [g(\mathbf{x}, \mathbf{u})dt + J(\mathbf{x}(t + dt), t + dt)] \\ &\approx \lim_{dt \rightarrow 0} \min_u \left[ g(\mathbf{x}, \mathbf{u})dt + J^*(\mathbf{x}, t) + \frac{\partial J^*}{\partial \mathbf{x}} \dot{\mathbf{x}}dt + \frac{\partial J^*}{\partial t} dt \right] \end{aligned}$$

Assume that  $J^*$  has no dependence on  $t$  and is a smooth function. The resulting infinite-horizon form of the HJB is shown in Equation 5.1

$$0 = \min_u \left[ g(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) \right] \quad (5.1)$$

If a policy  $\pi(\mathbf{x}, t)$  and a cost-to-go function  $J^\pi(\mathbf{x}, t)$  are found and the policy  $\pi$  minimizes the right hand side of the HJB for all  $\mathbf{x}$ , then the policy and cost-to-go function are optimal and denoted by the  $*$  superscript as shown below.

$$J^\pi(\mathbf{x}, t) = J^*(\mathbf{x}, t), \quad \pi(\mathbf{x}, t) = \pi^*(\mathbf{x}, t)$$

The infinite-horizon LQR cost function is of the unique form  $g(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T \mathbf{Q}\mathbf{x} + \mathbf{u}^T \mathbf{R}\mathbf{u}$  where  $\mathbf{Q}$  is a symmetric positive semi-definite matrix ( $\mathbf{Q} = \mathbf{Q}^T \geq 0$ ) and  $\mathbf{R}$  is a symmetric positive definite matrix ( $\mathbf{R} = \mathbf{R}^T > 0$ ). The  $\mathbf{Q}$  and  $\mathbf{R}$  matrices allow

the control system designer to penalize state errors and control input magnitudes to modify the overall system behavior.

Since  $J^*$  cannot depend on time, we guess the function  $J^*(\mathbf{x}) = \mathbf{x}^T \mathbf{S} \mathbf{x}$  where  $\mathbf{S}$  is a symmetric positive definite matrix ( $\mathbf{S} = \mathbf{S}^T > 0$ ). Substituting in  $\frac{\partial J^*}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{S}$ , the resulting HJB equation is reduced to Equation 5.2.

$$0 = \min_{\mathbf{u}} [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + 2\mathbf{x}^T \mathbf{S} (\mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u})] \quad (5.2)$$

To find the minimum  $\mathbf{u}$ , the gradient is set to zero due to the positive definite quadratic form of  $\mathbf{u}$

$$\frac{\partial}{\partial \mathbf{u}} = 0 = 2\mathbf{u}^T \mathbf{R} + 2\mathbf{x}^T \mathbf{S} \mathbf{B}$$

This reduces to the optimal policy  $\mathbf{u}^*$

$$\mathbf{u}^* = \pi^*(\mathbf{x}) = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} \mathbf{x} = -\mathbf{K} \mathbf{x}$$

Substituting the gain matrix  $\mathbf{K}$  into Equation 5.2 results in

$$0 = \mathbf{x}^T [\mathbf{Q} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + 2\mathbf{S} \mathbf{A}] \mathbf{x}$$

This further reduces to the final algebraic Ricatti equation, Equation 5.3.

$$0 = \mathbf{S} \mathbf{A} + \mathbf{A}^T \mathbf{S} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{Q} \quad (5.3)$$

Equation 5.3 is used to solve for the matrix  $\mathbf{S}$  which is used to calculate the optimal feedback gain  $\mathbf{K}$ . This can be done in MATLAB using the function  $[\mathbf{K}, \mathbf{S}] = \text{lqr}(\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R})$ .

In calculating the gain matrix  $\mathbf{K}$ , the resultant value moves in the direction of steepest descent of the cost-to-go function represented by the  $-\mathbf{S} \mathbf{x}$  term. However, the controller is limited by the control inputs  $\mathbf{u}$  so the term  $-\mathbf{B} \mathbf{S} \mathbf{x}$  is the projection of the steepest descent onto the feasible control space. The  $\mathbf{R}^{-1}$  term manipulates the

descent relative to the user defined weightings on the control inputs.

## 5.2 LQR Implementation

Using the Simulink model verified in Section 3.3.1, the adapted dynamics in Equations 3.12, 3.13, and 3.14 are used to create a LTI state space model of the system. The LQR controller requires a linear model to estimate the optimal gains. The linearization process removes the offsets from the system dynamics. These offsets are later accounted for in the controller design after the gains are calculated. Note that a rotation matrix is used to rotate the desired state in global frame to a desired state in body frame which results in a body frame error vector.

For the state space model of the quadrotor system, nine states are used. These states are  $[{}^b x, {}^b y, {}^b x, \phi, \theta, \psi, {}^b \dot{x}, {}^b \dot{y}, {}^b \dot{z}]^T$ . The entire nine state state space model of the system is uncontrollable. Instead, each element of the position matrix  $[{}^b x, {}^b y, {}^b z, \psi]^T$  is broken into a smaller state space model and individual LQR controllers are designed to control each of these states.

In practice, the LQR controller resulted in steady state error from the desired position. This was removed using an integral term on the position and heading states. The LQR gains were then empirically hand tuned in an iterative process. Initially, the motion capture measurements were used for state feedback. The quadrotor was commanded to hover around the desired translational position of (0, 0, 1.5) m and a zero degree heading. Without integral control or offsets, the LQR gains were modified to command a behavior which could quickly respond to position errors and disturbances with minimal overshoot. Once this behavior was satisfactory, the offsets were adjusted to reduce any large steady state error. Finally, the integrator gains were tuned to remove any additional steady state error without exciting any oscillatory modes of the system. The final LQR gains were within one percent of the ones computed below. The LQR gains were modified to handle the slightly more noisy and less accurate measurements outputted from the Extended Kalman Filter (EKF). The same iterative tuning procedure was done and the resulting gains were of smaller magnitude reflecting the lower accuracy of the estimated state when compared with the

state computed by the motion capture system. The specific values of the individual  $\mathbf{K}$ ,  $\mathbf{Q}$ , and  $\mathbf{R}$  matrices are described below.

The  ${}^b x$  position is modeled using the state space equations below.

$$\begin{bmatrix} {}^b \dot{x} \\ {}^b \ddot{x} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -K_{\dot{x}} & K_p \\ 0 & 0 & -\frac{1}{a_{\theta}} \end{bmatrix} \begin{bmatrix} {}^b x \\ {}^b \dot{x} \\ \theta \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{K_{\theta}}{a_{\theta}} & 0 & 0 \end{bmatrix} \begin{bmatrix} U_{\phi_d} \\ U_{\theta_d} \\ U_{\dot{\psi}_d} \\ U_z \end{bmatrix} \quad (5.4)$$

The cost matrices for the LQR calculations are

$$\mathbf{Q}_{\mathbf{x}} = \begin{bmatrix} 350000 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 250000 \end{bmatrix} \quad \mathbf{R}_{\mathbf{x}} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (5.5)$$

These matrices produce the following  $\mathbf{K}_{\mathbf{x}}$  matrix of gains.

$$\mathbf{K}_{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 \\ 1870 & 1420 & 370 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.6)$$

The  ${}^b y$  position is modeled using the state space equations below.

$$\begin{bmatrix} {}^b \dot{y} \\ {}^b \ddot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -K_{\dot{y}} & K_r \\ 0 & 0 & -\frac{1}{a_{\phi}} \end{bmatrix} \begin{bmatrix} {}^b y \\ {}^b \dot{y} \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{K_{\phi}}{a_{\phi}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_{\phi_d} \\ U_{\theta_d} \\ U_{\dot{\psi}_d} \\ U_z \end{bmatrix} \quad (5.7)$$

The cost matrices for the LQR calculations are

$$\mathbf{Q}_y = \begin{bmatrix} 375000 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 250000 \end{bmatrix} \quad \mathbf{R}_y = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (5.8)$$

These matrices produce the following  $\mathbf{K}_y$  matrix of gains.

$$\mathbf{K}_y = \begin{bmatrix} 1940 & 1490 & 360 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.9)$$

The  $\psi$  heading is modeled using the state space equations below.

$$\begin{bmatrix} \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} \psi \end{bmatrix} + \begin{bmatrix} 0 & 0 & K_{\dot{\psi}} & 0 \end{bmatrix} \begin{bmatrix} U_{\phi_d} \\ U_{\theta_d} \\ U_{\dot{\psi}_d} \\ U_z \end{bmatrix} \quad (5.10)$$

The cost matrices for the LQR calculations are

$$\mathbf{Q}_\psi = \begin{bmatrix} 2500000 \end{bmatrix} \quad \mathbf{R}_\psi = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (5.11)$$



These matrices produce the following  $\mathbf{K}_\psi$  matrix of gains.

$$\mathbf{K}_\psi = \begin{bmatrix} 0 \\ 0 \\ 5000 \\ 0 \end{bmatrix} \quad (5.12)$$

The  $z$  position is modeled using the state space equations below.

$$\begin{bmatrix} b\dot{z} \\ b\ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -K_z \end{bmatrix} \begin{bmatrix} b_z \\ b\dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_z \end{bmatrix} \begin{bmatrix} U_{\phi_d} \\ U_{\theta_d} \\ U_{\psi_d} \\ U_z \end{bmatrix} \quad (5.13)$$

The cost matrices for the LQR calculations are

$$\mathbf{Q}_z = \begin{bmatrix} 25000 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{R}_z = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (5.14)$$

These matrices produce the following  $\mathbf{K}_z$  matrix of gains.

$$\mathbf{K}_z = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1580 & 990 \end{bmatrix} \quad (5.15)$$

As mentioned previously, offsets are computed to account for the affine nature of the learned model. After the input gains are calculated, the linear offsets can be computed and converted to offsets for the control inputs. This accounts for any offsets due to the additional payload not aligning with the natural  $z$  axis of the quadrotor. The thrust control offset is estimated to be the control value which resulted in a stable

hover without altitude changes.

