



Theses and Dissertations

---

2008-06-22

## Vision-Assisted Control of a Hovering Air Vehicle in an Indoor Setting

Neil G. Johnson  
*Brigham Young University - Provo*

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



Part of the [Mechanical Engineering Commons](#)

---

### BYU ScholarsArchive Citation

Johnson, Neil G., "Vision-Assisted Control of a Hovering Air Vehicle in an Indoor Setting" (2008). *Theses and Dissertations*. 1425.

<https://scholarsarchive.byu.edu/etd/1425>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact [scholarsarchive@byu.edu](mailto:scholarsarchive@byu.edu), [ellen\\_amatangelo@byu.edu](mailto:ellen_amatangelo@byu.edu).

VISION-ASSISTED CONTROL OF A HOVERING  
AIR VEHICLE IN AN INDOOR SETTING

by

Neil G. Johnson

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

August 2008



Copyright © 2008 Neil G. Johnson

All Rights Reserved



BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Neil G. Johnson

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

---

Date

---

Tim W. McLain, Chair

---

Date

---

Randal W. Beard

---

Date

---

Mark B. Colton



## BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Neil G. Johnson in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

---

Date

---

Tim W. McLain  
Chair, Graduate Committee

Accepted for the Department

---

Matthew R. Jones  
Graduate Coordinator

Accepted for the College

---

Alan R. Parkinson  
Dean, Ira A. Fulton College of  
Engineering and Technology



## ABSTRACT

# VISION-ASSISTED CONTROL OF A HOVERING AIR VEHICLE IN AN INDOOR SETTING

Neil G. Johnson

Department of Mechanical Engineering

Master of Science

The quadrotor helicopter is a unique flying vehicle which uses the thrust from four motors to provide hover flight capability. The uncoupled nature of the longitudinal and lateral axes and its ability to support large payloads with respect to its size make it an attractive vehicle for autonomous vehicle research. In this thesis, the quadrotor is modeled based on first principles and a proportional-derivative control method is applied for attitude stabilization and position control. A unique means of using an optic flow sensor for velocity and position estimation in an indoor setting is presented with flight results. Reliable hover flight and hallway following capabilities are exhibited in GPS-denied indoor flight using only onboard sensors.

Attitude angles can be reliably estimated in the short run by integrating the angular rates from MEMS gyros, but noise on the signal leads to drift which renders the measurement unsuitable to attitude estimation. Typical methods of providing vector attitude corrections such as accelerometers and magnetometers have inherent weaknesses on hovering vehicles. Thus, an additional vector measurement is necessary to correct attitude readings for long-term flights. Two methods of using image



processing to determine vanishing points in a hallway are demonstrated. The more promising of the two uses a Hough transform to detect lines in the image and forms a histogram of the intersections to detect likely vanishing point candidates. Once the vanishing point is detected, it acts as a vector measurement to correct attitude estimates on the quadrotor vehicle. Results using onboard vision to estimate heading are demonstrated on a test stand. Together, these capabilities improve the utility of the quadrotor platform for indoor flight without the need of any external sensing capability.



## ACKNOWLEDGMENTS

A huge group of people have supported me in completing this thesis work and in pursuing a master's degree. Among them are friends, family, and professors that have meant a lot to me personally and academically.

To start, my wonderful wife Kelley, who married me during my pursuit of this degree, has been a continual support and source of encouragement. My family has been very supportive as well, and I wish to acknowledge my parents, siblings, and in-laws for backing me up in this endeavor. In particular, Martin Evans has been incredibly helpful with actual flight testing conducted around campus.

Several professors have aided me and have become friends during the previous years. Dr. McLain, my advisor, has offered constructive advice and helped me focus on feasible subjects. Dr. Beard has the ability to make difficult concepts seem not only easy but exciting. Dr. Taylor, with his intuition for computer vision, has repeatedly shown how simple things can be if viewed the right way. Dr. Colton has offered help talking through gyroscopic dynamics and other difficult topics.

The MAGICC Lab has been a wonderful place to work, full of brainiacs and well-rounded individuals. Among those that came before, I'd like to acknowledge the help of Andrew Eldredge, Dave Johansen, Steve Griffiths, Brandon Call, Blake Barber, Joe Jackson, and Nate Knoebel. Of those that I have worked with more recently, I'd like to thank Greg Aldridge (for sticking with his helicopter design despite criticism), Travis Millet (for always finding easy solutions to hard problems and explaining things in terms of farm animals), Bryce Ready (for offering concise, clear explanations of the most incomprehensible topics), Ben Heiner (for suggesting a move to the Pico-ITX and assisting greatly with onboard vision development), Andres Rodriguez (for helping me model and re-model the optic flow sensor), Jacob Bishop (for



several helpful insights), James Hall (a great example of thoroughly thinking things through), Evan Andersen (especially for his LaTeX and Lyx help), and Jared Yates (A genius with all things robotic).

I would like to dedicate this thesis to my son, Zachary.



# Table of Contents

<b>List of Tables</b> . . . . .	<b>xii</b>
<b>List of Figures</b> . . . . .	<b>xiv</b>
<b>Chapter 1 Introduction</b> . . . . .	<b>1</b>
1.1 Position and Attitude Estimation . . . . .	1
1.2 System Description . . . . .	3
1.3 Contributions of this Work . . . . .	4
1.4 Document Organization . . . . .	5
<b>Chapter 2 Hover Flight Dynamics and Control</b> . . . . .	<b>7</b>
2.1 Related Work in Hover Flight . . . . .	7
2.1.1 General Helicopter Research . . . . .	8
2.1.2 Quadrotor Helicopter Research . . . . .	9
2.2 Quadrotor Dynamic Model . . . . .	12
2.2.1 Coordinate Frames . . . . .	12
2.2.2 Vehicle State Variables . . . . .	13
2.3 Quadrotor Force and Torque Model . . . . .	16
2.4 Inertial Model for Control . . . . .	18
2.5 Control Loops . . . . .	20
2.5.1 Heading Control . . . . .	20
2.5.2 Attitude Control . . . . .	21
2.5.3 Position and Velocity Control . . . . .	23
2.5.4 Altitude Control . . . . .	24
2.6 Summary . . . . .	26
<b>Chapter 3 Sensors and Estimation</b> . . . . .	<b>27</b>
3.1 Onboard Computing . . . . .	27
3.1.1 Kestrel Autopilot . . . . .	28
3.1.2 Onboard Vision Processing . . . . .	29
3.2 Sensors . . . . .	31
3.2.1 Ultrasound . . . . .	33
3.2.2 Optic Flow Sensor . . . . .	34
3.2.3 Accelerometers and Rate Gyros . . . . .	40
3.3 Attitude Estimation and Data Fusion . . . . .	43
3.3.1 Kalman Filtering Basics . . . . .	43
3.3.2 Attitude Kalman Filter . . . . .	45

3.3.3	Adding Vision Estimates to the Filter . . . . .	48
3.4	Summary . . . . .	50
<b>Chapter 4</b>	<b>Vision-aided Hallway Following . . . . .</b>	<b>53</b>
4.1	Related Computer Vision Research . . . . .	54
4.1.1	Image Preprocessing . . . . .	55
4.1.2	Vanishing Point Estimation Methods . . . . .	56
4.1.3	Pose Estimation and Three-dimensional Tracking Methods . . . . .	57
4.1.4	Visually-Aided Mobile Robot Navigation . . . . .	58
4.2	Image Processing Description . . . . .	60
4.2.1	Edge Detection . . . . .	60
4.2.2	Line Detection . . . . .	64
4.2.3	Line Grouping/Filtering . . . . .	68
4.2.4	Vanishing Point Detection . . . . .	68
4.3	Pose Estimation From Vanishing Points . . . . .	73
4.3.1	Attitude Estimation . . . . .	74
4.3.2	Position Estimation . . . . .	75
4.4	Results . . . . .	76
4.4.1	Burns Line Detection . . . . .	77
4.4.2	Hough Transform Method . . . . .	79
4.5	Summary . . . . .	81
<b>Chapter 5</b>	<b>Results and Discussion of Results . . . . .</b>	<b>83</b>
5.1	Reliable Position Hold . . . . .	83
5.2	Heading Hold using Computer Vision . . . . .	86
5.3	Hallway Following Using The Optic Flow Sensor . . . . .	88
5.4	Discussion of Results . . . . .	89
<b>Chapter 6</b>	<b>Conclusion and Future Work . . . . .</b>	<b>95</b>
6.1	Conclusions . . . . .	95
6.2	Future Work . . . . .	97
<b>References</b>	<b>. . . . .</b>	<b>99</b>
<b>Appendix A</b>	<b>Coaxial Helicopter Dynamics . . . . .</b>	<b>105</b>
A.1	Coaxial Helicopter Platform . . . . .	105
A.1.1	Quadrotor Vehicle Platform . . . . .	105
A.2	Coaxial Helicopter Dynamics . . . . .	106
A.2.1	Physical Properties of the Helicopter . . . . .	107
A.2.2	Rotor Reference Frames and Notation . . . . .	107
A.2.3	Blade Element Method . . . . .	110
A.2.4	Lift and Drag Calculations . . . . .	110
A.2.5	Flybar Modeling . . . . .	111
A.2.6	Flapping Equations of motion . . . . .	114

A.2.7	Specific Modeling for a Coaxial Helicopter with One Actuated Rotor . . . . .	114
A.2.8	Coupling Rotors with Rigid Body . . . . .	118
<b>Appendix B</b>	<b>Perspective Geometry and Hallway Simulation . . . . .</b>	<b>121</b>
B.1	Perspective Geometry . . . . .	121
B.1.1	Homogeneous Representation of Points and Lines . . . . .	122
B.1.2	Rigid-body Transformations in Homogeneous Coordinates . .	123
B.1.3	Camera Projection Model . . . . .	125
B.2	Hallway Projection Simulation . . . . .	127
B.2.1	Line Tracking Method . . . . .	129



## List of Tables

2.1	PD Gains for Quadrotor Control . . . . .	20
4.1	Common Image Operators for Edge Detection . . . . .	63
A.1	Coaxial Helicopter Physical Parameters . . . . .	107



## List of Figures

1.1	Quadrotor Helicopter Image . . . . .	3
2.1	Vehicle Coordinate Frames . . . . .	14
2.2	Quadrotor Top View . . . . .	17
2.3	Roll Control Block diagram . . . . .	21
2.4	Pitch and Roll Angle Tracking . . . . .	22
2.5	Attitude Angle Tracking . . . . .	23
2.6	Position and Velocity Control Loops . . . . .	24
2.7	Position Hold Telemetry Data . . . . .	25
2.8	Altitude Tracking Results . . . . .	26
3.1	Kestrel Autopilot . . . . .	28
3.2	Gumstix Processor . . . . .	29
3.3	Pico-ITX Small Computer . . . . .	31
3.4	Remote Desktop Over Wireless . . . . .	32
3.5	Maxbotix Ultrasound Sensor . . . . .	33
3.6	Optic Flow Sensor . . . . .	34
3.7	Optic Flow Sensor Geometry . . . . .	36
3.8	Optic Flow Sensor Lab Test . . . . .	39
3.9	Quadrotor Free Body Diagram . . . . .	41
3.10	Accelerometer as a Proof Mass . . . . .	42
3.11	Naive Attitude Estimation . . . . .	49
3.12	Attitude Estimation Using Data Fusion . . . . .	50
4.1	Computer Vision Algorithm Flow Chart . . . . .	61
4.2	Gradient Magnitude Image From the Sobel Edge Detector . . . . .	62
4.3	Gradient Direction Image . . . . .	63
4.4	Hough Transform Image . . . . .	67
4.5	Line Tracing Algorithm Results . . . . .	67
4.6	Example Hallway Image . . . . .	69
4.7	Vanishing Point Error Definition . . . . .	71
4.8	Example of Vanishing Point Detection . . . . .	73
4.9	Line Angle Definitions . . . . .	76
4.10	HeliVisionApp Application . . . . .	77
4.11	Burns Line Detector Successes . . . . .	79
4.12	Burns Line Detector Failures . . . . .	80
4.13	Finding All Vanishing Points . . . . .	81

5.1	Position Hold Results . . . . .	84
5.2	Yaw Estimation Using Vanishing Point Detection . . . . .	87
5.3	Comparison of Estimation Noise of Computer Vision and Gyro Integration . . . . .	88
5.4	Hallway Following Telemetry Plot . . . . .	89
5.5	Hallway Following Telemetry Data . . . . .	90
A.1	Coaxial Helicopter Image . . . . .	106
A.2	Rotor Reference Planes . . . . .	108
A.3	Azimuth Direction Definition . . . . .	109
A.4	Additional Degrees of Freedom of a Moving Blade . . . . .	109
A.5	Gyroscopic Flybar Behavior . . . . .	113
A.6	Rotor Blade Flapping Moments . . . . .	115
B.1	Rigid-body Transformation . . . . .	124
B.2	Hallway Simulation . . . . .	129

# Chapter 1

## Introduction

Hovering unmanned air vehicles (UAVs) that are capable of vertical takeoff and landing have many advantages over traditional fixed-wing aircraft such as their ability to maneuver and navigate in confined spaces. These capabilities are important for tasks such as search and rescue missions, surveillance, and disaster aftermath searches. The quadrotor helicopter is a convenient platform for developing hover flight algorithms because it is easy to construct, resilient to vibration, and has decoupled dynamics.

Estimating full pose (attitude and position) of a six-degree-of-freedom vehicle is a significant challenge. Two main approaches are typically used to estimate pose: (1) an array of sensors which independently measure each state variable, and (2) computer vision techniques which seek to measure one or a series of states. In this thesis, it is shown that both methods can be used for pose estimation and that in particular, optical sensors provide position and velocity feedback that can be used as an aid in the flight of UAVs. A major goal of this work is to demonstrate methods of navigating indoor hallways using a combination of sensors and computer vision. This capability is demonstrated and several approaches are compared.

### 1.1 Position and Attitude Estimation

Indoor navigation poses several challenges to flying vehicles which are compounded in hover flight. Among these are the problems of attitude and position estimation. Attitude estimation is a solved problem when high-quality instrumentation is utilized. Inertial navigation units typically use Kalman filtering techniques

to fuse data observations from rate gyros, three-axis accelerometers, and three-axis magnetometers. Integrating the rate gyros provides good estimates of attitude angles over short durations, and errors depend greatly on the quality of the gyros used. Expensive rate gyros such as laser ring gyroscopes make excellent sensors for large aircraft, but small vehicles with limited payloads restrict the available sensors to MEMS gyros which offer lower quality of estimation. Noise on the signal, when integrated, produces unbounded error in attitude estimates in a matter of seconds, especially on vibrating aircraft, which means that an additional sensor reading is necessary to bound the error. Therefore, it is proposed that visual sensing augment the attitude estimate. Specifically, video processing can be used to determine vanishing points which provide a set of non-drifting vector measurements. This thesis proposes the use of a particular algorithm that uses a Hough transform to detect lines and vanishing points to correct drift in attitude estimation.