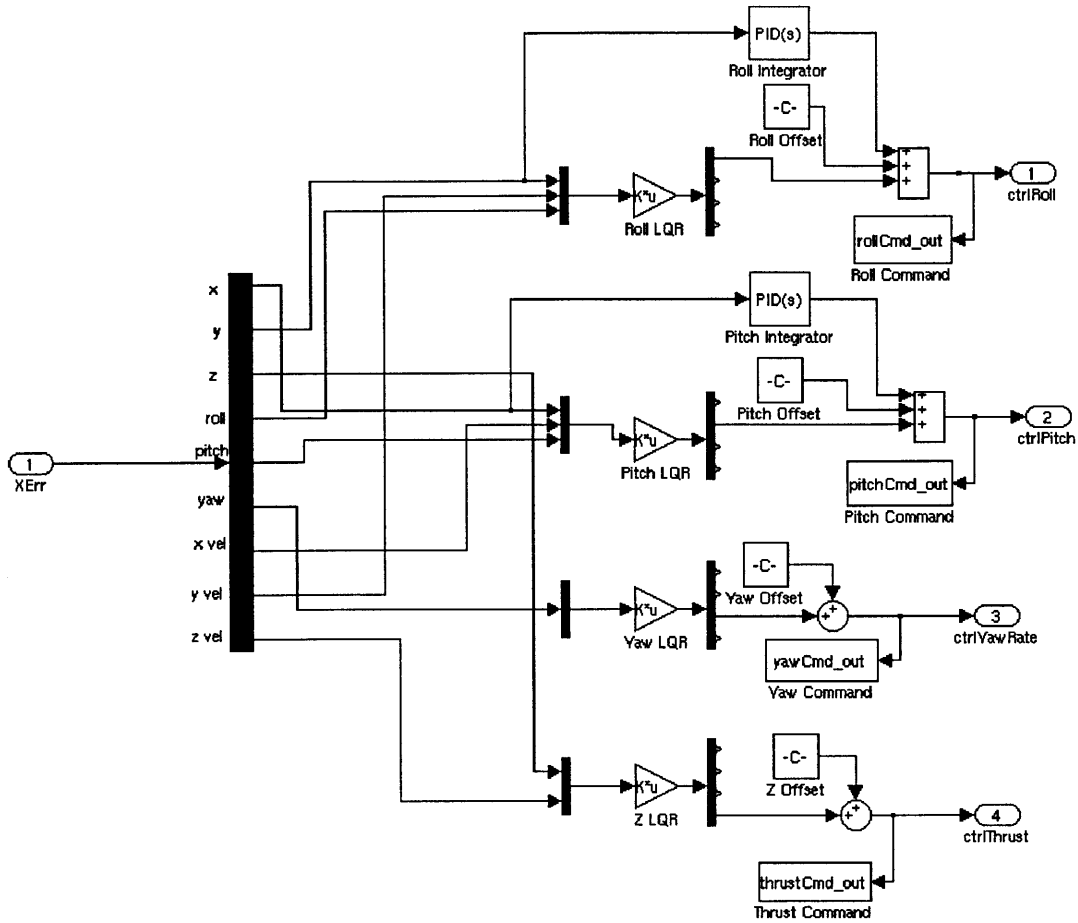


Figure 5-1: Simulink LQR Controller Block Diagram

This LQR controller is summarized visually in Figure 5-1 and the entire feedback loop is visible in Figure 5-2. In order to verify the controller, a typical trial was simulated using the Simulink model. The desired trajectory was taken from a physical run of the system and was created by the vision system output. Using the LQR gains described above, the simulated performance of the quadrotor is shown in Figure 5-3. The controller causes the quadrotor to converge on the desired trajectory and maintain close proximity to the desired trajectory throughout the path.

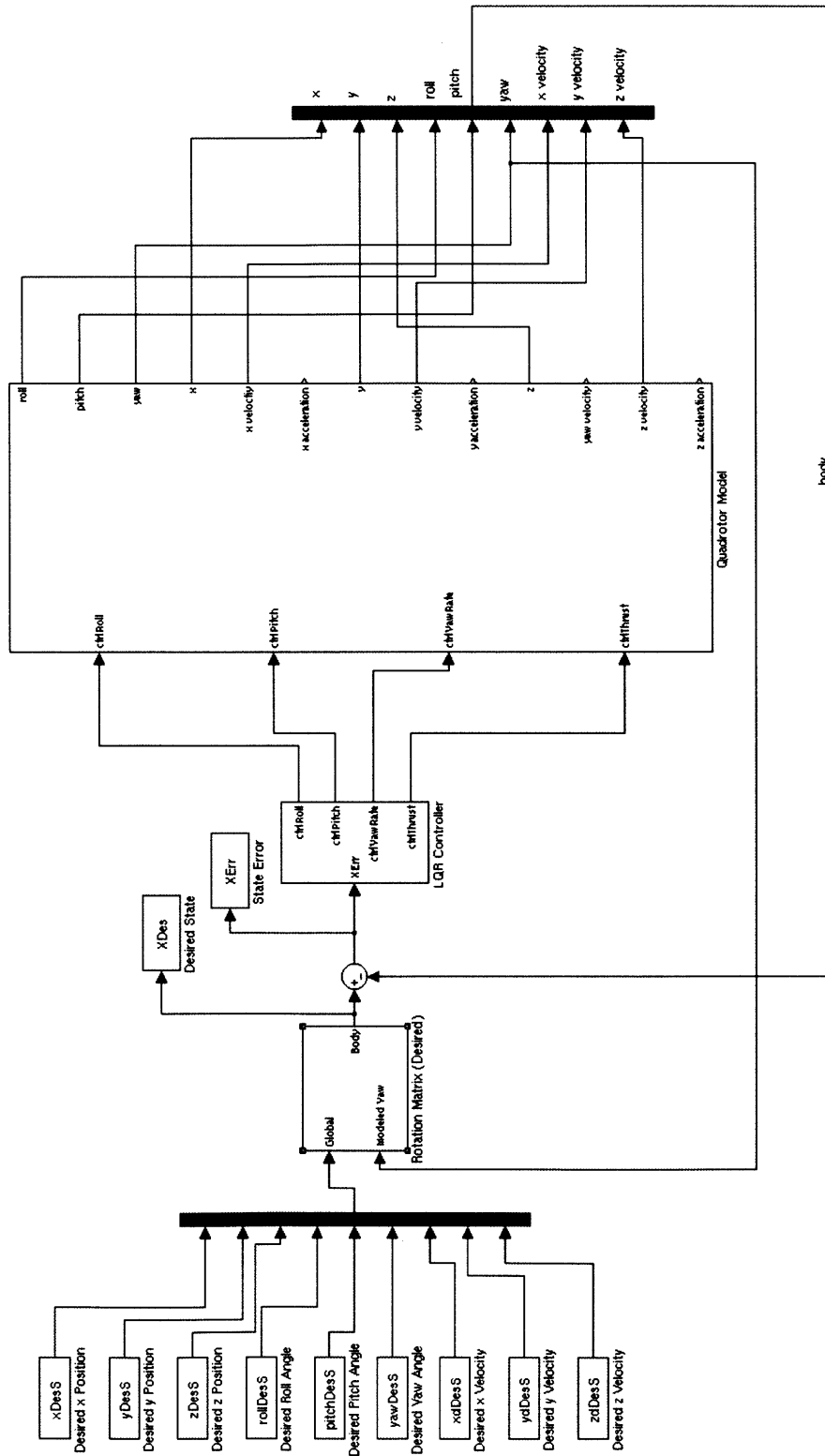


Figure 5-2: Simulink Feedback Block Diagram

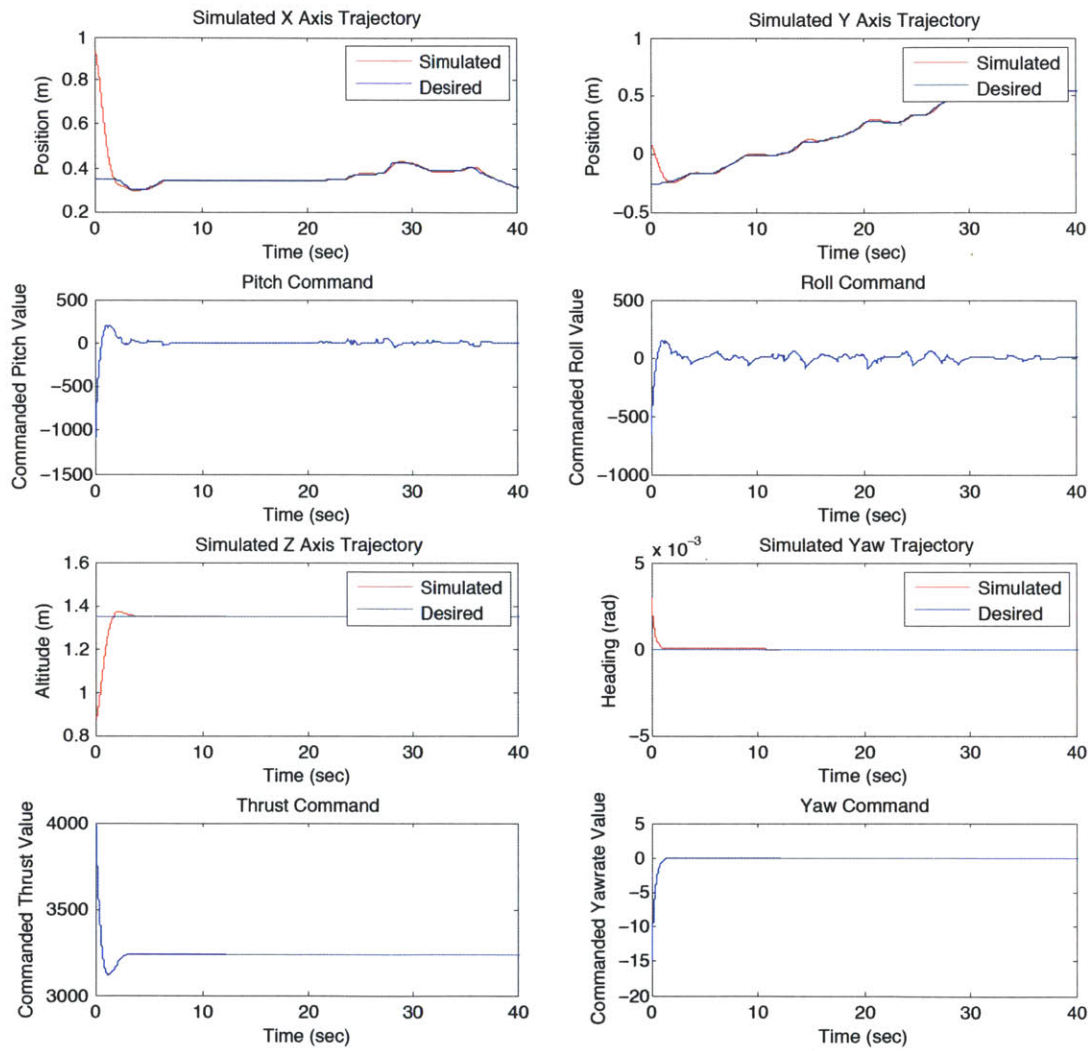


Figure 5-3: Simulated LQR Trajectory Tracking

# Chapter 6

## Experimental Results

In order to evaluate the real world performance of the vision system, the algorithms from Chapter 4 were tested on video footage of whales taken from field experiments and the internet. Of primary importance is the footage taken from the field tests in Argentina. The goal of this thesis is to provide a platform capable of autonomously tracking these whales utilizing only on-board computation. The algorithm had a 98.99% recall for this footage and the algorithm. Therefore, the algorithm would be effective at identifying whales in this environment.

In order to evaluate the control systems developed in Chapter 5 for visual servoing, the quadrotor was tasked with autonomously tracking an iCreate platform. An indoor motion capture system was used to calculate an accurate global state of the robot similar to Global Positioning Satellite (GPS) since GPS reception is available outdoors. However, we also desire the ability to continue to track the targets in environments without reliable GPS signals or even in GPS denied environments. The Extended Kalman Filter (EKF) described in Section 3.4 is used to fuse Inertial Measurement Unit (IMU) measurements with the vision system's position estimates to estimate the vehicle state independent of any external localization source. Visual servoing results are presented which show the accuracy of the quadrotor hovering over a static target with and without motion capture state feedback. We then program the iCreate to move along a varied trajectory and present results of the quadrotor tracking the moving target with and without external localization.

## 6.1 Experimental Set-up, Methodology, and Metrics for Object Identification

The primary data set for evaluating the vision system is the video footage we recorded using a robotic helicopter. From this data set, six clips are extracted resulting in 3,537 frames. All but one of these clips was recorded at an angle directly overhead of the whales. The ability of the camera on-board the robot to pan and tilt allowed for one clip to be recorded at an angled perspective. Video footage found on the web is used to increase the diversity of viewpoints, sea conditions and species. These fifteen clips comprising 9,543 frames include clips of Blue, Grey, and Humpback whales recorded at both overhead, angled, and surface perspectives. In both sets of data, some clips contain multiple identifiable whales. Other clips contain whales that appear from outside the frame or are occluded by another whale. The clips also vary in duration as well as environmental factors. A summary of the properties of the whale clips analyzed is shown in Table 6.1.

The algorithms are processed on a desktop computer for this series of experiments. For each clip, an output video is recorded which depicts every contour identified as a target with a bounding box. After each run, the output video is analyzed frame by frame to compute the performance of the algorithms. All thresholds and heuristic values remain constant for each video clip. The threshold and heuristic constants are manually tuned for each clip in order to improve performance.

The following metrics are used to evaluate the vision system's performance:

- Rate of false positives.
- Rate of false negatives.
- Rate of contour splitting.

Contour splitting occurs when one object is represented by multiple contours. While the threshold values may prove effective at the beginning of the clip, lighting, white balance and angle changes can cause these numbers to be less representative of the

Table 6.1: Properties of Video Clips Analyzed

Clip Name	Frames	View Angle	Species	# Whales
Web1	281	90°	Blue	1
Web2	393	90°	Blue	1
Web3	358	90°	Blue	2
Web4	1117	45°	Blue	2
Web5	1047	45°	Blue	2
Web6	343	45°	Blue	2
Web7	233	90°	Blue	1
Web8	1209	0°	Grey	2
Web9	1268	0°	Humpback	2
Web10	625	0°	Humpback	1
Web11	514	0°	Humpback	1
Web12	364	0°	Humpback	2
Web13	546	0°	Humpback	1
Web14	584	0°	Humpback	1
Web15	661	0°	Humpback	1
Argentina1	520	45°	Southern Right	3
Argentina2	259	90°	Southern Right	5
Argentina3	657	90°	Southern Right	2
Argentina4	97	90°	Southern Right	6
Argentina5	1343	90°	Southern Right	4
Argentina6	661	90°	Southern Right	2

target as time progresses. Therefore, the target is sometimes represented as separate objects. This does not result in a false positive since the correct target is still identified, but users of the vision system can use the amount of splitting as a soft measure of hue (H) and saturation (S) variation throughout the duration of the clip. False positives and false negatives are counted in each frame for each object identified.

## **6.2 Results of Object Identification Algorithms**

This section details the specific results of the vision system's performance on pre-recorded video footage. The clips are separated by camera view angle and aggregate performance statistics are presented. Finally, we present results tracking boats to show that the vision system is able to adapt to any target with homogenous HS characteristics which vary from the background.

### **6.2.1 Tracking with an Overhead Perspective**

Nine clips comprising 4,282 frames were analyzed with a mixture of quadrotor footage from Argentina as well as helicopter based footage from the web. Using this view angle resulted in a large distance between the camera and the water's surface as shown in the sample frame in Figure 6-1. This distance reduced significant lighting changes in the image which resulted in more consistent H and S values for both the whales and the background. The only noticeable adverse effect of this viewpoint was caused by the wakes of the whales. When the whales were on the water's surface, their wake's H and S characteristics were very similar to the H and S characteristics of the whales. The performance statistics for all clips with an overhead view are shown in Table 6.2.

In particular, one clip caused a high number of false positives. In this clip, the whales were near the beach and the waves on the beach could be seen as well. The false positives occurred mainly along the beach line where the waves broke along the beach. The size difference between a large group of whales and a single whale in the image made size-based filtering ineffective. However, other logic could be used to improve the performance in this clip and others similar to it. Whales on the surface do cause waves, but these waves account for only a small number of pixels in the





Figure 6-1: Sample Frame From Footage Recorded at an Overhead Perspective

bounding box that surrounds the whale. By calculating the estimated percentage of *wave* pixels in each bounding box, false positives can be rejected. A similar method of calculating the number of *ocean* pixels and *beach* pixels was used. Using these additional logic arguments on the problematic clips, the vision system's performance was vastly improved to the statistics shown in the bottom half of Table 6.2.

Table 6.2: Vision System Results: Overhead Perspective

	Minimum	Average	Maximum	Stand. Dev.
False Negative	0.00%	1.69%	14.43%	4.79%
False Positive	0.00%	26.85%	193.81%	63.93%
Splitting	0.00%	2.45%	13.85%	4.58%
Using Pixel Value Logic				
False Negative	0.00%	0.21%	1.03%	0.43%
False Positive	0.00%	1.54%	6.01%	2.56%
Splitting	0.00%	2.36%	9.90 %	3.32%

### 6.2.2 Tracking at an Angled Perspective

The four clips of 3,027 frames in this section were also recorded from above the whale, but from an oblique angle that caused the camera to be closer to the water's surface as shown in Figure 6-2. As expected, the percentages shown in Table 6.3 were slightly higher than the aerial perspective statistics when the outliers are removed. Specifically, the rate of false positives is higher. The closer proximity of the camera



Figure 6-2: Sample Frame From Footage Recorded at an Angled Perspective

Table 6.3: Vision System Results: Angled Perspective

	Minimum	Average	Maximum	Stand. Dev.
False Negative	0.00%	2.43%	8.46%	4.04%
False Positive	1.92%	21.09%	68.5%	31.75%
Splitting	0.00%	1.46%	5.38%	2.63%
Without Max Trial				
False Negative	0.00%	3.15%	8.46%	4.63%
False Positive	1.92%	5.28%	9.36%	3.77%
Splitting	0.00%	1.94%	5.38%	2.99%

to the whales caused lighting effects to become more pronounced resulting in higher false positive rates.