Position estimation also introduces challenges that we often take for granted. A human's ability to ascertain his/her location relative to obstacles and surroundings is phenomenal. For an autonomous vehicle to determine position, active sensors are often employed such as laser range finders and ultrasound ranging devices. In outdoor environments, global positioning systems (GPS) give reliable passive absolute position readings by triangulating a receiver's position with respect to orbiting satellites. GPS signals are not typically usable indoors, and in any case, they give no indication of relative orientation to surrounding objects unless a map is provided. Despite these difficulties, great advances have been made in the areas of position and attitude sensing, and this thesis seeks to take the most promising applications and put them into effect on a hovering vehicle. To overcome the challenge of estimating position, it is proposed that an optic flow sensor be used. Methods for using this sensor to estimate velocity and to dead-reckon position are developed and results showing much-improved hover flight are demonstrated.

## 1.2 System Description

The field of ground robotics has developed a firm foundation of research in cooperative tasking, indoor environment navigation, and vision-aided navigation. Aerial robotics has now emerged as a prime research field, and estimation and control theory have been extensively applied to both fixed-wing and hovering vehicles. The Multi-AGent Intelligent Coordination and Control Laboratory (MAGICC Lab) at Brigham Young University has focused first on ground robotics and then aerial robotics, and a hardware platform based on the Kestrel autopilot has been developed for flying fixed-wing vehicles [1]. This same autopilot has been used to apply adaptive quaternion control in both hover and steady-level flight for a Tailsitter VTOL (Vertical Take-off and Landing) aircraft [2]. These technologies are now being leveraged for use on other types of flying vehicles.



Figure 1.1: **Quadrotor Helicopter:** The quadrotor helicopter is the primary vehicle used in this research. It is capable of sustained hover flights and of holding substantial payloads.

The vehicle chosen for flight demonstrations in this thesis is the quadrotor helicopter pictured in Figure 1.1. This four-rotor hovering vehicle is capable of vertical take-off and landing and sustained hover. The quadrotor is steadily becoming a

favorite aircraft among research groups due to its decoupled longitudinal and lateral dynamics, its payload capacity, and its relative simplicity of design. The goal of this thesis is to demonstrate methods for combining knowledge from the domains of dynamic control and estimation with computer vision techniques for estimating attitude and position and to demonstrate this capability by flying down an indoor hallway. Several existing methods have been tested and tried in simulation and hover flight using the Kestrel autopilot and additional sensors. A dynamic model for the quadrotor aircraft is presented including a force and torque analysis and differential equations describing the six-degree-of-freedom model. A proportional-derivative control system is chosen for its simplicity and effectiveness and flight results demonstrating attitude and position control are included.

### 1.3 Contributions of this Work

This thesis work combines helicopter control with visual perception and makes several contributions to the body of knowledge regarding quadrotor control and sensing. Building on knowledge from many domains, this work demonstrates how onboard sensors on a flying vehicle can be improved and assisted by computer vision. The main contributions are the following:

- The first example to our knowledge of using an optic flow sensor on a quadrotor helicopter with reliable position estimation demonstrated in position hold. This was demonstrated with a seven-minute flight and several other long flights with no pilot corrections.
- The first example to our knowledge of computing heading from vanishing points onboard a helicopter and using that estimate to maintain a desired heading in real-time.
- Several methods for determining vanishing points are used for attitude estimation. A functional vanishing point detection algorithm was developed and implemented on the quadrotor with flight results.

- A reliable code base has been developed for estimating pose and controlling quadrotors in hover.
- Methods for modeling the projection of a hallway onto the imaging plane of a monocular camera have been explored and modeled.
- Flight down a hallway has been demonstrated using velocity commands from a human pilot.
- Onboard vision has been developed as a usable sensor with real-time visualization on a remote workstation. This greatly aids computer vision algorithm development and testing.

#### **1.4 Document Organization**

In Chapter 2, dynamic models for control and estimation are presented for the quadrotor helicopter and flight test results show the improved hover capability due to optical sensors. Estimation methods using data fusion from multiple sensors are presented showing improvements in real-world environments in Chapter 3. Computer vision techniques for detecting lines in indoor environments and estimating orientation and position are presented and compared in Chapter 4. Chapter 5 shows flight results of a hovering vehicle in a hallway using an array of onboard sensors including vision processing. Problems encountered in combining visual estimation with sensor estimation are discussed. Finally, conclusions and possible future work are discussed in Chapter 6. Some previous work on modeling coaxial helicopters is included in Appendix A and some background material on perspective geometry is included in Appendix B.



## Chapter 2

### Hover Flight Dynamics and Control

Helicopters come in a variety of forms, but in general they have the ability to take off and land vertically and to maneuver in a holonomic fashion, i.e. they can move in any direction with equal ease. The quadrotor helicopter consists of four simple propellers arranged at equal distance from the vehicle's center. These helicopters are capable of lifting large amounts of weight and have relatively simple dynamic characteristics when compared with single-rotor and coaxial helicopters. Little to no coupling exists between longitudinal and lateral axes, and force and torque calculations are simple. This chapter presents a six-degree-of-freedom model for the quadrotor helicopter and demonstrates linear control techniques for controlling the aircraft in the hover flight regime using successive loop closure. Flight results are presented to show stable hover flight and low-velocity tracking near hover.

The coaxial helicopter was initially used for both simulation and flight-testing. However, all the results presented in this thesis were collected on a quadrotor helicopter platform due to constraints on weight and instrumentation. A complete model of the forces and torques generated by the twin rotors of the coaxial helicopter is included in Appendix A along with a discussion of rotor and flybar dynamics. This chapter will discuss the quadrotor model and the control systems selected for hover flight.

#### 2.1 Related Work in Hover Flight

A great deal of research has been conducted on hovering vehicles. The related work which directly applies to this research is split into two main categories: general

helicopters and quadrotor helicopters. Initial research for this thesis focused on single-rotor and coaxial helicopters, but towards the end the emphasis was shifted to the quadrotor helicopter. Because these types of vehicles are similar in nature, work related to both is included here.

### **2.1.1 General Helicopter Research**

Large and small-scale autonomous helicopters have been developed at many schools and research institutions. Carnegie Mellon University [3], Stanford [4], MIT [5], USC [6] and several other schools have flown single-rotor helicopters autonomously with high levels of control. In association with the BYU MAGICC Lab, helicopter research involving stereo vision has been completed with NASA and the U.S. Army [7]. Commonly, GPS is used to estimate position outdoors, and inertial navigation units are used to estimate attitude. The work by Saripalli et al. [8] describes the estimation and control design of two helicopters. One, the USC helicopter, uses expensive inertial measurement sensors which cost an order of magnitude more than their helicopter platform. The other, developed by the CSIRO group at the University of Queensland, uses low-cost inertial sensors coupled with vision sensors to provide a position estimate. While avionics systems have been developed for helicopters in many research institutions, expensive hardware is typically required to demonstrate reliable flight.

Researchers at Carnegie Mellon University have demonstrated autonomous flight and navigation of helicopters using inertial sensors and visual odometry [9]. First, detailed state-space models of the helicopter dynamics were estimated using system identification techniques in the frequency domain. This model provides a base for designing effective controllers despite stabilizer bar dynamics and active yaw rate damping on commercial grade helicopters [3]. The simplified model developed by Mettler was very influential in deriving the coaxial helicopter model included in Appendix A.

In [10], a helicopter equipped with stereo vision cameras and inertial sensors is used to collect data which is then post-processed to demonstrate a combined visual

and inertial navigation system. The capability of using visual recognition as feedback has been established on helicopters in GPS-denied environments where known features such as building windows can be identified in the image frame [11], [12]. Flight results are presented in structured environments under conditions of GPS dropout using visual odometry and visual servoing techniques. Rather than simply handling GPS dropouts, the work in this thesis is targeted at completely replacing the GPS sensor with onboard sensors for estimating position and velocity. In addition, different approaches are required for handling visual pose estimation in indoor and outdoor environments. Point feature tracking employed in the systems mentioned do not typically work well in indoor environments which are not rich with well-defined point features. Indoor environments are generally rich with edges and lines which provide more reliable estimates, as corroborated by Kemp in [13].

### **2.1.2 Quadrotor Helicopter Research**

The quadrotor helicopter has become a popular vehicle for conducting hover flight research. Universities around the world have built quadrotors, and thus a great deal of related work on quadrotors is taking place. In [14], Tayebi proposes a feedback control scheme with one proportional and two derivative terms which he calls PD<sup>2</sup> control. The proportional term is related to the vector quaternion and the derivative terms are related to the vehicle angular velocity and the vector quaternion velocity. His results show exponential attitude stabilization and an accompanying video demonstrates the controller implemented on a test stand quadrotor. While useful, quaternions are not necessary to describe small attitude angles ( $< 45^\circ$ ). Dynamics of a tethered vehicle also vary substantially from real-flight behavior which Tayebi does not address.

McKerrow [15] provides a useful model of the Draganflyer helicopter, a commercially available quadrotor aircraft with rate damping to aid in piloted flight. He describes many of the dynamic characteristics of the quadrotor which make it difficult to control. In particular, he mentions that it is an underactuated system because it uses four actuators to produce motion in six degrees of freedom; that only very small

forces oppose the motion giving it a very dynamic behavior; and that high coupling exists between attitude angles and directional accelerations. These dynamic characteristics provide challenges to successful flight. Included in McKerrow’s model are gyroscopic torques and Coriolis accelerations which provide a more detailed model than is presented in this work. However, it will be shown that adequate control was provided by the sensor suite of our quadrotor to hover in a tight radius without human intervention. This is in contrast to McKerrow’s 2004 results in which stable hover was very difficult to achieve.

In [16], Altug et al. uses an offboard camera to determine the pose of a quadrotor helicopter in simulation and restricted flight. Simulations show results for a feedback-linearization control approach as well as a backstepping-like method. All pose estimation is computed offboard using an overhead camera, and a tethering system restrains motion in the  $x$ - and  $y$ -directions. Thus while this is a good demonstration of control methodologies, it was not proven to be an effective position control scheme on a truly autonomous system. In a later effort [17], a dual camera approach is employed to provide a more reliable pose estimate and results are collected on a tethered system.

Bouabdallah et al.[18] present several design considerations for designing micro quadrotors as part of the OS4 project at the Swiss Federal Institute of Technology, but the results are limited to simulation and test-bench data collection. In [19] results using PID and LQ controllers are collected by the same authors, and an autonomous flight is attempted, though the vehicle is still tethered for power and apparently does not perform stable position hold. In later work on the same test stand [20], sliding mode and backstepping controllers are tested with the conclusion that the backstepping controller is more suitable. While these results are useful comparisons of various control methodologies, they do not present comprehensive control strategies for position hold nor do they present convincing results that maintainable hover was obtained.

Promising flight results are obtained by Roberts et al. [21] in which a quadrotor is outfitted with infrared triangulation sensors that allow the vehicle to navigate

hands-off for extended periods of time within an enclosed environment. The infrared sensors maximize their distance from nearby walls allowing the vehicle to hover in a fairly tight radius. However, the implementation of the special infrared sensors only appears to work in fully-enclosed areas. Thus, this particular application has not been shown to have widespread applicability to real-world situations.

Several schools have developed testbeds involving autonomous flying vehicles. Stanford, for example, has a set of quadrotor vehicles called STARMAC [4] which demonstrates reliable outdoor flight using carrier phase differential GPS positioning. The STARMAC testbed is now on its second design phase and has presented useful results regarding quadrotor performance at large deviations from the hover flight regime [22]. The STARMAC group appears to rely on established absolute positioning systems such as GPS (outdoor) and overhead cameras (indoor) to provide position feedback.

A unique indoor testbed involving quadrotor vehicles as well as ground vehicles has been developed at MIT with the primary purpose of developing algorithms for fault detection, isolation, and recovery [23]. Using a very precise VICON camera system, they demonstrate the ability to control several off-the-shelf quadrotor vehicles simultaneously. The VICON system is capable of sub-millimeter position accuracy and sub-degree attitude estimation. Using LQR controllers based on linear dynamics, very stable flights of quadrotors are demonstrated. This allows their research to focus on higher level tasks such as multi-agent tasking and health monitoring for persistent surveillance and mission planning. Sophisticated camera systems like VICON are expensive and require offboard cameras, thus they do not represent real-world position sensing capabilities. The work in this thesis targets reasonable position accuracy using only onboard sensors which cost much less than the VICON system.

In summary, the quadrotor is growing very popular as a test vehicle for autonomous flight and navigation. Much work has been done to develop effective attitude-stabilizing controllers, but nearly all institutions that work on such systems admit to many challenges with controlling the vehicle in real flight. Flight results in

this thesis of a stable hands-off hover flight rank among the better results documented in the literature.

## 2.2 Quadrotor Dynamic Model

The quadrotor is actuated by modulating the throttle command of each of the four motors. Changing the throttle of all motors together produces vertical motion. Pitching moments are produced by increasing the thrust of the front motor while decreasing that of the rear motor or vice versa. Roll moments are produced in a similar fashion by adjusting the thrust of the right motor with respect to the left. Yawing moments are slightly more subtle: if the front and rear motors (which spin clockwise) spin faster than the left and right motors (spinning counterclockwise), yawing results due to the difference in rotor drag moments on the respective motors.

To develop control laws and estimation schemes, it is useful to model the dynamics of the quadrotor and understand how forces and torques are generated on the vehicle. This section will develop such a model by first introducing the coordinate frames and state variable notation that will be used throughout this thesis. Then, dynamic and kinematic differential equations will be shown based on the quadrotor model found in [24]. A simple force and torque model is derived and used in control laws developed later in the chapter.

### 2.2.1 Coordinate Frames

The attitude and position of the quadrotor aircraft can be described by a series of state variables. It is desired to express the position of the aircraft in the world frame which is oriented with its  $x$ -axis pointing north,  $y$ -axis facing east, and  $z$ -axis directed toward the center of the Earth. The vehicle frame is translated from the world frame by a position vector  $P_v$ , which is composed of the three position states  $p_n$ ,  $p_e$ , and  $p_d$ . The height above the world origin is denoted  $h$  and is simply  $-p_d$ . The vehicle frame is therefore a coordinate frame which translates with the vehicle but remains oriented parallel to the world frame.

Three additional frames shown in Figure 2.1 are necessary to describe the vehicle's orientation using the Euler 3-2-1 coordinate system. Three angles,  $\psi$ ,  $\theta$ , and  $\phi$  describe the vehicle's rotations relative to these three successive frames. First, the vehicle-1 frame is found by rotating the vehicle in a right-handed rotation about the vehicle  $z$ -axis by the angle  $\psi$ . The vehicle-2 frame is then found by rotating about the vehicle-1  $y$ -axis by the angle  $\theta$ . Finally, the body-frame is found by rotating about the vehicle-2  $x$ -axis by the angle  $\phi$ . The vehicle's pose can then be described by the three position states  $p_n$ ,  $p_e$ , and  $p_d$ , and the three Euler angles  $\phi$ ,  $\theta$ , and  $\psi$ . Other systems can be used for describing the attitude angles, but Euler angles are chosen for their intuitive meanings and because attitude angles near Euler angle singularities do not occur in regular quadrotor flight.