As evident in the statistics, the vision system performed poorly for one clip in particular. This was due to one of the whales having an extended wake that was consistently identified as a false positive. Unfortunately, the size of the wake was very similar to the size of the whale reducing the effectiveness of size-based filtering. Also, the H and S characteristics of the wake were very similar to the H and S characteristics of the whales. Trying to identify the wake by the amount of *wave* pixels was also ineffective due to the combination of similar size and H and S properties of the wake and the target whales. This clip shows a drawback with the algorithm and potential for further refinement.

### 6.2.3 Tracking from the Surface

The surface perspective proved to be the most difficult viewpoint for this vision system. This is due to extreme lighting changes and changing of camera orientation. Eight clips from the internet comprising 5,771 frames were used for evaluation. The results are shown in Table 6.4. The majority of these clips appeared to be recorded from the side of a boat in very close proximity to the whales as seen in Figure 6-3.



Figure 6-3: Sample Frame From Footage Recorded at a Surface Perspective

The whales would often surface and dive, making their H and S values inconsistent and resulting in false negatives. They would also change orientation in the water, sometimes showing a white underbelly, which further hindered the success of the vision system. Lastly, sunlight reflecting off the water caused an increase in the rate of false positives. Decreasing the proximity between the camera and the target exaggerated the effect of environmental factors. Viewed from high above, these lighting changes were not noticeable and did not cause the same amount of disturbance compared to clips recorded from surface level.

Table 6.4: Vision System Results: Surface Perspective

	Minimum	Average	Maximum	Stand. Dev.
False Negative	0.00%	10.8%	45.06%	18.15%
False Positive	0.00%	13.8%	45.50%	17.51%
Splitting	0.00%	6.11%	24.73%	9.15%

#### 6.2.4 Performance Comparison of View Angle

As evident by the statistics shown in Table 6.2, Table 6.3, and Table 6.4, the vision system’s performance was slightly degraded for clips recorded at an angled perspective and even more deteriorated for clips recorded at a surface perspective when compared to those recorded at an overhead perspective. While both overhead and angled views had specific issues such as beach breaks and surface wakes that required additional heuristics to rectify, the vision system was more accurate with clips taken from an overhead perspective. As mentioned previously, the overhead perspective typically created a larger distance between the whales and the camera compared to an angled perspective. This resulted in a more distinct target H and S histogram that was less prone to false positives. Using the vision system on-board a flying robot should result in video footage similar to those taken with an overhead perspective and the vision system has shown strong performance for this case.

The vision system performed relatively poorly on the surface level perspective whale clips due to the similar H and S characteristics of the target and the background as shown in Table 6.4. Clips that had average background S values within the S threshold of the target had significantly higher false positive and contour splitting rates. This occurred in three of the eight clips recorded from a surface perspective. These clips also had average background H values within five values of the target H threshold range. When the background H values were near the target H value range but the average background S values were not near the target S threshold range there was not a significant decrease in performance of the vision system. The H plane was not affected by lighting changes which may explain this result. Lighting changes are accounted for in the S plane, which is more noticeable at close proximity to the target.



These effects, combined with background H values similar to that of the target cause the vision system's performance to deteriorate at the surface perspective.

### 6.2.5 Boat Identification

The object identification and tracking algorithms can be used to identify other types of objects at sea, for example boats. Clips were taken from the internet to gather a variety of boats in various environments. A sample frame from one of these clips is shown in Figure 6-4. Seven clips were analyzed with five clips recorded from a surface perspective, one recorded from an angled perspective, and one recorded from an overhead perspective. Specific details about each clip are shown in Table 6.5. The results of the algorithm's performance are summarized in Table 6.6. The vision system was very successful at tracking boats in these clips primarily because of the H and S difference between the boats and the background. A main cause of false positives and splitting in the whale clips was the similar H and S characteristics of the whales and the ocean. The boats were painted distinct colors that contrasted well from the background. In one clip, only the rear of the boat was visible, and the rear contained a stripe. This stripe caused the rear to be identified as two separate objects which caused a large amount of splitting to occur for that specific clip.

Table 6.5: Properties of Video Clips Analyzed

Clip Name	Frames	View Angle	Boat Type	# Boats
Boat1	989	0°	Tug	2
Boat2	823	0°	Tug	2
Boat3	795	0°	Tug	2
Boat4	860	0°	Cruise	1
Boat5	1261	0°	Cargo	2
Boat6	551	90°	Cargo	1
Boat7	544	45°	Cargo	1

The boat clips also displayed the ability of the vision system to track multiple targets with differing H and S characteristics. To demonstrate this property, a clip with one red boat and one blue boat was used. A sample frame from this clip was shown previously in Figure 4-3(b). This clip resulted in no false positives or false negatives and only 0.24% instances of splitting. Without foreknowledge of the H

and S characteristics of the multiple targets, the user must create the new target histogram in real-time. This may cause the new target to be unidentified for a brief period of time while the thresholding process is completed.



Figure 6-4: Sample Frame From Boat Footage

Table 6.6: Vision System Results: Boat Identification

	Minimum	Average	Maximum	Stand. Dev.
False Negative	0.00%	0.30%	1.09%	0.51%
False Positive	0.00%	0.00%	0.00%	0.00%
Splitting	0.00%	4.02%	19.44%	7.49 %

### 6.3 Experimental Set-up, Methodology, and Metrics for Visual Servoing

To evaluate the performance of the LQR controllers, the vision system was used to identify the target and created a desired trajectory to position the quadrotor above the target. The iCreate target was static for these tests. These experiments evaluated the performance of the vision system processed on-board a computationally impoverished platform with a low-quality camera. This contradicted the previous evaluation of the object tracking algorithms which used a desktop computer with clips taken from high-quality cameras. We demonstrate the ability to run the control and vision systems simultaneously in order to hover the robot stably.

The first control strategy utilized the motion capture system to sense the quadrotor's pose and the vision system's position estimates to determine the translational error to the target. This method was intended to simulate target tracking in an outdoor environment. Since GPS is available outdoors to give an accurate estimate of the position of the robot, the motion capture system was used to emulate this sensor. Butterworth filters were used on the translational positions to compute smooth velocity values for the controller. This approach was also used to define a baseline performance for the tracking system.

In the first experiment, the quadrotor control module received a global pose estimate from the system as well as the attitude compensated estimate of the target's position from the vision system. The vision system's position estimates were used to create a desired trajectory from the latest quadrotor position to the estimated target location. Once a new vision estimate was received, the process was repeated and the desired trajectory extended.

A linear interpolation function was used to compute successive waypoints along the desired trajectory to send to the LQR controllers. This function adjusted the distance between successive waypoints as a function of the magnitude of the estimated error to the target. When the error was small, the distance between waypoints was also small. This improved the accuracy of the tracking by reducing high frequency noise in the vision system output when the quadrotor was directly over the target. However, when the estimated error was large, a greater distance between waypoints was used. This caused the quadrotor to move faster and therefore reduce the error faster.

The second control strategy removed external localization and relied entirely on visual position estimates and IMU measurements. This control strategy utilized an EKF to estimate the pose of the quadrotor to provide state feedback. Since there was no external localization source, the target centroid was the origin of the global coordinate frame and the desired quadrotor position was this origin. As the target moved, the coordinate frame origin also moved and the quadrotor attempted to stabilize around this dynamic origin. This estimated pose was sent to the control module which computed commands to maneuver the quadrotor to the center of the target.

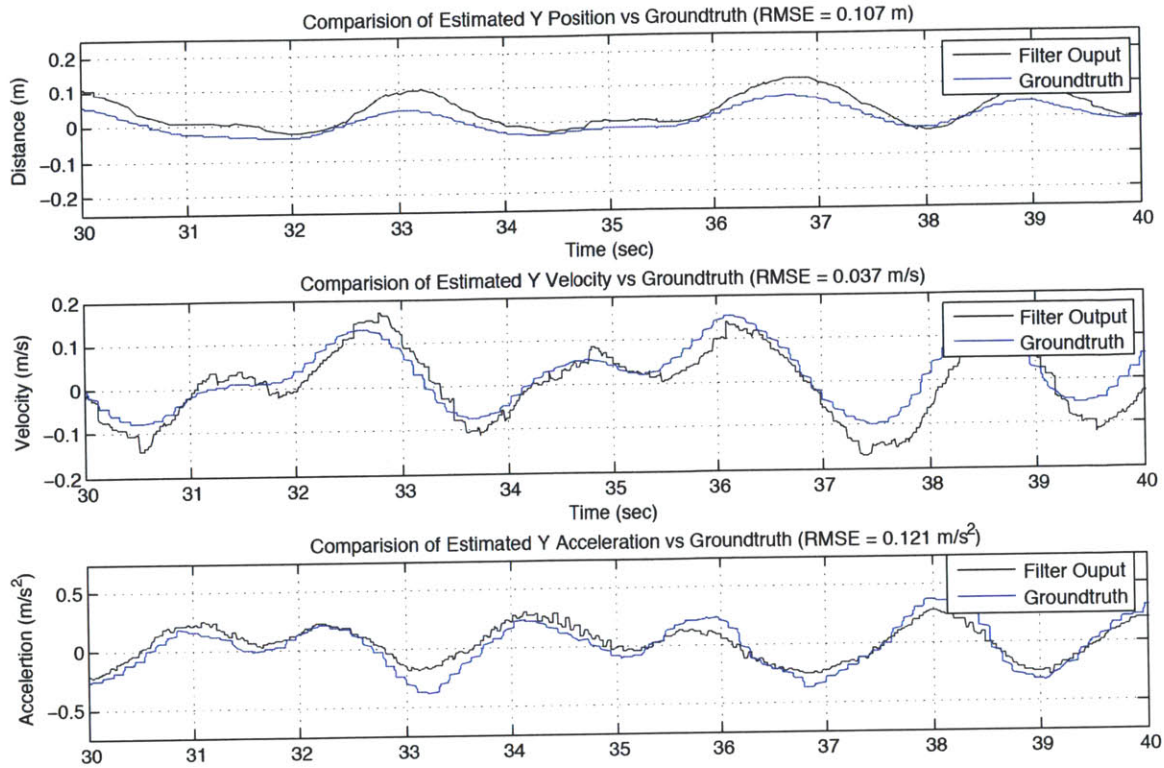


Figure 6-5: Comparison of EKF Output to Ground Truth

An example plot of the estimated values compared to the ground truth is shown below in Figure 6-5. For this sample of data, the position estimate had a Root Mean Square (RMS) error of 0.107 m, a velocity RMS error of 0.037 m/s, and an acceleration RMS error of 0.121 m/s<sup>2</sup>. This proved to be a good state estimate. The velocity signal is the critical estimate for stable control. The EKF velocity estimate was fairly smooth with a minimal delay. High LQR gains would have amplified the estimation errors so LQR gains of smaller magnitudes were needed for stable control.

## 6.4 Results of Visual Servoing Experiments

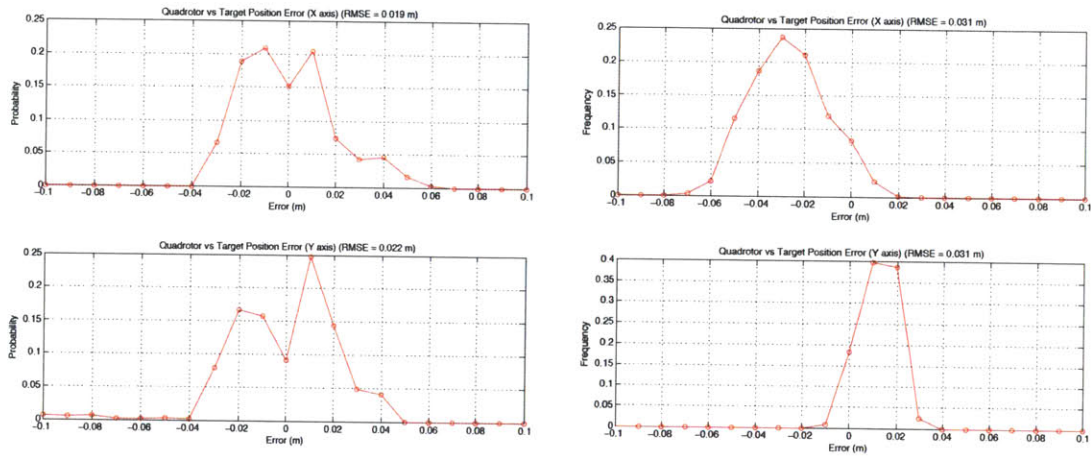
For each of these tests, the quadrotor was flown autonomously for around thirty seconds at a desired altitude of 1.35 m. All of the measurements as well as ground truth data was logged utilizing the LCM framework. This data was then post-processed in MATLAB. The main criteria for evaluation was the error between the quadrotor and the target centroid. We also analyzed the error between the estimated target location computed by the vision system and the ground truth distance. A



sample plot is shown in Figure 6-10(b). Errors in vision system cascaded to errors in the visual servoing performance. This data was used to improve the accuracy of the image coordinate frame to robot coordinate frame transformation.

### 6.4.1 Visual Servoing With Motion Capture Output

Two hovering trials were performed initially to evaluate the controller performance using motion capture state feedback. The first, shown in Figure 6-6(a), uses the motion capture feedback to stabilize around a constant desired position. The RMS error in the  $x$  axis is 0.019 m and 0.022 m in the  $y$  axis. Due to the integrator and offsets, the mean steady state error is under a millimeter in each axis. This demonstrates that the LQR controllers were properly tuned.



(a) Hovering Without Using Visual Position Estimates (b) Hovering Using Visual Position Estimates

Figure 6-6: Visual Servoing Utilizing Motion Capture State Feedback

The second experiment, shown in Figure 6-6(b), uses motion capture feedback but the desired position is determined by the vision system. The inherent inaccuracies in the vision system cause the desired waypoint to move slightly. This results in an increase in the RMS error between the quadrotor and the target centroid. The RMS error was 0.31 m in each axis for this experiment.

### 6.4.2 Visual Servoing with EKF Output

Next, the position controller was evaluated by hovering over a static target without the use of external localization. Figure 6-7 shows the results of this experiment.

The error here is larger than the previous visual servoing experiments which used the motion capture system for state feedback. This reflects the inaccuracies of the state estimate from the filter and the limitations of the LQR position controller. Specifically, this trial had an  $x$  axis RMS error of 0.055 m and a RMS error of 0.075 m in the  $y$  axis.

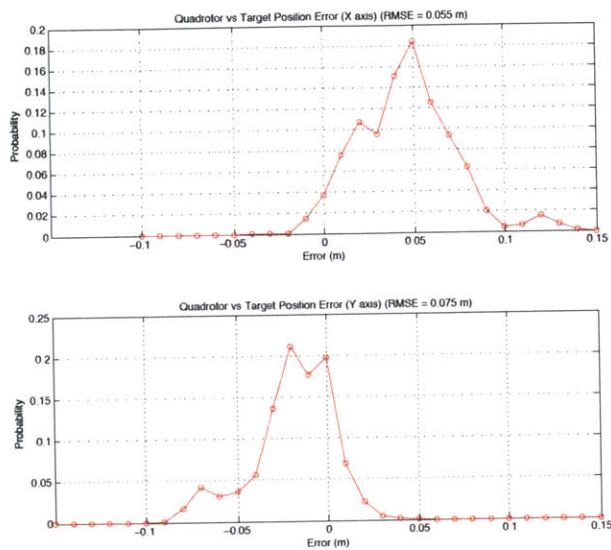


Figure 6-7: Hovering Utilizing EKF State Estimation

## 6.5 Experimental Set-up, Methodology, and Metrics for Dynamic Object Tracking

The final set of experiments tasked the *Hummingbird* with tracking the iCreate through a varied trajectory. The iCreate was programmed to move along the  $x$  axis around 0.5 m, rotate ninety degrees, and move along the  $y$  axis another 0.5 m. The iCreate moved at a constant speed and the trajectory stayed consistent throughout the experiments. The control system first used motion capture feedback for tracking and then relied only on the EKF output for state feedback.

The same performance metrics were used for these experiments. The RMS error was calculated between the quadrotor and the target centroid as recorded by the motion capture system.

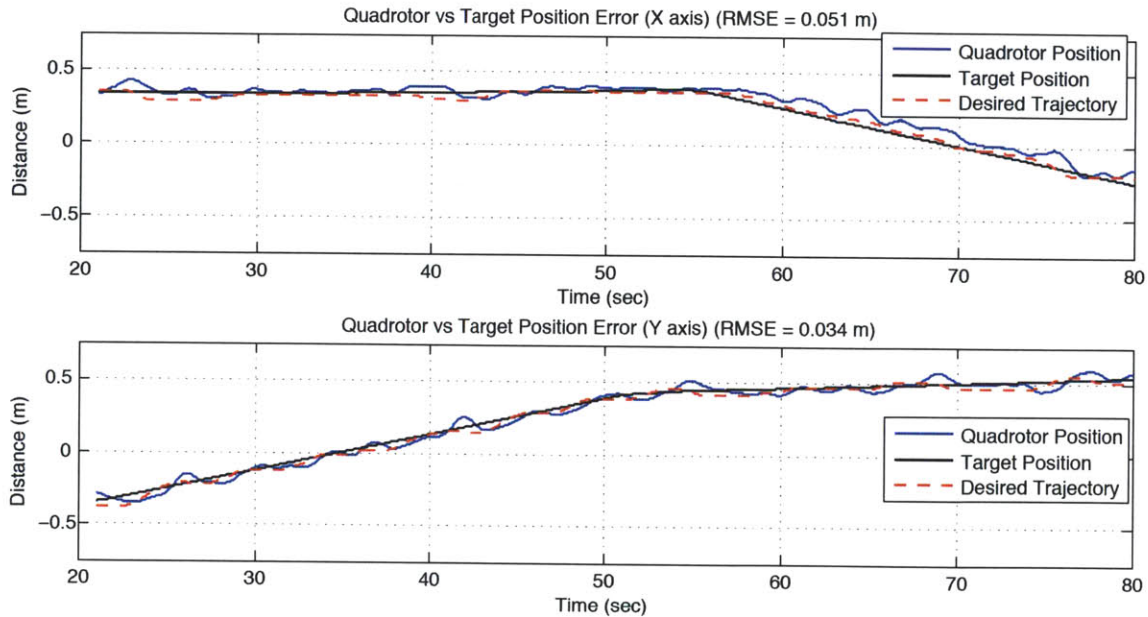


Figure 6-8: Quadrotor Target Tracking Ground Truth Positions Utilizing Motion Capture State Feedback

## 6.6 Results of Dynamic Object Tracking

### Experiments

These experiments showed the accuracy and robustness of the complete visual servoing control system. The quadrotor was able to track the iCreate through a varied trajectory in a consistent manner. With a slight loss in performance accuracy, the quadrotor is shown to track the iCreate using only vision and IMU data.