### 2.2.2 Vehicle State Variables

The state variables which describe the rigid-body dynamics and kinematics of a general hovering aircraft are the following:

- $p_n$  - the position of the helicopter in the inertial  $x$ -direction (north position)
- $p_e$  - the position of the helicopter in the inertial  $y$ -direction (east position)
- $h$  - the height of the helicopter above the ground (altitude, negative inertial  $z$ -direction)
- $u$  - the velocity in the body-frame  $x$ -direction (from the center of mass towards the nose of the aircraft)
- $v$  - the velocity in the body-frame  $y$ -direction (from the center of mass towards the right "wing")
- $w$  - the velocity in the body-frame  $z$ -direction (from the center of mass towards the bottom of the aircraft)
- $\phi$  - the roll angle defined with respect to the vehicle-2 frame

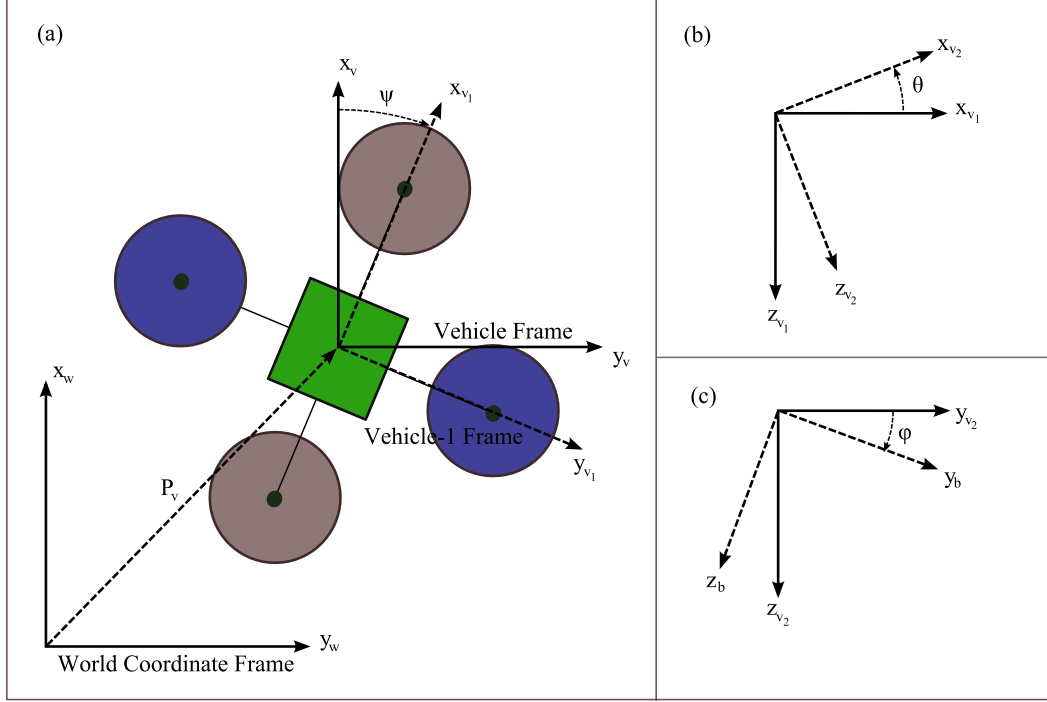


Figure 2.1: **Vehicle Coordinate Frames:** Transformations from the world frame to the body frame through all intermediate coordinate frames are demonstrated. (a) The world frame is a global inertial frame with the  $x_w$ -direction facing north,  $y_w$  facing east, and  $z_w$  down. The vehicle frame is simply translated with respect to the world frame with the origin at the vehicle's center of mass. The vehicle-1 frame shares its origin with the vehicle frame, but its axes are rotated about the vehicle frame's  $z$ -axis by the angle  $\psi$ . (b) The vehicle-2 frame is constructed by rotating about the vehicle-1  $y$ -axis by the angle  $\theta$ . (c) Finally, the body frame is found by rotating about the vehicle-2  $x$ -axis by the angle  $\phi$ .

- $\theta$  - the pitch angle defined with respect to the vehicle-1 frame
- $\psi$  - the yaw angle defined with respect to the vehicle frame
- $p$  - the roll rate measured around the  $x$ -axis in the body frame
- $q$  - the pitch rate measured around the  $y$ -axis in the body frame
- $r$  - the yaw rate measured around the  $z$ -axis in the body frame

The twelve equations describing the rigid-body dynamics and kinematics can be derived using first principles. In the following equations, the changes in the states are related to the current values of the states and the inputs. The total thrust force from

all four motors is denoted  $F_T$ , and the quantities  $\tau_\phi$ ,  $\tau_\theta$ , and  $\tau_\psi$ , represent torques about the  $x$ ,  $y$ , and  $z$  axes. Derivations of how these forces and torques relate to the individual motor commands are given in the next section. Throughout the paper,  $c\theta$  and  $s\phi$  are defined to represent  $\cos \phi$  and  $\sin \phi$  respectively. The twelve differential equations describing the quadrotor's dynamics and kinematics are:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\theta s\psi \\ s\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.1)$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \begin{bmatrix} -g \sin \theta \\ g \cos \theta \sin \phi \\ g \cos \theta \cos \phi \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F_T \end{bmatrix} \quad (2.2)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.3)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_4(p^2 - r^2) \\ \Gamma_6 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 \tau_\phi + \Gamma_4 \tau_\psi \\ \frac{1}{J_y} \tau_\theta \\ \Gamma_4 \tau_\phi + \Gamma_7 \tau_\psi \end{bmatrix} \quad (2.4)$$

where

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma} \quad (2.5)$$

$$\Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma} \quad (2.6)$$

$$\Gamma_3 = \frac{J_z}{\Gamma} \quad (2.7)$$

$$\Gamma_4 = \frac{J_{xz}}{\Gamma} \quad (2.8)$$

$$\Gamma_5 = \frac{J_z - J_x}{J_y} \quad (2.9)$$

$$\Gamma_6 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma} \quad (2.10)$$

$$\Gamma_7 = \frac{J_x}{\Gamma} \quad (2.11)$$

$$\Gamma = J_x J_z - J_{xz}^2. \quad (2.12)$$

Derivations of these equations using a similar notation are taken from lecture notes from Dr. Randal Beard [24]. This model provides a straightforward manner of relating the forces and torques to the states at a given point in time provided initial conditions. From these general equations, a simplified model for control can be easily developed.

### 2.3 Quadrotor Force and Torque Model

Describing the relations for the forces and torques due to the propellers of a quadrotor is simple when compared to the rotor dynamic equations of the coaxial helicopter. It is typically assumed that the effects of flapping blade motion contribute little to the behavior of quadrotors, and therefore we will use a simple model for the forces and torques generated by the four propellers. These forces and torques can then be used in the above dynamic equations (2.1-2.4) to provide a full simulation of the quadrotor. We also make the simplifying assumption that aerodynamic forces and moments due to the body of the aircraft are negligible near hover. Therefore, our model consists of forces and moments due to gravity and the thrust generated by the four propellers. The moment arm of each motor is assumed to be the same, denoted  $L$ , as shown in Figure 2.2.

The total force  $F_T$  due to the motors is then

$$F_T = F_f + F_r + F_b + F_l. \quad (2.13)$$

The rolling and pitching torques are caused by differential thrust in the lateral and longitudinal directions respectively:

$$\tau_\phi = L(F_l - F_r), \quad (2.14)$$

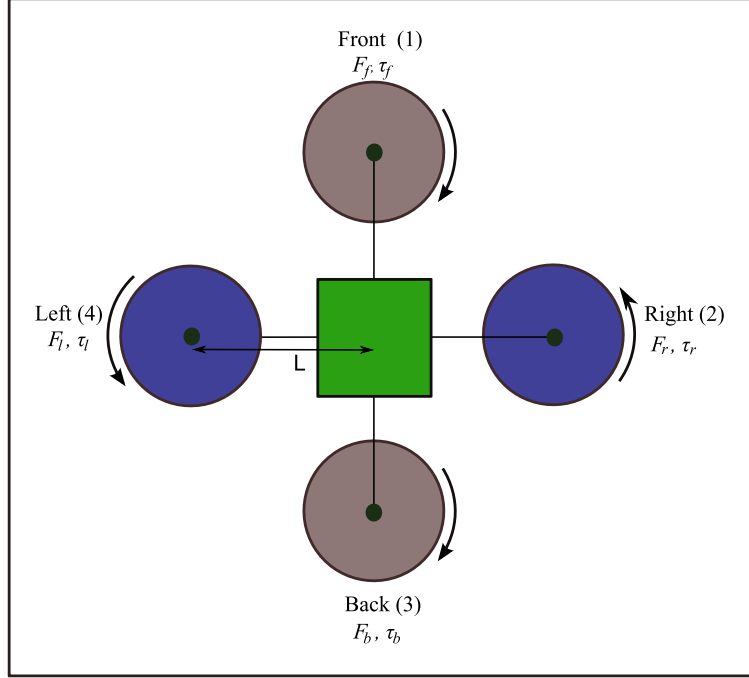


Figure 2.2: **Quadrotor Top View:** Each motor produces an upward force,  $F$ , and a torque,  $\tau$ , on the vehicle. As indicated, the front and back motors spin clockwise and the right and left motors spin counter-clockwise.

$$\tau_\theta = L(F_f - F_b). \quad (2.15)$$

To generate yawing motion, the front and back (clockwise) motors spin faster or slower relative to the left and right (counter-clockwise) motors causing a difference in rotor drag forces and an effective yawing moment. The direction of the torque is opposite the direction of the propeller motion, thus

$$\tau_\psi = \tau_r + \tau_l - \tau_f - \tau_b. \quad (2.16)$$

Force tests conducted on the motors showed a mostly linear correlation between the PWM (pulse width modulation) output of speed controllers and the the force and torque generated by the motor. Therefore, we will model the force and torque of each motor according to

$$F_* = k_1 \delta_* \quad (2.17)$$

$$\tau_* = k_2 \delta_*, \quad (2.18)$$

where  $k_1$  and  $k_2$  are constants that must be determined experimentally,  $\delta_*$  is the motor command signal, and  $*$  represents  $f$ ,  $r$ ,  $b$ , and  $l$ . In matrix form, we can express the forces and torques on the quadrotor as

$$\begin{bmatrix} F_T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -Lk_1 & 0 & Lk_1 \\ Lk_1 & 0 & Lk_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{bmatrix} \begin{bmatrix} \delta_f \\ \delta_r \\ \delta_b \\ \delta_l \end{bmatrix}. \quad (2.19)$$

Control strategies will be derived in terms of these forces and torques. Motor commands are then found by inverting the above matrix.

## 2.4 Inertial Model for Control

A simplified model of the quadrotor dynamics suitable for control can be derived from the equations described above. First, assuming small angles, Equation 2.3 becomes

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (2.20)$$

To simplify the angular rate equations (2.4), two assumptions are made. First, the coriolis terms  $pq$ ,  $qr$ , and  $pr$  are assumed to be negligible. In addition, due to the symmetry of the quadrotor, the off-axis moments of inertia are neglected. The new angular rate equations are then

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\theta \\ \frac{1}{J_z} \tau_\psi \end{bmatrix} \quad (2.21)$$

and, combining Equations 2.20 and 2.21, the attitude dynamics can be simplified to

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\theta \\ \frac{1}{J_z} \tau_\psi \end{bmatrix}. \quad (2.22)$$

This simplified model makes it clear that the attitude dynamics are due primarily to the torques induced by differential thrust on the motors.

To develop a simplified model for the position dynamics, Equation 2.1 is differentiated yielding

$$\begin{bmatrix} \ddot{p}_n \\ \ddot{p}_e \\ \ddot{h} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\theta s\psi \\ s\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}. \quad (2.23)$$

Neglecting the coriolis terms and assuming small angles, Equation 2.2 becomes

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g - \frac{F_T}{m} \end{bmatrix}. \quad (2.24)$$

Combining Equations 2.23 and 2.24 results in a simple model for the position dynamics:

$$\begin{bmatrix} \ddot{p}_n \\ \ddot{p}_e \\ \ddot{h} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \begin{bmatrix} -c\phi s\theta c\psi - s\phi s\psi \\ -c\phi s\theta s\psi + s\phi c\psi \\ c\phi c\theta \end{bmatrix} \frac{F_T}{m} \quad (2.25)$$

where  $g$  is the gravitational constant and  $F_T$  is the total thrust produced by the four motors. According to this simple model, the lateral and longitudinal dynamics are uncoupled and directional accelerations are directly tied to attitude angles. This suggests that nested linear controllers can effectively control the system. Therefore, successive loops are closed around attitude and then position as discussed in the next section.

## 2.5 Control Loops

Attitude tracking is essential to position tracking and thus acts as the inner loop. Velocity and position controllers are built as outer loops whose output is a set of desired attitude angles. Yawing torques are easily cancelled using equal force on each motor and yawing dynamics are treated separately from all the other loops with good results. Thus, four systems of loops were designed:  $\theta \rightarrow p_n$ ,  $\phi \rightarrow p_e$ ,  $h$ , and  $\psi$ . Each of the control systems will be described below. The PD gains used in hover flight are presented in Table 2.1.

Table 2.1: PD Gains for Quadrotor Control

Control Loop	$K_p$	$K_d$	Units of output
Heading Hold	150	50	PWM
Roll ( $\phi$ )	75	15	PWM
Pitch ( $\theta$ )	75	20	PWM
Lateral Velocity ( $v$ )	0.6	0.3	radians of desired roll
Longitudinal Velocity ( $u$ )	0.6	0.3	radians of desired pitch
Lateral Position ( $p_y$ )	0.4	0.1	m/s of desired velocity
Longitudinal Position ( $p_x$ )	0.4	0.1	m/s of desired velocity
Altitude ( $h$ )	80	60	PWM

### 2.5.1 Heading Control

Two methods of controlling heading were attempted. First, a simple yaw-damping scheme was implemented. This effectively slowed yawing, but the vehicle slowly drifted around the yaw axis. This drift is attributed to imperfections in the motor correlation to PWM output and imperfect motor mounting. The second method used PD control and was very effective in holding heading. Provided an accurate heading estimate, the vehicle was able to hold heading consistently. Gains were tuned using a yaw plate which constrained vehicular motion to the yawing direction. These gains required no additional tuning in real flight.

### 2.5.2 Attitude Control

It is essential to control attitude angles well to achieve autonomous flight. Since the quadrotor has decoupled lateral and longitudinal dynamics, control can be applied independently on each axis. PD controllers were applied to control pitch and roll. A block diagram representing the roll controller is shown in Figure 2.3. This inner loop contains two successive loops. Rather than numerically differentiating roll error to compute the derivative of the error, a desired roll rate of zero was used and roll rate error was computed according to

$$\frac{d}{dt}(\phi^c - \phi) \approx 0 - p \quad (2.26)$$

where  $\phi^c$  is the commanded roll angle. Control effort was saturated to use higher control gains without causing divergence. This inner loop was tuned empirically by setting desired angle commands from an RC transmitter. The pitch inner loop is identical and therefore not pictured. All of the control gains for the system are described in Table 2.1.

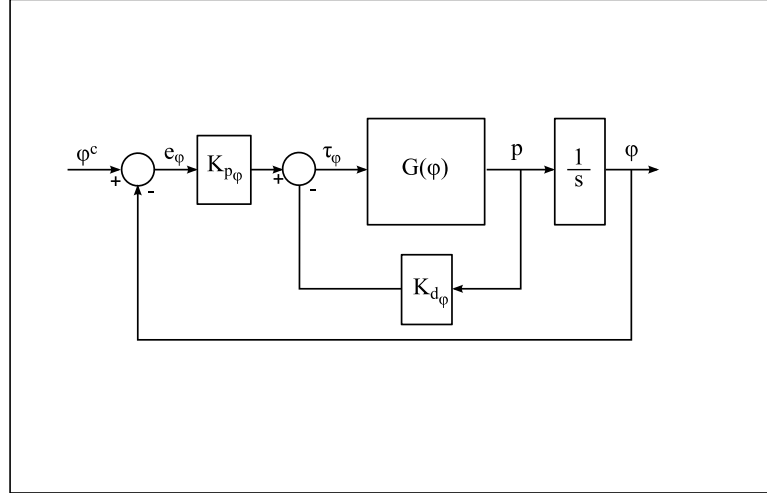


Figure 2.3: **Roll Block Diagram:** PD Control is sufficient to control attitude angles.

As the block diagram suggests, the roll control effort  $\tau_\phi$  is calculated according to

$$\tau_\phi = K_{p_\phi}(\phi^c - \phi) - K_{d_\phi}p. \quad (2.27)$$

Holding nonzero attitude angles on a quadrotor vehicle is problematic in an indoor setting because the vehicle quickly accelerates in the horizontal direction. Therefore, by constraining the quadrotor's motion in the lab, results were collected for pitch and roll angles before testing in real flight. First, the quadrotor was constrained to rotate about the  $y$ -axis and pitch angles were tracked as shown in Figure 2.4 (a). Then, the quadrotor was constrained to rotate about the  $x$ -axis and roll angle tracking was shown in Figure 2.4 (b). These results show the ability to track within a few degrees of the desired angle.

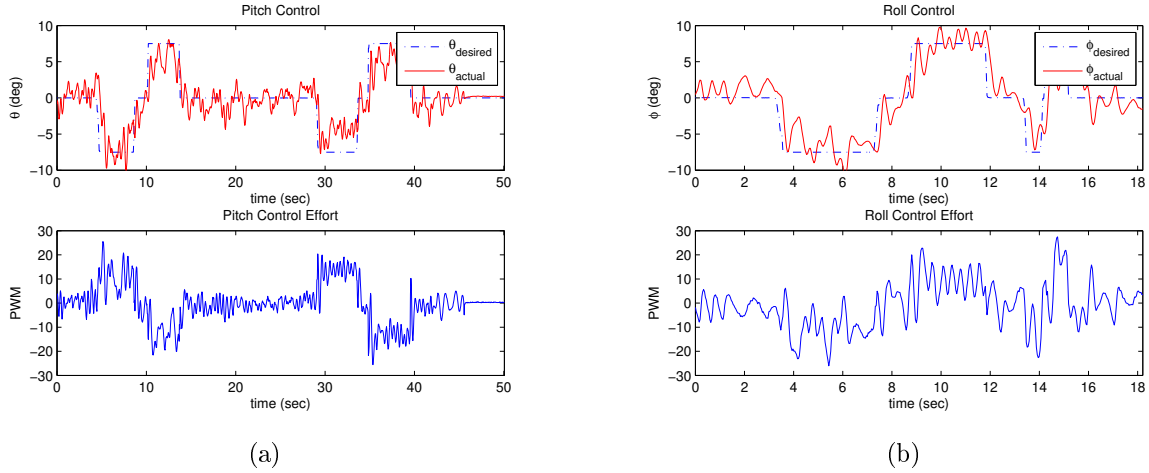


Figure 2.4: **Pitch and Roll Angle Tracking:** While restraining the quadrotor to move about each axis independently, attitude angles were successfully tracked using PD control. Subfigure (a) demonstrates pitch control and the associated PWM effort while (b) shows roll control and effort.

After gaining confidence in attitude angle tracking, outer loops were added to control the quadrotor's position. The flight results in Figure 2.5 demonstrate adequate attitude tracking to maintain hover on the quadrotor vehicle using the previously mentioned PD loop structure.

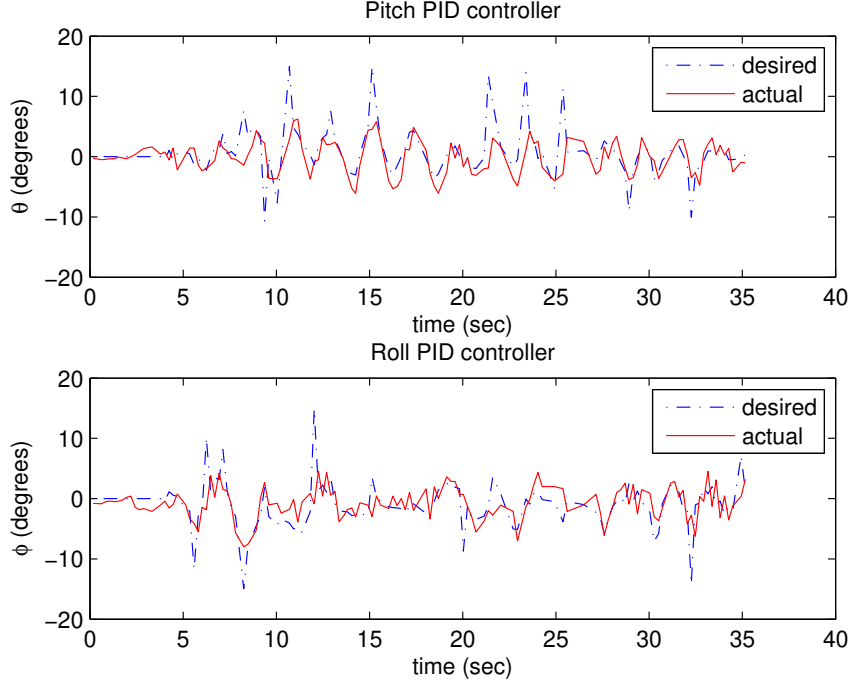


Figure 2.5: **Attitude Angle Tracking:** Pitch and roll tracking is demonstrated while in a hover condition. Desired pitch and roll angles are generated by position control in an outer loop as discussed in Section 2.5.3. Attitude tracking is uncoupled and adequate to maintain position.

### 2.5.3 Position and Velocity Control

Position and velocity control were implemented as successive loops around the inner attitude loops. It is assumed that the velocity dynamics are significantly slower than the attitude dynamics, so the inner attitude loops are treated as a block with unity gain and desired attitude angles are computed as the output from the velocity loop. Similarly, the desired velocity commands are generated by calculating control effort from position error. This creates a three tiered successive loop system where desired positions are the input, as shown in Figure 2.6.

The velocity and position estimates returned from the optic flow sensor are surprisingly robust and drift very little. Control based on these position and velocity estimates produces a stable hover. This was demonstrated in several hovering experiments. Sustained hover in a tight circle was maintained over a period of seven minutes without any repositioning by a human pilot. Figure 2.7 shows optic flow

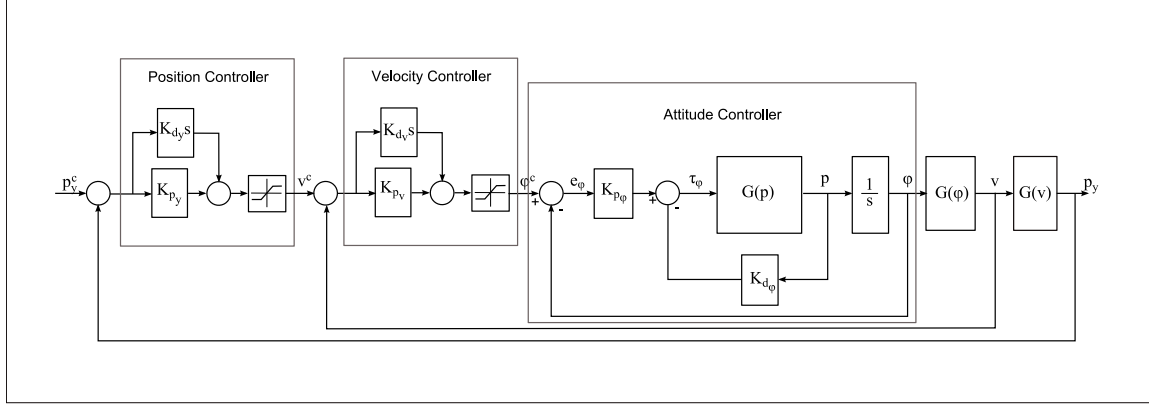


Figure 2.6: **Position and Velocity Control Loops:** Successive loops allow control gains to be tuned independently in a cascade fashion. This block diagram illustrates the successive loops used to control position with velocity damping.

position readings over a flight of approximately 75 seconds in duration. Though the figure indicates less than 0.5 meters of position drift over the entire flight, video of the flight indicates some accumulated error in the position estimate returned by the optic flow sensor. However, the true position drift is estimated to be less than one meter of the course of the flight.

#### 2.5.4 Altitude Control

Altitude was the most challenging of the controllers to implement successfully due to coupling with attitude angles, unmodeled behavior due to battery depletion, and ground effect. First, pitch and roll motions away from level resulted in substantial loss of lift that contributed to oscillations in altitude. Limits had to be imposed on the altitude controller to prevent serious oscillations. Second, as batteries were depleted, the motor response changed which called for changes in the altitude limits. Finally, ground effect, an aerodynamic effect where the wash from the propellers causes extra buoyancy, made altitude a nonlinear function of thrust near ground level. The optic flow sensor performs best close to the ground, and therefore, when it was in use, the vehicle was close to or in ground effect. In addition, height readings close to the ground were inaccurate when using the ultrasound sensor which cannot range less than 0.15 meters. All of these effects contributed to difficulties in altitude control

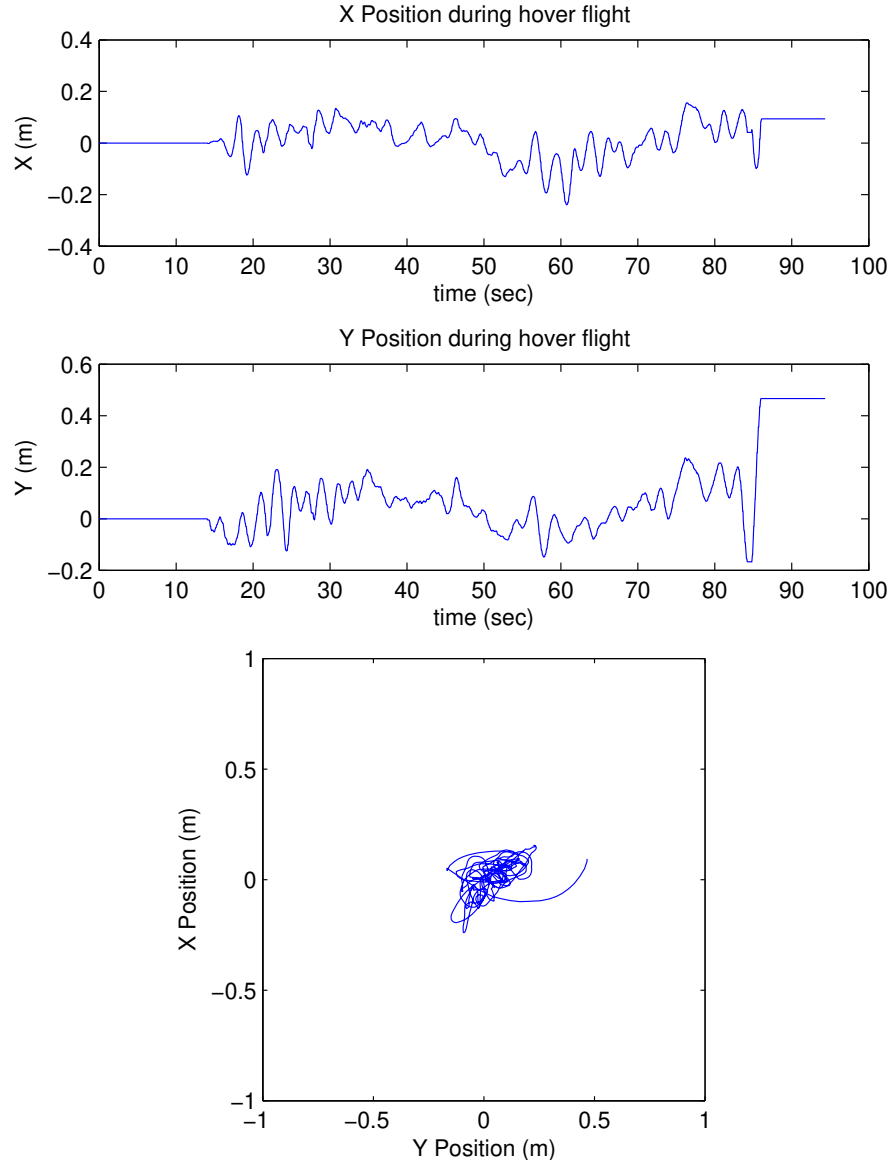


Figure 2.7: **Position Hold:** A sustained flight of over 70 seconds in duration was completed without manual piloting. This demonstrates the ability of the helicopter to hover well provided a good position/velocity measurement. The desired position is always zero meters and has not been pictured.

exhibited by a steady-state offset of 0.125 meters as shown in Figure 2.8. Good results were obtained by limiting control effort, adding integral control, and fine-tuning the altitude controller.

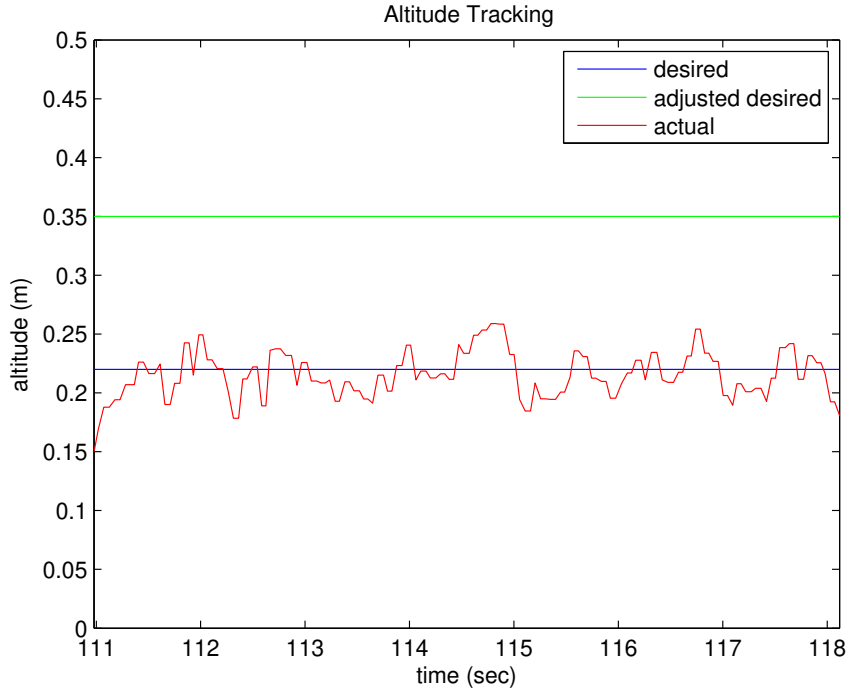


Figure 2.8: **Altitude Tracking:** A steady-state offset exists on altitude making the actual value slightly below the desired value. This is not ideal, but flight was maintained at a desired altitude by specifying a height slightly greater than the actual desired height.