#### 6.6.1 Dynamic Object Tracking with Motion Capture

##### Output

A sample plot of experimental data is shown in Figure 6-8 for tracking the moving iCreate using motion capture state feedback. This plot shows the desired trajectory in a dashed red line, the ground truth quadrotor position in blue, and the ground truth target position in black. This specific trial had a RMS error of 0.051 m in the  $x$  axis and a RMS error of 0.034 m in the  $y$  axis between the target centroid and the quadrotor trajectory.

The results of visual servoing with a known vehicle pose are shown in Table 6.7. The data was computed over ten consecutive successful trials. The average RMS error was approximately 6.6 cm in the  $x$  axis and 4.2 cm in the  $y$  axis. Several factors affected the performance of tracking but the primary source of error for both tracking experiments was the vision system. While the attitude compensation is effective when the distance between the quadrotor and target is small, as the error between the quadrotor and target increases so does the error between the vision position estimate and the ground truth. Additionally, errors in the vision system output created errors in the desired trajectory.

Table 6.7: Dynamic Object Tracking: Motion Capture State Feedback

Statistic	$x$ Axis RMS error (m)	$y$ Axis RMS error (m)
Mean	0.066	0.042
Minimum	0.051	0.028
Maximum	0.083	0.052
Stand. Dev.	0.010	0.008

### 6.6.2 Dynamic Object Tracking with EKF Output

Finally, the EKF state estimate was used to provide state feedback for the controller while the quadrotor followed a moving surface target. A plot of a sample trial utilizing the EKF output for state feedback is shown in Figure 6-9. Here the blue line is the trajectory of the quadrotor and the black line is the target location. For this trial, the quadrotor trajectory had a RMS error of 0.083 m in the  $x$  axis and a RMS error of 0.095 m in the  $y$  axis. Figure 6-10(a) shows the raw output of the vision system. This is the pixel location of target centroid in the image plane. Lastly, Figure 6-10(b) shows the vision system output accounting for camera offset and quadrotor attitude and finally converted into meters. This is compared to the ground truth error in black as recorded by the motion capture system. The vision system had a RMS error of 0.049 m in the  $x$  axis and a RMS error of 0.076 m in the  $y$  axis.

The tracking errors from the experiments using the EKF output for state feedback are shown below in Table 6.8. These were also computed from ten consecutive successful trials. While the performance is slightly more varied and less accurate



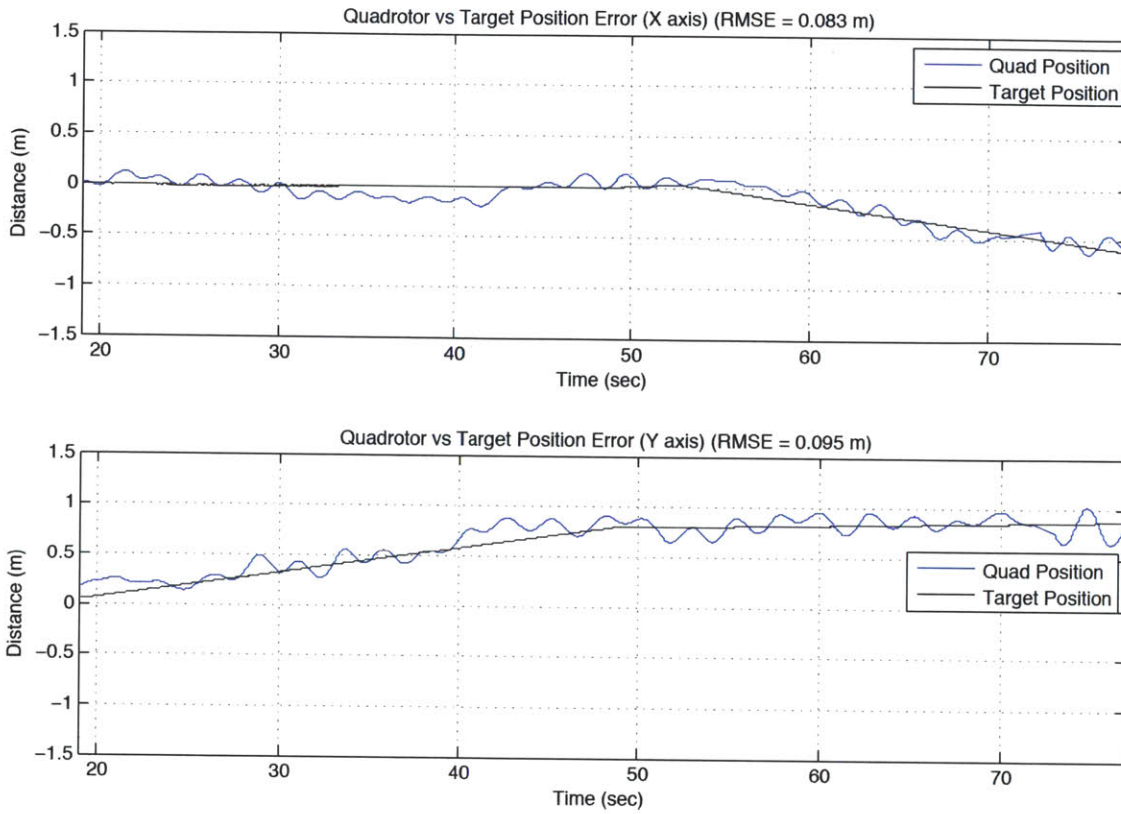
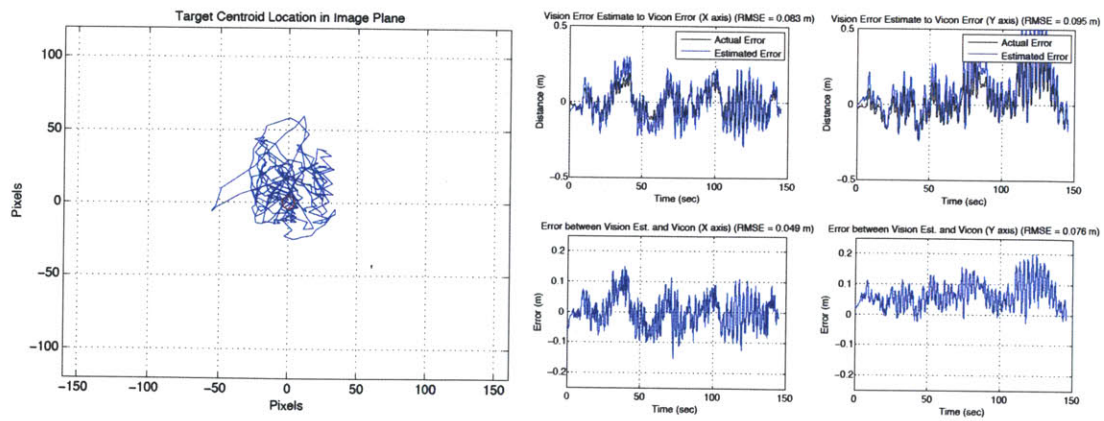


Figure 6-9: Quadrotor Target Tracking Ground Truth Positions Utilizing EKF State Feedback



(a) Target Position in Image Plane

(b) Error In Vision Position Estimation

Figure 6-10: Vision System Performance With EKF State Feedback

Table 6.8: Dynamic Object Tracking: EKF State Feedback

Statistic	$x$ Axis RMS error (m)	$y$ Axis RMS error (m)
Mean	0.068	0.095
Minimum	0.056	0.050
Maximum	0.084	0.134
Stand. Dev.	0.009	0.025

than the tracking with a known pose, the performance is still acceptable. The main reason in the performance difference was the accuracy in the EKF output and time delays. While the EKF output is reasonably accurate, its velocity estimate is noisy. This limits the derivative-like gain that can be used, and which impacts the control quality. There is also an inherent delay using the filter and for our system, this was around 0.06 seconds.

# Chapter 7

## Conclusions

This thesis focuses on identifying objects, specifically whales in an ocean environment using a single camera and computer system mounted onto a payload limited aerial vehicle. It is shown that the same algorithms can be used to identify any object that can be described by homogeneous hue-saturation values which are distinct from the background. By examining video clips of whales taken from various settings and perspectives, the strengths and limitations of the vision system are identified along with methods to improve accuracy. From an overhead perspective, lighting changes are less significant and the majority of false positives come from waves created by the beach or the wakes of the whales.

Through experimental tests using the quadrotor platform, the vision system not only identifies the intended target, but calculates an estimated centroid location which is used to control the quadrotor's position and enable the quadrotor to track the target. It is shown that the target can make directional changes and the quadrotor is able to follow the target through a varying trajectory. Additionally, an Extended Kalman Filter (EKF) is used to estimate the quadrotor state relative to the centroid of the target using position estimates from the vision system as well as Inertial Measurement Unit (IMU) data. This allows the quadrotor to track the surface target independent of an external localization source.