## 2.6 Summary

In this chapter, much of the related work regarding helicopter control was discussed and compared to the results shown herein. Dynamic vehicle models describing how state variables relate to inputs, forces, and torques were developed for use in simulation and control. Finally, the implemented control system composed of PD control for heading, attitude, position, and altitude control were described, and real-world results on a hovering quadrotor were exhibited.

## Chapter 3

### Sensors and Estimation

The control loops mentioned in Chapter 2 rely on fast, reliable estimates of the pose (attitude and position) and velocity states of the vehicle. A six-degree-of-freedom system presents several estimation issues which deserve discussion. Attitude estimation and rotation representation are areas of discussion and research on their own. Position estimation then adds additional complexity. To tackle these problems, a sensor array has been selected and estimation schemes developed. This chapter first describes the physical hardware including the onboard computing capabilities and the full sensor suite on the vehicle. Models are derived for each sensor and estimation schemes are shown. Kalman filtering provides a framework for fusing attitude and position estimates from various sensors. Time and measurement update equations used in onboard Kalman filters are presented with results from the hovering aircraft.

#### 3.1 Onboard Computing

Two onboard computers are used to interface the sensors together and provide low-level control. The Kestrel autopilot acts as a small inertial measurement unit (IMU) housing the three-axis accelerometers and gyros. It also contains interfaces for analog input and SPI communication which are used for the ultrasound and optic flow sensor respectively. To handle vision processing, a great deal of additional computation power is needed. Two solutions have been proposed for this: the Gumstix Verdex Processor [25], and the VIA EPIA PX Pico-ITX small computer [26].

### 3.1.1 Kestrel Autopilot

The Kestrel autopilot, pictured in Figure 3.1, contains an array of sensors intended for autonomous flight of fixed-wing UAVs which include three-axis accelerometers, three-axis rate gyros, barometric altitude sensor, and differential airspeed sensor. The barometric and differential pressure sensors are considerably less accurate at small changes of altitude and low airspeeds (less than five meters per second) and therefore are not used on the quadrotor. This autopilot is marketed by Procerus Technologies [27] and provides the basis for hover flight on the coaxial and quadrotor helicopter platforms discussed in this thesis. In summary, the autopilot functions as the heart of the estimation and control algorithms implemented on the quadrotor. The attitude estimation, servo output, and ground communication are all interfaced through the autopilot. The groundstation runs a program called Virtual Cockpit that communicates to the autopilot using a spread spectrum serial modem. This groundstation application has been adapted to fit the needs of the quadrotor system. The UAV autopilot and communication and simulation software used in this thesis are based on previous systems discussed further in [28].

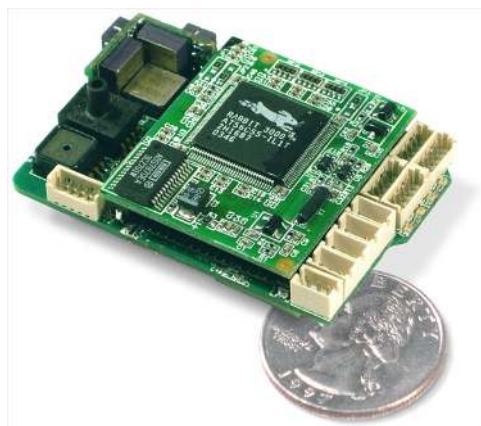


Figure 3.1: **Kestrel Autopilot:** The autopilot interfaces with all sensors, computes estimates of attitude and position, and send PWM control signals to the four motors.

### 3.1.2 Onboard Vision Processing

While the Kestrel autopilot interfaces with sensors and provides low-level estimation and control, additional processor power is desirable to process video onboard. Two onboard computers were used at different stages in this research which enabled onboard video processing for the helicopter: the Gumstix Verdex Processor and the VIA EPIA PX Pico-ITX small form factor computer.

***Gumstix Processor*** The Gumstix line of processors are famous for their small size, approximately the size of a stick of gum, weighing just 2.5 oz. (see Figure 3.2), and their impressive processor speeds. The Verdex processor, for example, runs at 600 MHz and is based on Intel XScale technology. The processor boards come with up to 128 MB RAM and 128 MB of flash memory which acts as a solid-state hard drive. Many daughter boards can be attached to either of the two interface ports on the processor board giving such capabilities as serial communication, PWM output, wired and wireless ethernet, microSD card storage, and USB On-The-Go support.



Figure 3.2: **Gumstix Processor:** The Gumstix processor is a very small Linux computer. Weighing just 2.5 oz., it provides enough processing power to perform moderate computer vision calculations.

Gumstix processors natively run a custom build of the Linux operating system, making them rather simple to navigate and use. Software is easily cross-compiled

using an ARM-Linux cross-compiler making software development fairly rapid. Image processing is possible on the Gumstix using libraries such as OpenCV [29] and EmbedCV [30], and images are easily captured using a standard web camera. Communication with other devices is accomplished through serial and ethernet interfaces.

***Pico-ITX Small Computer*** Though the Gumstix wins hands-down when it comes to size, the Pico-ITX provides faster computational speeds for image processing and is still a very small computer. Incorporating a 1.0 GHz VIA C7 Processor and containing all the standard peripherals of a laptop computer (including wired ethernet, VGA monitor support, USB, and PS2 mouse and keyboard), the Pico-ITX provides a familiar working environment. Therefore, the Pico-ITX was selected as the processor of choice for developing onboard computer vision applications. In this research, a solid-state hard-drive was attached to the SATA connector on the main board providing 4 GB of storage space and Ubuntu Linux was installed to provide a stable operating system. In order to provide wireless connectivity while in flight, a VNT6656G6A40 54 Mbps wireless USB module [31] was installed.

Two methods of remotely viewing applications are possible. SSH tunneling with X-Windows forwarding provides a simple method of running remote applications, but the connection is rather slow. NoMachine [32], a desktop virtualization company, provides much faster remote desktop access to the Pico-ITX computer than X-Windows forwarding. Their server, NxServer was therefore installed on the quadrotor and a client application runs on the ground station. Thus, there are two wireless ground links: the autopilot modem connection (rather slow at a baud rate of 115200) and the digital video link implemented through the Pico-ITX. The Pico-ITX is configured to start an ad-hoc network upon boot-up to which the ground station can connect.

The major convenience of working with the Pico-ITX is the ability to remotely compile and run programs and get real-time visual feedback from the remote system. Figure 3.4 shows a screenshot of a ground station computer connected wirelessly to the Pico-ITX onboard the quadrotor. Using this remote interface greatly simplifies and

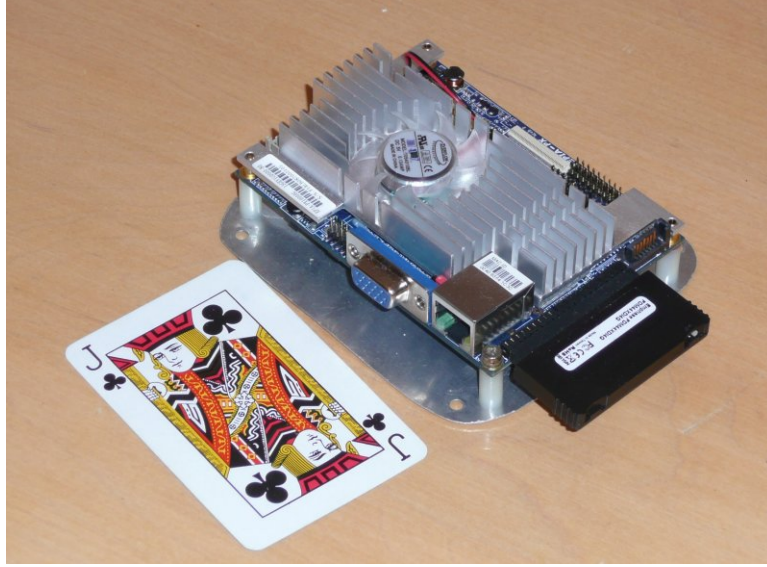


Figure 3.3: **Pico-ITX Processor Board:** The Pico-ITX form factor computer provides a substantial amount of processing power in a very small size. Running Ubuntu Linux, it is capable of running computer vision algorithms in real-time and of interfacing to remote computers over wireless remote desktop connections.

improves the software development system for the quadrotor. Programs are written using standard C and C++ and compiled onboard. Then, streams of processed video are viewed in real-time on the groundstation. Such a system has been invaluable in moving forward with computer vision research on the quadrotor platform.

### 3.2 Sensors

The quadrotor helicopter used in this research is outfitted with several sensors to estimate the essential states necessary to control and maneuver the aircraft. The full sensor suite includes:

1. A triad of MEMS rate gyros oriented along each axis to measure body-frame angular rates  $p$ ,  $q$ , and  $r$ ,
2. A triad of MEMS accelerometers which measure specific forces along the axes of the vehicle body frame,
3. An ultrasound sensor positioned downward facing to measure height above ground (HAG),

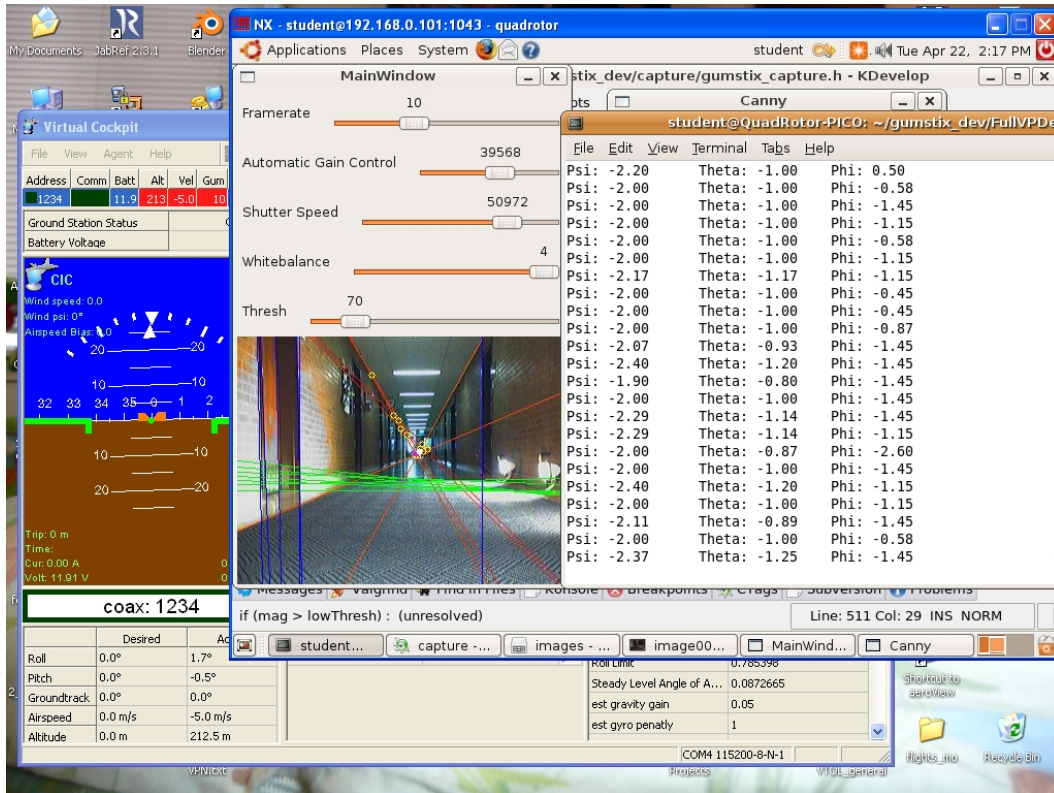


Figure 3.4: **Remote Desktop Over Wireless:** A PC running Windows XP views a computer vision application run on a Ubuntu Linux PC on the quadrotor in real-time.

4. An optic flow sensor that measures pixel flow across an optical sensor much like an optical computer mouse, and
5. A standard web camera (Logitech QuickCam Pro 4000) providing video frames at a maximum rate of 30 frames per second.

Operating in a GPS-denied environment requires specialized sensors to measure position and height above ground (HAG). The ultrasound range sensor measures HAG while the OFS, previously used for such applications as height-above-ground sensing and wind estimation, is used to determine displacements and velocities in the horizontal plane. Using the downward-facing ultrasound and optic flow sensor together, accurate position measurements are obtained with very little drift due to integration. Thus, hover flight and position tracking can be accomplished without external position sensing.

Specifications for each of the computing systems will be mentioned in the following sections. Sensor models will then be presented for each sensor or set of sensors except for the video camera which will be discussed in Chapter 4. Finally, estimation techniques based on these sensor models will be presented.

### 3.2.1 Ultrasound

Ultrasonic transducers provide absolute measures of position for feedback control purposes. A single downward facing sonar sensor is used to provide a measurement of height above ground. The sensors on the helicopter are made by Maxbotix (pictured in Figure 3.5) and have noise characteristics which are important to model.

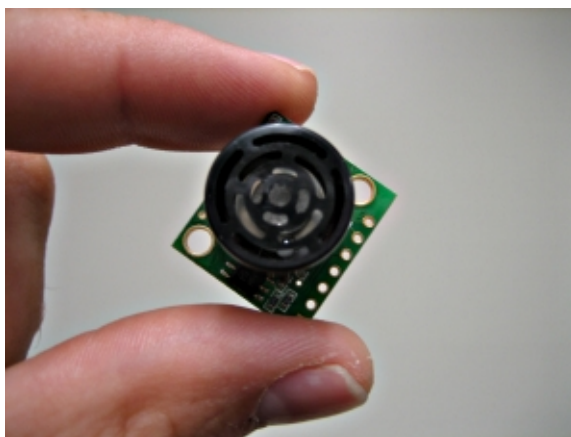


Figure 3.5: **Maxbotix Ultrasound Sensor:** Ultrasound sensing provides fairly reliable height estimates and can additionally be used as a range sensor for obstacle or wall detection.

The sensors have a range of approximately seven meters. Readings are also constrained to be above 0.15 meters (6 inches). This results in a deadband at the base of the sensor measurement which can lead to problems with maintaining heights close to the ground. Ultrasound sensor readings occur at approximately 20 Hz. Using an analog-to-digital converter on the autopilot, it is easy to collect the readings from the sensor. Readings from the ultrasound sensor are fairly reliable, but noise does occur. Two filtering techniques were used to improve the signal and reject outliers.

First, a slew filter prevents the sensor from trusting spurious readings by preventing large jumps. Then an alpha filter is applied to smooth adjacent measurements.

### 3.2.2 Optic Flow Sensor

The optic flow sensor is based on a small integrated circuit common to optical mice used for computer input. The sensor used in this research is based on the Avago ADNS 3080 optic chip [33] shown in Figure 3.6. It contains a  $30 \times 30$  CMOS pixel array which is sampled at 6400 frames per second. The output of the sensor is a number of pixels that flows across the imaging array in the  $x$ - and  $y$ -direction which will be called  $\gamma_x$  and  $\gamma_y$ .

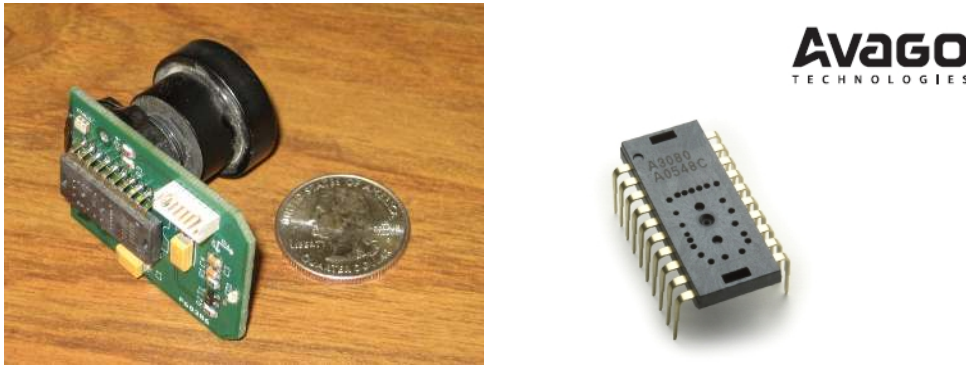


Figure 3.6: **Optic Flow Sensor:** A custom-made optic flow sensor, pictured at left, has been developed in the MAGICC Lab. It is based on the Avago ADNS 3080 optical mouse chip pictured at right.

Optic flow readings operate best at a light intensity of at least  $80 \text{ mW/m}^2$  and at a wavelength between 600 and 700 nm (which is in the red portion of the light spectrum). This leads to difficulties with using such sensors in fluorescent light, which contains very little of this portion of the spectrum. For this reason, early tests of the system were performed in the presence of bright flood lights which contained adequate red light. The sensor also responds best at a nominal height corresponding to the nominal focal length of the sensor. A variety of lenses can be fitted to the sensor to allow for different situations and imaging needs. A lens with a narrow field of view is

often advantageous because it allows frequent measurements to be taken and allows the sensor to detect motion over surfaces with fine textures such as carpeting or tile.

Optic flow sensors have found applications in several different UAV-related projects. Griffiths [34] utilized an optic flow sensor based on an optical mouse chip on a small UAV to avoid obstacles. In this application, the optic flow sensor was used to measure distance rather than velocities. The velocity readings from an airspeed sensor and a GPS were combined with the pixel count reading to measure height-above-ground or distance to a canyon wall. A downward-facing sensor was combined with two laterally-oriented sensors (one out each side) to estimate flow on opposing walls and guide the vehicle safely through a canyon by maintaining equal optical flow on each side.

In Barrows and Neely [35], compact, lightweight optic flow sensors were built to provide visual sensing for micro air vehicles. These mixed-signal VLSI sensors demonstrated the ability to use the optic flow sensor to avoid collisions with walls.

At low altitudes, the optic flow sensor works well as a HAG sensor and has been used as an aid in precision landing [36]. The optic flow sensor was used as a ranging sensor by comparing the flow of features detected by the imaging array to the known ground-speed reported by a GPS receiver. The optic flow sensor can thus be utilized to compensate for drift in barometric sensors or to allow for varying take-off and landing ground heights without providing a complete terrain map.

Rodriguez, et al. [37] have used the ratio of longitudinal to lateral optical flow to estimate the crab angle of a flying vehicle. From this measurement, combined with ground track from the GPS and the airspeed, they have shown a method to compute windspeed without loitering in place or relying on magnetometers.

Evidence has demonstrated that flying insects such as flies and bees use optical flow in their visual perception to control their speed and height. In [38], the authors simulate the behavior of bees using an optic flow reading. Similarly, in [39], efforts are made to mimic the flight behavior of flies to control very small indoor flying vehicles.

The optic flow sensor has been established as an effective sensor for estimating velocity, and can be used as a range sensor when velocity measurements are already

provided. Differential readings can be used to maintain distance between walls or other surfaces. In this work, the optic flow sensor is used as a velocity and position indicator and under the right conditions, it provides very reliable readings which greatly improve the hover flight behavior of helicopters.

**Optic Flow Sensor Model** Figure 3.7 shows a diagram of the geometry underlying the optic flow sensor. The upper triangle is the imaging frustrum of the optical lens. The quantity  $FOV$  is the lens field of view expressed in radians. The quantity  $\Delta\phi$ , also in radians, represents an angular change through which the vehicle has rotated. On the focal plane, the quantity  $p_n$  represents the total pixel count of the sensor in each dimension. The sensor in use is symmetric with  $p_n = 30$ . The quantity  $\Delta x_m$  represents the distance swept out on the ground in meters, and  $h$  denotes the height above ground.

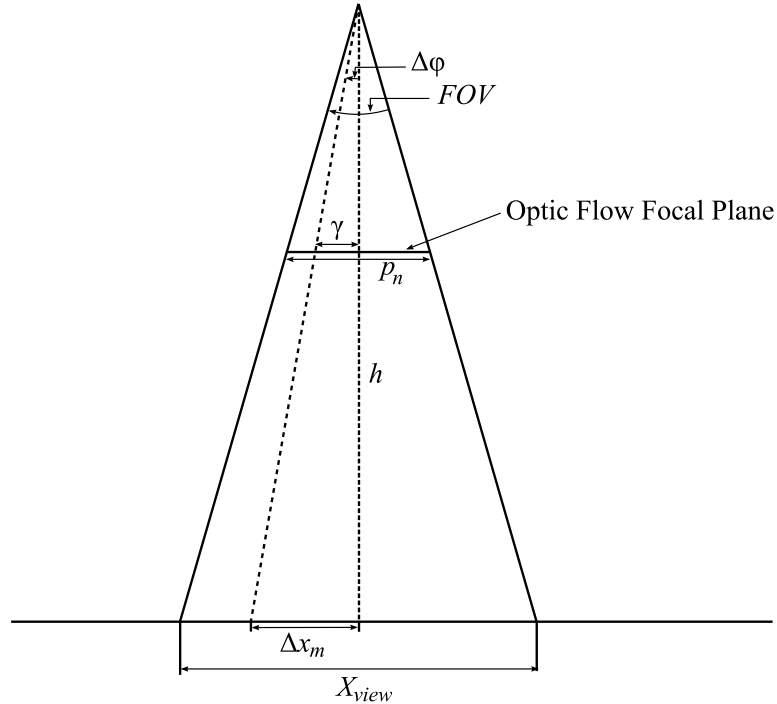


Figure 3.7: **Optic Flow Sensor Geometry:** The upper triangle is an enlarged version of the viewing frustrum of the optic flow lens. The quantities  $\phi$  and  $FOV$  are angles; the quantities  $\gamma$  and  $p_n$  are in pixels; and the quantities  $h$ ,  $X_{view}$ , and  $\Delta x_m$  are in meters.

The optic flow sensor reads the combination of two quantities: the pixel change due to actual translation and the pixel change due to a small change in angle,  $\Delta\phi$ . Using assumptions proposed in [37] and using similar triangles an equation relating the total pixel reading  $\gamma$  and  $\Delta x_m$  can be written as

$$\frac{\Delta x_m}{X_{view}} = \frac{\gamma}{p_n} \quad (3.1)$$

where

$$X_{view} = 2h \tan\left(\frac{FOV}{2}\right) \approx h \cdot FOV. \quad (3.2)$$

A similar equation can be computed for the correlation between  $\gamma$  and  $\Delta\phi$  which results in a model for the total pixel reading,  $\gamma$ , due to translation and rotation:

$$\gamma = \frac{p_n}{FOV} \left( \frac{\Delta x_m}{h} + \Delta\phi \right). \quad (3.3)$$

An optic flow pixel reading  $\gamma$  thus comes from both angular and linear motion. If an estimate of rigid-body rotation is available, the translational motion of the sensor can be detected as discussed in the next section.

### ***Velocity Estimation and Position Dead Reckoning from Optic Flow Sensor***

Inverting the optical flow sensor model given in Equation 3.3, we obtain equations for determining change in position as a function of the  $\gamma_x$  and  $\gamma_y$  pixel readings from the sensor:

$$\Delta x_m = h \left( \frac{FOV}{p_n} \gamma_x - \Delta\theta \right) \quad (3.4)$$

$$\Delta y_m = h \left( \frac{FOV}{p_n} \gamma_y + \Delta\phi \right) \quad (3.5)$$

where the signs in front of the last term are selected according to angular sign conventions.  $\Delta\theta$  and  $\Delta\phi$  are termed the pitch- and roll-effect and are easily subtracted out of the pixel measurements to provide estimates of position and velocity.

A simple lab test demonstrates the effectiveness of the optic flow sensor velocity readings in Figure 3.8. The helicopter, outfitted with a downward facing ultrasound sensor and an optic flow sensor, was subjected to perturbations in attitude angles and position. Plot (a) shows the height reading from the ultrasound sensor. The first half of the test was performed at a low height of approximately 0.5 meters (shown in segments 1-4). Then, the helicopter was raised to 0.75 meters and the test was repeated (shown in segments 5-8). The surface quality, shown in (b), changes depending on the height above the imaged surface. Though this can be adjusted by focusing the optic flow lens, no focusing was performed during the experiment. Surface quality (denoted *squal*) readings above 40 are typically reliable. The optic flow sensor performs best at low heights (less than approximately 1.5 meters in the current lens configuration).

The test was divided into four main time segments: pitching,  $x$ -translation, rolling, and  $y$ -translation. These sections are delimited by number in the lower six plots of 3.8. The test was performed by first swinging the helicopter around the the body-frame  $y$ -axis in a pitching motion (Segment 1 and 5), then moving the vehicle forward and backward quickly in the body-frame  $x$ -direction (Segments 2 and 6). Then, the vehicle was rolled about the body-frame  $x$ -axis (Segments 3 and 7) and subsequently translated in the body-frame  $y$ -direction (Segments 4 and 8). The pitching motion results in pixel movement across the optic flow sensor which must be subtracted out to return a good position or velocity reading. The angular motion and translation both register pixel movement as shown in plots (e) and (f), but the angular motion is effectively removed in plots (g) and (h). Thus, the optic flow sensor is effective at detecting changes in position when angular motion is correctly removed.

Though absolute position cannot be determined from the optic flow sensor, by adding up the incremental changes in  $x$ - and  $y$ - position, translation relative to a starting location can be estimated through dead reckoning. In testing, position feedback from the optic flow sensor allowed for very effective hovering flights in which the autopilot was able to hover autonomously in a one-meter radius circle for nearly the entire life of the battery. A portion of these results is shown in Figure 2.6. In

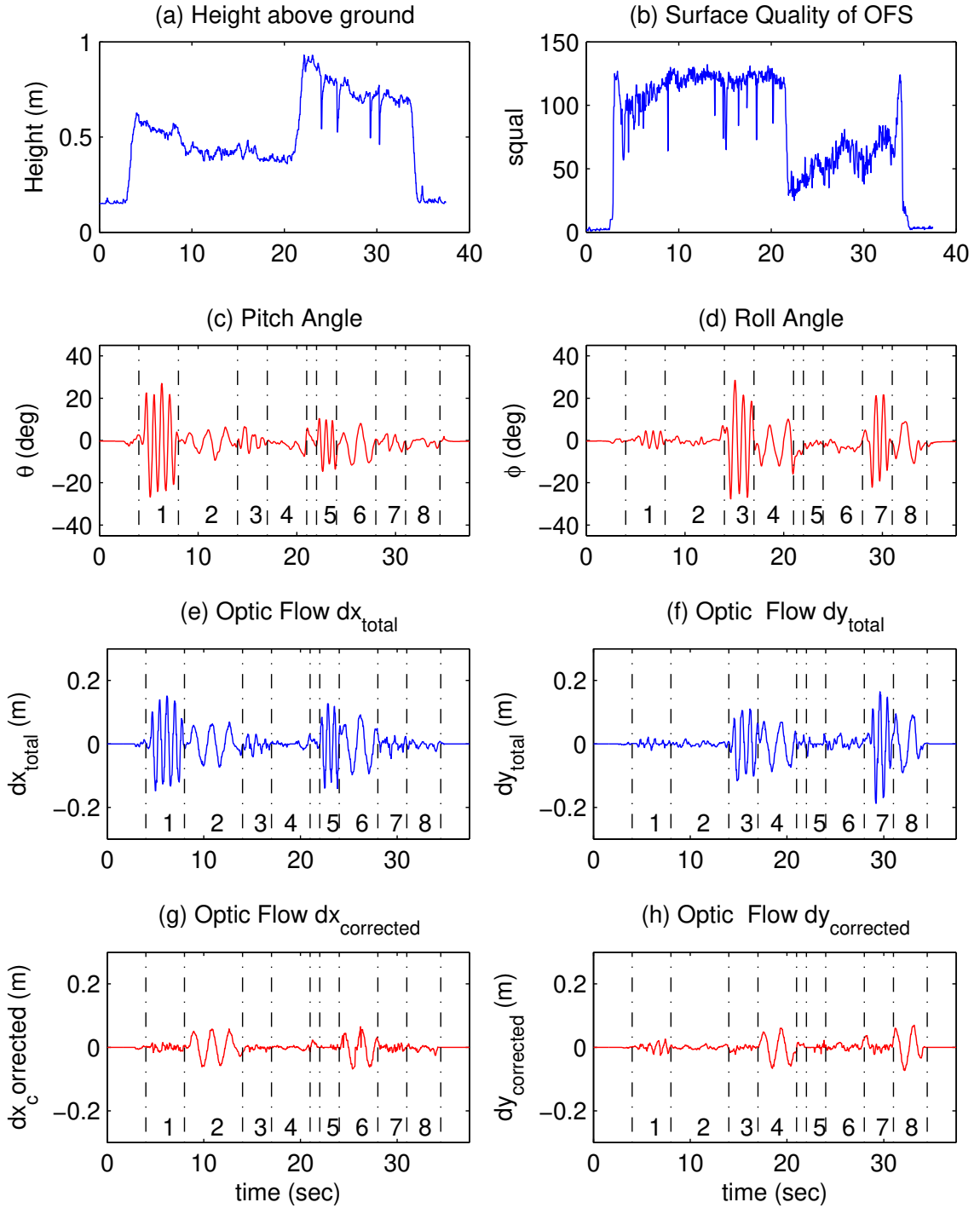


Figure 3.8: **Optic Flow Sensor Lab Test Results:** A lab test demonstrates the effectiveness of velocity readings from an optic flow sensor. Pixel readings due to the pitch and roll in plots (c) and (d) are effectively removed from the optic flow reading resulting in the reliable differential position measurements in (g) and (h).

effect, the optic flow sensor provides reliable position estimates which are in some ways superior to GPS readings. For example, GPS readings provide absolute position estimates relying on the precise timing of orbiting satellites. Typical GPS receivers can receive position estimates at rates of 1-4 Hz (though more expensive units provide slightly higher rates). At these relatively slow rates, instantaneous velocity estimates can have significant error. In addition, GPS receivers typically yield unreliable velocity estimates at vehicle speeds less than five meters per second. In general, hover flight takes place at low velocities, so the optic flow sensor's ability to return fast, accurate velocity updates at rates between 20 and 50 Hz makes it superior for velocity estimation.

Despite their high accuracy and capability, optic flow sensors have several drawbacks that must be recognized. First, the imaged surface must be of a certain quality for the sensor to properly detect pixel movement. An ideal surface is similar to a mousepad: plenty of surface texture near the resolution of the lens. Strongly textured carpets work best while shiny surfaces do not. Lighting conditions are also very important. As mentioned earlier, the sensor responds best in the range of 600-700 nm which means that they work poorly in fluorescent light and poorly-lit rooms. In initial flight experiments in this work, flood lights were used to light the hallway. Then, to improve the system, a bright flashlight bulb was added to the underside of the quadrotor to provide adequate lighting for the sensor. This spotlight gave surface quality readings equivalent to those provided by the flood lights. In addition, the optic flow sensor operates best near a certain height which depends on the focal length or field-of-view of the lens. In these experiments, the optimal range was between 0.1 and 2 meters. Possible improvements to the sensor will be discussed in the future work at the end of this thesis. It is clear that under the right conditions, the optic flow sensor performs very well.