## 7.1 Lessons Learned

We faced several unique challenges in designing an autonomous system to control an aerial vehicle with fast dynamics utilizing estimated state feedback. Several key issues stand out as strong lessons learned which should be noted for future similar robotic endeavors:

- *Open Coding Environment* - With the increased interest in software development for robotics, several tools and libraries have become common for software engineers. Systems such as ROS and LCM include tools to pass messages between modules in a common and efficient way, making it easier to debug code and implement new features. These libraries also include visualization tools and the ability to replay logged messages. This allows the user to simulate real world data without the need to run the physical experiment. Lastly, since the source code is open source, all developers are able to share code and improve on the existing codebase. These tools may require initial overhead to use properly, but the payoff in debugging efficiency is invaluable.
- *Systematic Controller Tuning* - The system identification and controller design process is a good tool to analyze the impact of various control schemes on the desired platform. It is often the case that the controller performs vastly different on the simulated dynamics when compared to the physical system due to model assumptions and inaccuracies. Using the simulated values as a baseline is a good place to start, but developing a systematic tuning process for controllers based on physical intuition saves time. The low-level LQR position controller was tuned beginning with the  $K$  matrix without any offsets or integrators. The offsets were then added and finally an integrator loop to remove steady state errors. The values that resulted in stable performance were then validated in the simulated environment.
- *Avoid Shortcuts* - When developing software, there is a strong desire to get the code to a working level to test it. Once it proves successful, there is little desire



to restructure the code to improve clarity, performance, and remove the temporary measures used for implementation. Unfortunately, this results in future bugs which waste time. Once the software works properly, it is recommended to completely rewrite and restructure the code, removing any artifacts of the debugging process which may cause future problems.

- *Method Suitability* - The limited on-board computational power directed the development of the vision system as well as the control strategy. Computationally fast algorithms were chosen to enable the ability for the robot to operate independently. In the future, a slightly larger robot should be used to help reduce the effects of the additional payload on the system's dynamics and overall flight time.

## 7.2 Future Works

A significant amount of future work is planned before our next field-work campaign. We are working on ways to improve the vision system to avoid the hue-saturation ambiguities between the wakes created by the whales and the whales themselves, perhaps by including intensity information as well. The current method of training defines a rectangular region in hue-saturation space which is unlikely to correspond well to the target. We are investigating alternative ways of training the vision system that eliminate this restriction, perhaps using active contours. We are also looking at using optical flow to estimate translational motion of the quadrotor. The current system relies on having an identified target in order to estimate the quadrotor state using the EKF. This is seen as a limitation to the system.

We would also like to use the vision system on other platforms. The Autonomous Underwater Vehicle (AUV) developed in DRL known as AMOUR is equipped with Point Grey Firefly MV2 cameras as well. This underwater robot has the ability to stabilize in six dimensions. However, there is no way to get position feedback since Global Positioning Satellite (GPS) reception can't be received underwater. Currently, multiple sensor nodes are deployed and using range estimates between the robots and the sensors, an estimated position is computed. Utilizing the vision system, the robot

could identify a target below it and use the vision system to hold its position above the intended target. Experiments tracking a colored panel are shown in Figures 7-1(a) and 7-1(b). Additionally, we wish to track the AMOUR robot when it has surfaced. Equipping the quadrotor with an optical modem receiver would allow the quadrotor to download large files from the underwater robot.

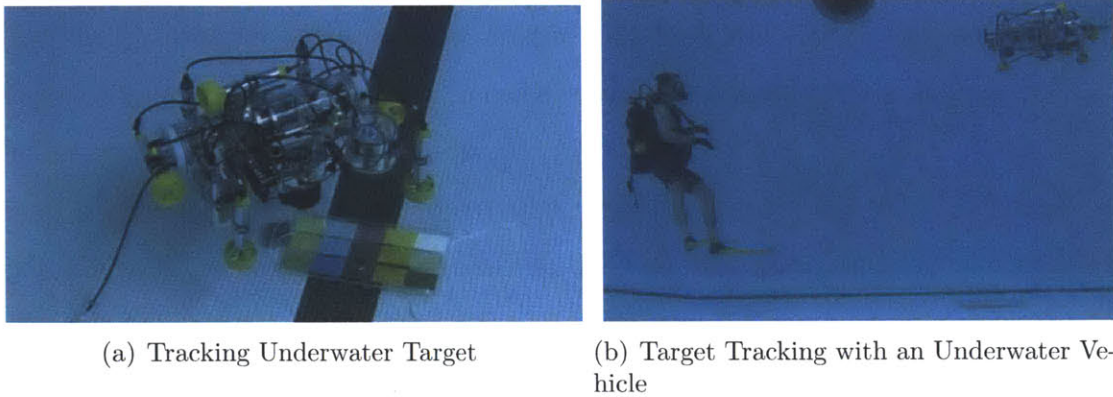


Figure 7-1: Future Work Integrating Vision System With AMOUR AUV

Our ultimate goal is to field test this system. With this in mind, we are also working to better understand the dynamics of our aerial robot to improve its performance. While a simplified model was used for our EKF, a model created through a more rigorous system identification process could prove advantageous if our position estimates from the vision system are less accurate outdoors. Improvements would include using complete nonlinear equations and accounting for aerodynamic effects such as blade flapping. Additionally, a slightly larger quadrotor known as the Ascending Technologies *Pelican* should be used. This quadrotor has a recommended payload of 500 g which would carry our system without any additional strain. This would allow for increased flight time and longer experiments as well.

We wish to incorporate GPS measurements into our system to allow for autonomous outdoor flight. The applications for this are varied. The main goal would be to track whales off the coast of Argentina, closing the loop from idea to implementation. The overall system architecture would vary slightly. Instead of using the LQR controllers developed in this research, the on-board GPS based position controller would be used. Since the quadrotor would be flown significantly higher

on performance. This would eliminate the need to estimate altitude based on vision, which may be infeasible in an outdoor environment due to noise, occlusions, and the presence of multiple targets.

THIS PAGE INTENTIONALLY LEFT BLANK

# Bibliography

- [1] N. Roy A. Bachrach, R. He. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, December 2009.
- [2] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *Proceedings of the SPIE Unmanned Systems Technology XI*, volume 7332, Orlando, F, 2009.
- [3] S. Ahrens, D. Levine, G. Andrews, and J.P. How. Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2643–2648, may 2009.
- [4] John G. Allen, Richard Y. D. Xu, and Jesse S. Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In *VIP '05: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 3–7, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
- [5] E. Altug, J.P. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 72 – 77 vol.1, 2002.
- [6] Omead Amidi. *An Autonomous Vision-Guided Helicopter*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1996.
- [7] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096–1097, May 2010.
- [8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.
- [9] Milomix’s Blog. Rq-1 preactor. <http://www.milomix.wordpress.com>.
- [10] M. Bldsch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 21 –28, may 2010.

- [11] Samir Bouabdallah. Design and control of an indoor micro quadrotor. In *Proc. of Int. Conf. on Robotics and Automation*, 2004.
- [12] Samir Bouabdallah, Pierpaolo Murriero, and Roland Siegwart. Towards autonomous indoor micro vtol. *Autonomous Robots*, 18:171–183, 2005. 10.1007/s10514-005-0724-z.
- [13] H. Bouadi, M. Bouchoucha, and M. Tadjine. Sliding mode control based on backstepping approach for an UAV type-quadrotor. *International Journal of Applied Mathematics and Computer Sciences*, 4(1):12–17, 2008.
- [14] Jean-Yves Bouguet. Camera calibration toolbox for matlab. [http://www.vision.caltech.edu/bouguetj/calib\\_doc/index.html](http://www.vision.caltech.edu/bouguetj/calib_doc/index.html).
- [15] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239, 2001.
- [16] Dr. Gary Rost Bradski and Adrian Kaehler. *Learning opencv, 1st edition*. O’Reilly Media, Inc., 2008.
- [17] G. Bradski. The opencv library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [18] G. Bradski, A. Kaehler, and V. Pisarevski. Learning-based computer vision with intel’s open source computer vision library. *Intel Technology Journal*, 9(2):119–130, May 2005.
- [19] A. Cherian, J. Andersh, V. Morellas, B. Mettler, and N. Papanikolopoulos. Motion estimation of a miniature helicopter using a single onboard camera. In *American Control Conference (ACC), 2010*, pages 4456–4461, July 2010.
- [20] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of UAV. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2010–2016, april 2007.
- [21] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Hilton Head, SC*, volume 2, pages 142–149, 2000.
- [22] P. Corke, R. Mahony, and P. Pounds. Modeling and control of a quad-rotor robot, 2006.
- [23] Defense Industry Daily. Mq-8a fire scout. <http://www.defenseindustrydaily.com/fcs-spinout-plans-detailed-01654/>.
- [24] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1052–1067, June 2007.