### **3.2.3 Accelerometers and Rate Gyros**

The rate gyros are key components in attitude estimation because attitude estimates can be found by simply integrating the Euler angle differential equation

given in Equation 2.3. The signal from MEMS rate gyros contains noise which leads to significant drift when integrated for attitude measurements.

To bound the drift due to integration, vector measurements of attitude are needed. Typical sensors which provide vector measurements are three-axis accelerometers and three-axis magnetometers. However, both of these types of sensors, in the size suited to small indoor aircraft, also suffer from accuracy issues. Though these sensors do not drift as does the integrated signal from rate gyros, they exhibit behaviors which make them less suitable to provide adequate attitude estimates on hovering aircraft.

***Accelerometer Problems Near Hover*** Accelerometers consist of a proof-mass suspended in a capsule. They can be used to estimate the orientation of the gravity vector, but this measurement is mixed with a reading of body-frame accelerations making the estimate ill-posed in some conditions. Studying a free-body diagram of an accelerometer on a hovering aircraft demonstrates one of the primary difficulties with accelerometers in a hovering condition.

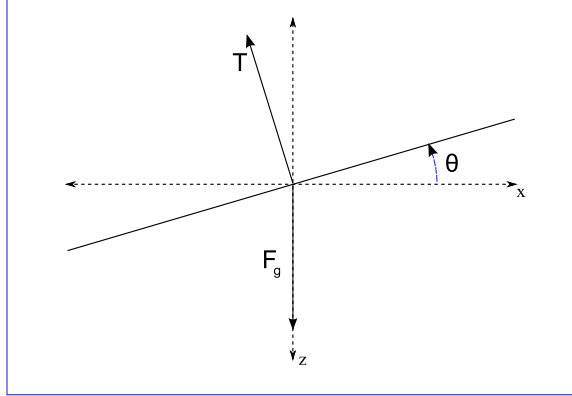


Figure 3.9: **Quadrotor Free Body Diagram:** Free body diagram of a hovering vehicle at an angle  $\theta$  with respect to horizontal.

It is assumed that the vehicle provides a thrust which is enough to compensate for the force of gravity and for the angle  $\theta$ . Thus,

$$Mg = T \cos \theta, \quad (3.6)$$

where  $M$  is the mass of the vehicle,  $T$  is the thrust force, oriented perpendicular to the vehicle, and  $g$  is the gravitational constant. The pitch angle  $\theta$  causes the vehicle to accelerate in the inertial  $x$ -direction in response to a force  $T \sin \theta$  with an acceleration  $a_x = \frac{T \sin \theta}{M}$ .

Now, we examine the forces on a proof mass representing an accelerometer oriented along the body-frame  $x$ -axis of the vehicle, illustrating that the  $x$ - and  $y$ -axis accelerometers will read close to zero on a hovering vehicle. The measurement of the accelerometer is the sum of accelerations induced by forces acting a proof mass as illustrated in Figure 3.10. Assuming that aerodynamic forces from the propellers and body are negligible near hover and summing forces in the body-frame  $x$ -direction we see that

$$\Sigma F_x^b = -mg \sin \theta + \frac{mT \sin \theta \cos \theta}{M}. \quad (3.7)$$

Using Equation 3.6, we see that the sum of forces in the body frame  $x$ -direction would be identically zero, meaning that any attempt to use the accelerometers as a vector measurement of gravity would give erroneous values near hover.

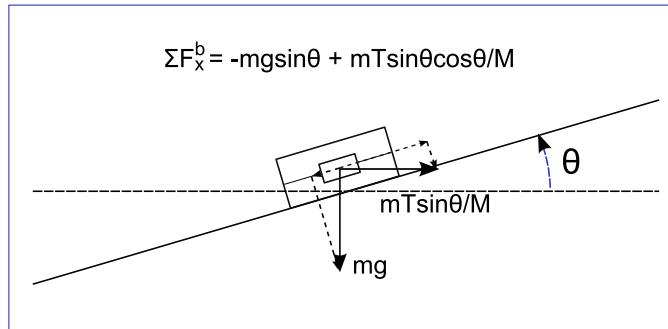


Figure 3.10: **Accelerometer as a Proof Mass:** The summation of forces acting on a proof mass in the body frame of a hovering vehicle illustrates the difficulty of measuring the gravity vector using a three-axis set of accelerometers.

**Magnetometer Problems** Magnetometers measure the magnetic field strength in a particular direction. When three such sensors are mounted in orthogonal directions on a vehicle, the magnetic field vector can be estimated. Since this magnetic field

vector is a function of geographic location, look-up tables provide the known declination and inclination values for the true vector orientation. A significant problem with magnetometers is the fact that they measure magnetic variations, which in an indoor setting can vary greatly due to disturbances from metal beams in ceilings and walls. In addition, nearby system components such as motors and electrical wiring can cause significant electromagnetic interference. If such disturbances are constant, they can be accounted for through calibration, but buildings often make magnetometers behave erratically, corrupting measurements. Better sensors, then, are needed to provide an unbounded correction to rate gyros in an indoor setting.

### 3.3 Attitude Estimation and Data Fusion

Kalman filtering provides the primary framework for fusing measurements on the hovering aircraft. Attitude is estimated using a Kalman filter which fuses rate gyro data and accelerometer measurements when the vehicle is stationary. Unfortunately, when hovering, accelerometer measurements are typically unreliable, and a separate, error-bounded measurement is sought from computer vision. This section discusses methods for estimating the attitude of the vehicle and proposes a fusion method for incorporating vision for bounding attitude error.

The filter follows the continuous-discrete implementation of the extended Kalman filter (see Lewis, pg. 263 [40]) and is split into a time update, computed at a specified rate, and a measurement update, computed whenever sensor readings are taken. The continuous portion needs to be computed at a fast sampling rate to accurately integrate the continuous differential equations for all states. The discrete portion allows each sensor measurement (observation) to be computed independently and then to influence the estimates of all other interrelated states.

#### 3.3.1 Kalman Filtering Basics

The Kalman Filter estimates the mean and covariance of a set of states,  $\mathbf{x}$ , of size  $n \times 1$  where the mean estimate is notated  $\hat{\mathbf{x}}$  and the covariance is an  $n \times n$

matrix  $\mathbf{P}$ . The Extended Kalman Filter relies on integrating a model of the system of the form

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \mathbf{u}) + \eta, \quad (3.8)$$

$$\mathbf{y} = \mathbf{c}(\mathbf{x}, \mathbf{u}) + \xi, \quad (3.9)$$

where  $\dot{\mathbf{x}}$  indicates the time derivative of  $\mathbf{x}$ ,  $\mathbf{u}$  is an array of inputs,  $\mathbf{y}$  is an observation, and the noise terms  $\eta$  and  $\xi$  represent process noise and measurement noise, respectively. Thus, we have some knowledge of how the states change with time and can express them as a function of the states and inputs. We also have a function  $\mathbf{c}(\mathbf{x}, \mathbf{u})$  which relates our observations (sensor readings) with the states and we have some idea of the amount of noise on our model and our sensors.

**Time Update** In its original form, the Kalman filter (KF) relies on  $\dot{\mathbf{x}}$  being a linear function of the states and inputs in the form  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ , where  $\mathbf{A}$  and  $\mathbf{B}$  are linear operators (matrices). The Kalman Filter is proven to be the optimal filter for minimizing the squared error if noise on both the model and observation is Gaussian with zero mean. We express this as

$$\eta \sim (0, \mathbf{Q}) \quad (3.10)$$

$$\xi \sim (0, \mathbf{R}) \quad (3.11)$$

where  $\sim (0, \mathbf{S})$  means that a variable is normally distributed with zero mean and covariance  $\mathbf{S}$ .

The extended Kalman filter (EKF) “extends” the regular KF by allowing the state equations to be nonlinear in form. An approximation is made when propagating the filter by replacing the state transition matrix  $\mathbf{A}$  with the Jacobian of  $\mathbf{a}(\mathbf{x}, \mathbf{u})$ . The time update portion of the filter integrates the equations of motion from Equation 3.8 and estimates the covariance matrix  $\mathbf{P}$  according to

$$\mathbf{P}^+ = \mathbf{P}^- + \Delta t \cdot (\mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^T + \mathbf{Q}), \quad (3.12)$$

where  $\mathbf{P}^-$  and  $\mathbf{P}^+$  indicate the covariance before and after the update, respectively, and  $\Delta t$  represents a small change in time.  $\mathbf{Q}$ , the model covariance, represents how much uncertainty is introduced on the estimate at each time step.

**Measurement Update** When a discrete measurement is taken from a sensor, we seek to fuse this measurement with the estimate from our model computed in the time update. The observation is expressed as a linear combination of the states in the matrix  $\mathbf{C}$ , and we compute the Kalman gain  $\mathbf{K}$  according to

$$\mathbf{K} = \mathbf{P}\mathbf{C}^T(\mathbf{C}\mathbf{P}\mathbf{C}^T + \mathbf{R})^{-1}. \quad (3.13)$$

The Kalman gain represents a stochastic ratio of how much we trust a given a measurement compared to how much we trust the model. In practice, we estimate the measurement covariance,  $\mathbf{R}$ , by guessing the rate of error growth due to measurements and then tune the model covariance,  $\mathbf{Q}$ , to give a relative “trust factor.” Once the Kalman gain is computed, we update the state mean and covariance according to

$$\hat{\mathbf{x}}^+ = \hat{\mathbf{x}}^- + \mathbf{K}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \quad (3.14)$$

$$\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{C})\mathbf{P}^-. \quad (3.15)$$

The Kalman filter allows a model to be propagated and corrected or fused with measurements assuming that the noise on the model and measurement is Gaussian with zero mean. Kalman filters are employed on the quadrotor to fuse rate gyro measurements with accelerometer measurements and vision measurements in order to estimate attitude and heading angles.

### 3.3.2 Attitude Kalman Filter

On a fixed-wing airplane, the attitude can be effectively estimated using a fusion of data from the rate gyros and accelerometers. However, due to problems already discussed, this method is less effective on a hovering vehicle because body-

frame forces effectively cancel gravity force readings on the accelerometers rendering them unsuitable for determining angular positions (see Section 3.2.3). However, a Kalman filter which fuses accelerometer and gyro readings is still useful for ground tests, and it has been found that the quadrotor can fly reliably without any additional attitude filtering if an outer loop involving velocity and position is provided. Thus, an attitude Kalman filter using accelerometers to bound attitude errors is still useful on the quadrotor and provides the basis for more complex filtering which incorporates visual measurements of attitude.

To develop an equation for the time update of the attitude Kalman filter, it is useful to think of the rate gyros as the model. Thus, the time update consists of integrating the reading from the rate gyros to develop an estimate of attitude. Using the measurements  $p_{gyro}$ ,  $q_{gyro}$ , and  $r_{gyro}$  and Equation 2.3, the time update of the Kalman filter is given in Algorithm 1.

---

**Algorithm 1 Kalman Filter Time Update:** The time update essentially integrates the rate gyro readings to provide an estimate of pitch and roll angles.

---

$$\begin{aligned}\hat{\mathbf{x}}^+ &= \begin{bmatrix} \hat{\phi}^+ \\ \hat{\theta}^+ \end{bmatrix} = \begin{bmatrix} \hat{\phi} \\ \hat{\theta} \end{bmatrix} + \Delta t \begin{bmatrix} p_{gyro} + q_{gyro} \sin \hat{\phi} \tan \hat{\theta} + r_{gyro} \cos \hat{\phi} \tan \hat{\theta} \\ q_{gyro} \cos \hat{\phi} - r_{gyro} \sin \hat{\phi} \end{bmatrix} \\ \mathbf{A} &= \frac{\partial \mathbf{a}(\hat{\mathbf{x}}, \mathbf{u})}{\partial \mathbf{x}} = \begin{bmatrix} q_{gyro} \cos \hat{\phi} \tan \hat{\theta} - r_{gyro} \sin \hat{\phi} \tan \hat{\theta} & \frac{1}{\cos^2 \hat{\theta}} (q_{gyro} \sin \hat{\theta} + r_{gyro} \cos \hat{\phi}) \\ -q_{gyro} \sin \hat{\phi} - r_{gyro} \cos \hat{\phi} & 0 \end{bmatrix} \\ \mathbf{P}^+ &= \mathbf{P} + \Delta t (\mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^T + \mathbf{Q})\end{aligned}$$


---

The measurement update of the filter consists of comparing the estimate predicted by the rate gyros in the time update with the reading produced by the accelerometers. The accelerometers measure the specific force in the body frame of the vehicle. The accelerometer measurement  $y_{accel}$  has three components corresponding

to the body frame coordinate axes and is expressed according to

$$\mathbf{y}_{accel} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \frac{1}{m} (\mathbf{F} - \mathbf{F}_{gravity}). \quad (3.16)$$

Explicitly, we have

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} + \begin{bmatrix} g \sin \theta \\ -g \cos \theta \sin \phi \\ -g \cos \theta \cos \phi \end{bmatrix}. \quad (3.17)$$

We do not have a method for directly measuring  $\dot{u}$ ,  $\dot{v}$ , or  $\dot{w}$ . In addition, attempts to measure  $u$  and  $v$  from the optic flow sensor could be detrimental since these measurements are coupled with rotational motion. Therefore, we assume that  $\dot{u} = \dot{v} = \dot{w} = u = v = w = 0$ . In addition, the accelerometer readings can be calibrated to be in units of  $[g]$ , so Equation 3.17 is normalized by  $g$  resulting in a simplified expression of accelerometer measurements in terms of the attitude angles  $\phi$  and  $\theta$  as follows:

$$\mathbf{y}_{accel} = \begin{bmatrix} -\sin \theta \\ -\cos \theta \sin \phi \\ -\cos \theta \cos \phi \end{bmatrix}. \quad (3.18)$$

This equation acts as  $\mathbf{c}(\mathbf{x}, \mathbf{u})$  for the Kalman filter. One final change to the filter was added for computational reasons. The computation of the Kalman gain given by Equation 3.14 would require a  $3 \times 3$  inverse to be computed at every measurement update. To reduce this computational load on the embedded processor, the measurement update is rearranged to require two  $2 \times 2$  inverses instead<sup>1</sup>. The resulting measurement update is summarized in Algorithm 2.

Lab tests were completed that exhibited the effectiveness of this attitude Kalman filter. Figure 3.11 shows attitude estimates onboard a helicopter with mo-

---

<sup>1</sup>The matrix inversion lemma can be used to demonstrate the equivalence of these two methods.

---

**Algorithm 2 Attitude Kalman Filter Measurement Update:** The measurement update consists of comparing the rate gyro estimate with the accelerometer attitude estimate using the Kalman gain  $\mathbf{K}$ .