- [25] D H Douglas and T K Peucker. Algorithms for the reduction of points required to represent a digitized line or its caricature. *Canadian Cartographer*, pages 112–122, 1973.
- [26] Parrot AR Drone.
- [27] M.G. Earl and R. D’Andrea. Real-time attitude estimation techniques applied to a four rotor helicopter. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 3956 – 3961 Vol.4, Dec. 2004.
- [28] D Eberli, D Scaramuzza, S Weiss, and R Siegwart. Vision based position control for MAVs using one single circular landmark. *Journal of Intelligent and Robotic Systems*, 61:495512, 2011.
- [29] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59:167–181, September 2004.
- [30] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In Martin A. Fischler and Oscar Firschein, editors, *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
- [31] E. Frazzoli, M.A. Dahleh, and E. Feron. Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In *Proc. American Control Conf.*, pages 4102–4107, Chicago, IL, June 2000.
- [32] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32 – 40, jan 1975.
- [33] C. Gablehouse. *Helicopters and autogiros: a history of rotating-wing and V/STOL aviation*. Lippincott, 1969.
- [34] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 361–366, April 2007.
- [35] Tarek Hamel, Robert Mahony, Rogelio Lozano, and James Ostrowski. Dynamic modeling and configuration stabilization for an x4-flyer.
- [36] John Hauser, Shankar Sastry, and George Meyer. Nonlinear control design for slightly non-minimum phase systems: application to v/stol aircraft. *Automatica*, 28:665–679, July 1992.
- [37] Ruijie He, Abraham Bachrach, Michael Achtelik, Alborz Geramifard, Daniel Gurdan, Samuel Prentice, Jan Stumpf, and Nicholas Roy. On the design and use of a micro air vehicle to track and avoid adversaries. *I. J. Robotic Res.*, 29(5):529–546, 2010.

- [38] Ruijie He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1814 –1820, May 2008.
- [39] Bruno Herisse, Francois-Xavier Russotto, Tarek Hamel, and Robert E. Mahony. Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow. In *IROS*, pages 801–806, 2008.
- [40] G. Hoffmann, D.G. Rajnarayan, S.L. Waslander, D. Dostal, J.S. Jang, and C.J. Tomlin. The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, volume 2, pages 12.E.4 – 121–10 Vol.2, oct. 2004.
- [41] Gabriel M. Hoffmann, Haomiao Huang, Steven L. Wasl, and Er Claire J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *In Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2007.
- [42] J.P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *Control Systems Magazine*, pages 28(2):51–64, 2008.
- [43] S. Hrabar, G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3309 – 3316, aug. 2005.
- [44] Albert S. Huang, Edwin Olson, and David Moore. Lcm: Lightweight communications and marshalling. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010.
- [45] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation, ICRA'09*, pages 2408–2413, Piscataway, NJ, USA, 2009. IEEE Press.
- [46] Draganfly Innovations Inc.
- [47] Anil K. Jain, Yu Zhong, and Sridhar Lakshmanan. Object matching using deformable templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18:267–278, March 1996.
- [48] A. Johnson, J. Montgomery, and L. Matthies. Vision guided landing of an autonomous helicopter in hazardous terrain. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3966 – 3971, April 2005.



- [49] Chia-Feng Juang, Wen-Kai Sun, and Guo-Cyuan Chen. Object detection by color histogram-based fuzzy classifier with support vector learning. *Neurocomputing*, 72(10-12):2464–2476, 2009.
- [50] Myungsoo Jun, S.I. Roumeliotis, and G.S. Sukhatme. State estimation of an autonomous helicopter using kalman filtering. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1346 –1353 vol.3, 1999.
- [51] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley series in probability and mathematical statistics. Applied probability and statistics. Wiley, 2005.
- [52] Joseph J. Kehoe, Ryan S. Causey, Mujahid Abdulrahim, and Rick Lind. Waypoint navigation for a micro air vehicle using vision-based attitude estimation. In *Proceedings of the 2005 AIAA Guidance, Navigation, and Control Conference*, 2005.
- [53] Nasser D. Kehtarnavaz, V. Peddigari, C. Chandan, W. Syed, Gilbert R. Hillman, and Bernd Würsig. Photo-identification of humpback and gray whales using affine moment invariants. In *SCIA*, pages 109–116, 2003.
- [54] A. Kreho, N. Kehtarnavaz, B. Araabi, G. Hillman, B. Wrsig, and D. Weller. Assisting manual dolphin identification by computer extraction of dorsal ratio. *Annals of Biomedical Engineering*, 27(6):830–838, 1999.
- [55] A. Kurdila, M. Nechyba, R. Prazenica, W. Dahmen, P. Binev, R. DeVore, and R. Sharpley. Vision-based control of micro-air-vehicles: progress and problems in estimation. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1635 – 1642 Vol.2, Dec 2004.
- [56] Hyung-Soo Lee and Daijin Kim. Robust face tracking by integration of two separate trackers: Skin color and facial shape. *Pattern Recognition*, 40(11):3225 – 3235, 2007.
- [57] Kyoung-Mi Lee and W. Nick Street. Automatic image segmentation and classification using on-line shape learning, 2000.
- [58] J. G. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge University Press, 2000.
- [59] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.
- [60] Sergei Lupashin, Angela Schllig, Michael Sherback, and Raffaello D’Andrea. A simple learning strategy for high-speed quadcopter multi-flips. In *ICRA’10*, pages 1642–1648, 2010.

- [61] P Martin. A different look at output tracking: control of a vtol aircraft. *Automatica*, 32:101–107, 1996.
- [62] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.
- [63] Microdrone md4 200.
- [64] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, To appear.
- [65] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav S. Sukhatme. Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23, 2006.
- [66] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *In Proceedings of the International Symposium on Experimental Robotics*, Delhi, India, Dec 2010. To Appear.
- [67] Torsten Merz, Simone Duranti, and Gianpaolo Conte. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In *In Proc. of the 9th International Symposium on Experimental Robotics*, 2004.
- [68] MiKroKopter.
- [69] A. Mkrtychyan, R. Schultz, and W Semke. Vision-based autopilot implementation using a quadrotor helicopter. In *AIAA Infotech@Aerospace Conference*, 2009.
- [70] T. Hamel N. Guenard and V. Moreau. Dynamic modeling and intuitive control strategy for an x4-flyer. In *In Proceedings of the International Conference on Control and Automation*, pages 141–146, June 2005.
- [71] Wayne Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, Birkerød, Denmark, Denmark, 1985.
- [72] Erik Nice. Design of a four rotor hovering vehicle. Master’s thesis, Cornell University, 2004.
- [73] David Nistr, Oleg Naroditsky, and James Bergen. Visual odometry, 2004.
- [74] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979.
- [75] Pounds P., P. Mahony, P. Hynes, and Roberts J. Design of a four-rotor aerial robot. In *Proceedings of Australian Conference on Robotics and Automation, Auckland*, pages 145–150, Nov. 2002.

- [76] Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 829–836, 2005.
- [77] P Pounds, R Mahony, and Peter Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, February 2010.
- [78] Paul Pounds, Robert Mahony, and Joel Gresham. Towards dynamically-favourable quad-rotor aerial robots. In *In Proc. of Australasian Conference on Robotics and Automation*, 2004.
- [79] Elena Rangelova, Mark J. Huiskes, and Eric J. Pauwels. Towards computer-assisted photo-identification of humpback whales. In *ICIP*, pages 1727–1730, 2004.
- [80] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23:309–314, 2004.
- [81] S. Salazar-Cruz, J. Escareo, D. Lara, and R. Lozano. Embedded control system for a four-rotor UAV. *International Journal of Adaptive Control and Signal Processing*, 21(2-3):189–204, 2007.
- [82] Srikanth Saripalli, James F. Montgomery, and Gaurav S. Sukhatme. Vision-based autonomous landing of an unmanned aerial vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2799–2804, 2002.
- [83] J. Schoonmaker, T. Wells, G. Gilbert, Y. Podobna, I. Petrosyuk, and J. Dirbas. Spectral detection and monitoring of marine mammals. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6946 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, May 2008.
- [84] H. Shim, T.J. Koo, F. Hoffmann, and S. Sastry. A comprehensive study of control design for an autonomous helicopter. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 4, pages 3653–3658 vol.4, dec 1998.
- [85] S. Soatto, R. Frezza, and P. Perona. Motion estimation via dynamic vision. *Automatic Control, IEEE Transactions on*, 41(3):393–413, March 1996.
- [86] Sai Prashanth Soundararaj, Arvind K. Sujeeth, and Ashutosh Saxena. Autonomous indoor helicopter flight using a single onboard camera. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS’09, pages 5307–5314, Piscataway, NJ, USA, 2009. IEEE Press.
- [87] S. Sural, Gang Qian, and S. Pramanik. Segmentation and histogram generation using the hsv color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, pages II–589 – II–592 vol.2, 2002.

- [88] Glenn P. Tournier, Mario Valenti, Jonathan P. How, and Eric Feron. Estimation and control of a quadrotor vehicle using monocular vision and moire patterns. In *In AIAA Guidance, Navigation and Control Conference*, pages 2006–6711. AIAA, 2006.
- [89] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *In AIAA Conference on Guidance, Navigation and Control*. AIAA, 2006.
- [90] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *International Conference on Computer Vision*, pages 212–219, 2005.
- [91] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.
- [92] L.A. Young, E.W. Aiken, J.L. Johnson, R. Demblewski, J. Andrews, and J. Klem. New concepts and perspectives on micro-rotorcraft and small autonomous rotary-wing vehicles. In *Proceedings of the 20th AIAA Applied Aero-dynamics Conference, St. Louis, MO, 2002*.
- [93] Vincent Zalzal. Kfilter - free c++ extended kalman filter library. <http://kalman.sourceforge.net/>.