---

$$\mathbf{C} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}} = \begin{bmatrix} 0 & \cos \theta \\ -\cos \phi \cos \theta & \sin \theta \sin \phi \\ \sin \phi \cos \theta & \sin \theta \cos \phi \end{bmatrix}$$

$$\mathbf{P} = (\mathbf{P}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})^{-1}$$

$$\mathbf{K} = \mathbf{P} \mathbf{C}^T \mathbf{R}^{-1}$$

$$\hat{\mathbf{x}}^+ = \hat{\mathbf{x}} + \mathbf{K} (y_{accel} - \mathbf{c}(\hat{\mathbf{x}}, \mathbf{u}))$$


---

tors running (to simulate the vibrations occurring in flight) using rate gyros and accelerometers independently. The test performed consisted of simply rotating the hand-held helicopter first about the  $x$ -axis and then the  $y$ -axis while running the motors. The red dashed line shows that the rate gyro reading drifts over a matter of seconds and quickly deviates from the true attitude which is zero-centered. The accelerometer attitude reading (in blue) does not drift, but is very noisy resulting in an unusable attitude estimate on its own.

However, combining the good high-frequency rate gyro signal with the non-drifting low-frequency signal provided by the accelerometers, we have an effective attitude estimate with bounded error and very little lag. The fused filter is shown in Figure 3.12. Accelerometer readings effectively bound the gyro estimate removing biases. This filter forms the base of a larger Kalman filter involving attitude estimates from computer vision. The filter is tuned to trust the accelerometers approximately 100 times less than the rate gyros. This is primarily due to noise induced by vibrations. In addition, the rate gyros are periodically calibrated to remove biases which slowly change with time.

### 3.3.3 Adding Vision Estimates to the Filter

As mentioned, this gyro/accelerometer filter provides the basis for a more complex Kalman filter. When computer vision estimates of attitude are computed,

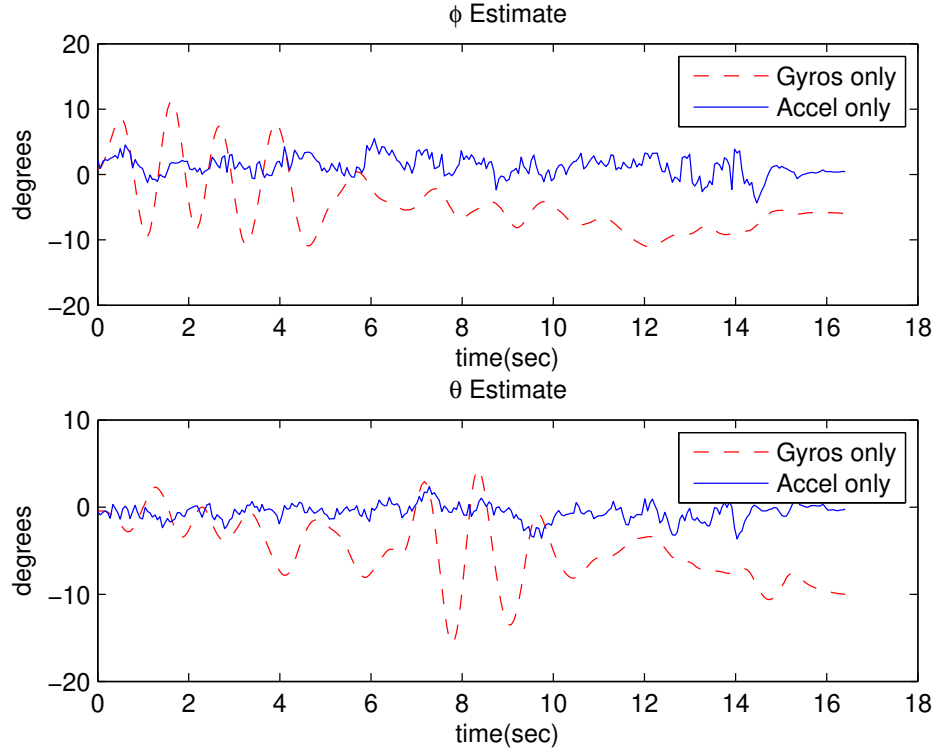


Figure 3.11: **Naive Attitude Estimation:** Naive Attitude Measurement using rate gyros or accelerometers independently.

they are passed over to the autopilot over a serial interface. When measurements arrive, measurement updates are performed to further correct the attitude readings using the visual estimate. Pitch and roll are estimated from separate computer vision algorithms making it necessary to perform two separate measurement updates. To update the estimate of  $\phi$ , the Kalman gain is computed using the variance computed in the gyro/accelerometer filter,  $p_\phi$  according to

$$K_\phi = \frac{p_\phi}{p_\phi + R_\phi}. \quad (3.19)$$

The measurement of the roll angle from computer vision is termed  $\phi_v$ , and the state and covariance are updated according to

$$\hat{\phi}^+ = \hat{\phi}^- + K_\phi(\phi_v - \hat{\phi}^-), \quad (3.20)$$

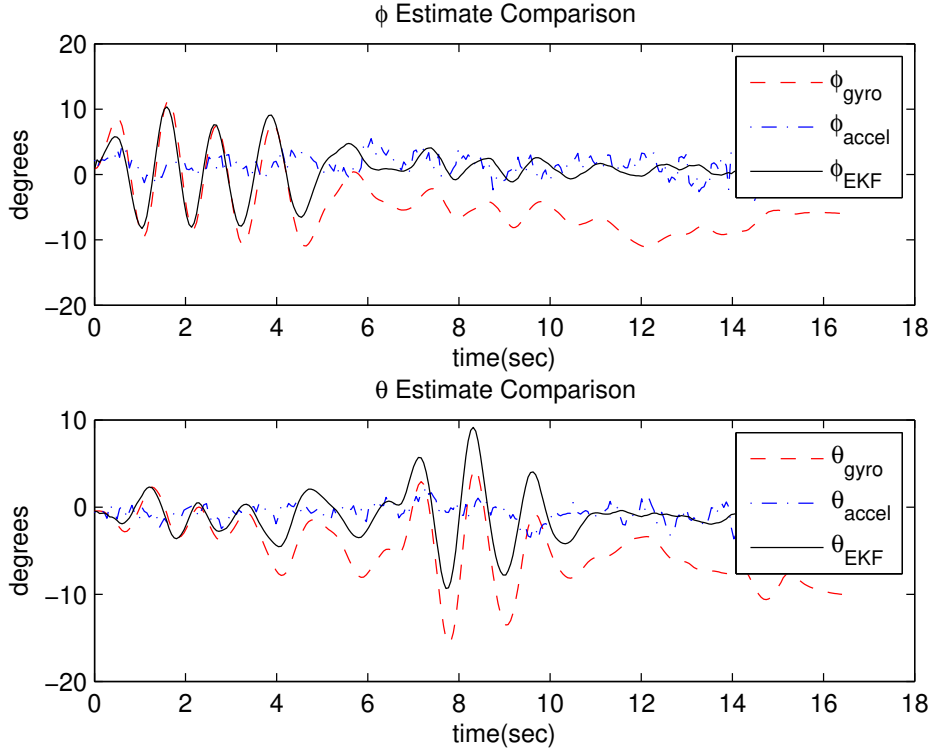


Figure 3.12: **Attitude Estimation using Data Fusion:** Kalman filtered attitude estimate using a fusion of accelerometer and rate gyro readings. The black line show the Kalman filtered estimate.

$$p_{\phi}^{+} = (1 - K_{\phi})p_{\phi}^{-}. \quad (3.21)$$

The Kalman filter allows the system to deal with unsteady rates of visual updates. Rate gyros provide adequate estimation in general, but visual updates provide corrections and bound errors. In addition, the attitude estimate from rate gyros provides a method for rejecting false visual readings, which is discussed further in Section 5.2.

### 3.4 Summary

The onboard computing abilities and sensors of the quadrotor were explained including details regarding sensor models and estimation. In particular, the optic flow sensor was explored as a means of providing reliable position and velocity estimates on a hovering vehicle. A simple method for removing the pitch and roll biases from

the optic flow reading has been demonstrated in a lab test. In addition, a Kalman filtering framework for attitude estimation has been developed and demonstrated with results from the quadrotor.



## Chapter 4

### Vision-aided Hallway Following

Computer vision is a well researched field with very promising applications in all areas of robotics. The amount of information available to humans through vision suggests that visual information be used more heavily in robotic applications. Attempting to match the pattern-recognition and visual perception capabilities of human eyesight is beyond the scope of this paper, but incorporating vision as a sensor on an instrumented aircraft is both logical and in some cases essential.

Details regarding homogeneous coordinates and Euclidean and camera projection matrices are included in Appendix B. Also included in the appendix is a demonstration of an application of perspective projection mathematics showing a model of a hallway being projected onto the imaging plane of the quadrotor’s forward-looking camera.

This chapter begins with an overview of many of the key related works in vanishing point detection and its prerequisites. The approach in this thesis starts with edge detection and line detection, both of which can be performed in various ways. Sections 4.2.1 and 4.2.2 compare the methods attempted for edge and line detection in this research and defend the final approach that uses the Canny edge detector and the Hough transform. Using the output of these preprocessing stages, two main methods of finding vanishing points are discussed and compared in Section 4.2.4. Methods for determining attitude and position from discovered vanishing points are explained in Section 4.3. Finally, results of each method are compared using onboard processing in Section 4.4.

## 4.1 Related Computer Vision Research

The goal of this chapter is to explain a method of detecting vanishing points in indoor hallways that is useful for attitude and position estimation on small air vehicles. Humans navigate using an array of sensors including tilt sensing (much like rate gyros), sound, touch, and most importantly vision. Visual cues give humans a vast amount of information which allows them to safely navigate buildings, drive cars, and fly planes and helicopters. Robotic vehicles currently tend to use specialized sensors such as accelerometers, rate gyros, sonar, laser range finders, and GPS before visual techniques are employed. This is due to the ease with which these types of sensors can be incorporated into a system. Historically, computer vision has been a difficult sensor to utilize in practice. This is due to several challenges inherent in determining patterns from digital images such as digital storage and computational needs, the difficulty of expressing perspective geometry in mathematical terms, and a sheer lack of understanding of how human visual perception works. The availability of onboard processing power for small UAVs has led to an increasing use of visual techniques in micro air vehicle (MAV) applications.

There are three basic ways to use vision for navigation: visual servoing, visual odometry, and simultaneous localization and mapping (SLAM). Visual servoing can be described as creating control laws based upon visual measurements. This technique is common in structured settings such as assembly line vision robots. The field of computer vision has been explored in excellent tutorials described in [41] and [42]. Visual odometry is a technique that uses optical flow or pose estimation to estimate position and velocity based on visual readings. Some key works in this include [43] and [44]. SLAM is a broad area of research which uses tracking of landmarks to localize one's position within a map while simultaneously building the map (see [45] and [46]). All of these methods provide means of navigating based on visual information.

In this work, a visual servoing approach is selected. Using edge detection methods to find the vanishing points of lines, MAV pose is estimated from vanishing point locations. Before introducing the proposed algorithms for vanishing point detection, related work in several subareas is discussed. This related work area is divided

into preprocessing methods, vanishing point estimation methods, three dimensional tracking methods, and mobile robot visually-aided navigation.

#### **4.1.1 Image Preprocessing**

Methods for edge detection and line detection are prerequisites for detecting vanishing points. Edge detection algorithms are well understood and implementations of many methods are included in open-source software packages (such as Intel’s OpenCV [29]). Of primary importance to this work, the Sobel convolution operator and Canny edge detector are used in many of the algorithms in this paper as pre-processor steps. Descriptions of these algorithms are mentioned in section 4.2.1, and more detailed descriptions are available in computer vision texts such as [47].

Two main line-finding techniques were explored in this thesis work. The Hough transform is a well-known method of finding patterns in an image based on a patent by Paul Hough in 1962 [48]. An accumulator is built up representing a parameterization of edge points. Peaks in the accumulator space represent lines in the image space. Other methods, like the Burns Line Detector [49], utilize the gradient direction in addition to the gradient magnitude to trace lines in a method involving connected components. Both methods were implemented and tested as part of this work.

Computer vision textbooks such as [50], [51], and [47] describe fundamentals of computer vision, special Euclidean transformations, image preprocessing, and multiple view geometry. The approaches proposed in this thesis are built on these methods. The Canny edge detector typically gives the best results for detecting hallway edges as long as proper thresholds are selected. A custom implementation of the Hough transform was developed which allows a user to observe how lines in image space show up in Hough space. These algorithms form a base for the vanishing point detection algorithm utilized in this thesis.

### 4.1.2 Vanishing Point Estimation Methods

Among the literature on vanishing point detection, two key articles are commonly cited in all new approaches: those of Barnard [52] and Magee and Aggarwal [53]. Barnard presents general methods for interpreting perspective projections of three dimensional data onto an imaging plane. He suggests projecting lines in the image plane onto the Gaussian sphere. The Gaussian sphere is an imaginary sphere centered at the camera’s focal point and lying tangent to the imaging plane. Lines projected from the image plane onto the Gaussian sphere become circles (called great circles) which intersect at vanishing points. This changes image points at infinity into finite points on the unit sphere. Bins in a histogram are then populated according to how many lines pass through them and the maximum points indicate a high likelihood of being vanishing points. Criticisms of Barnard’s method include that his parameterization of the Gaussian sphere is “ad hoc” (Magee and Aggarwal claim) meaning that a histogram over the unit sphere produces unevenly spaced grid points.

Magee and Aggarwal propose a different approach which determines lines and vanishing points from successive cross products of homogeneous coordinates, a consequence of the duality of points and lines. This duality is a well-known phenomenon in the computer vision literature and is further discussed in Appendix B.1. Magee and Aggarwal use a similar parameterization of the unit sphere but use a distance metric on the geodetic surface rather than Barnard’s uneven histogram over the unit sphere. They add a constraint that the intersection of two line segments cannot lie on either of the segments in order to decrease the computational load. The Magee-Aggarwal algorithm requires computations on the order of  $n(n - 1)/2$  where  $n$  is the number of intersections. Tracing great circles, as required by Barnard’s method, requires computations on the order of  $nm$  where  $m$  is the number of divisions in each great circle. Therefore, where high precision is required, Barnard’s method may be more computationally intensive.

Some work has been done to extend these vanishing point detection algorithms by adding additional constraints and attempting to make them more robust to real-world scenarios. Schuster et al. [54] use a method based on Magee and Aggarwal’s

algorithm which extends line segments by a given factor and excludes points in those extended regions. They use a simple clustering algorithm on the accumulator elements to group regions close to one another and then compute a weighted average (centroid) on these regions to estimate the vanishing points. They use this algorithm with a camera pointed towards the ceiling and rely on the structure of the ceiling tiles to indicate orientation of a ground robot.

Many extensions of these algorithms, as well as innovative methods quite different from either, have been proposed. These include a cascaded Hough transform [55], a voting scheme [56], a likelihood function [57], and line clustering techniques [58]. This is by no means a comprehensive list, nor is it an attempt to give a complete survey of vanishing point detection techniques. The work in this thesis is based primarily on Barnard’s and Magee and Aggarwal’s approaches. Thus, though these additional methods have merit, they will not be discussed further.

On the quadrotor aircraft, algorithms have been developed that follow the methods of both Barnard and Magee and Aggarwal. Both have their benefits and their limitations, and both will be shown to provide decent estimates of vanishing points which give needed attitude estimates for the vehicle. These methods combined with an understanding of perspective geometry will be shown to also provide an estimate of position relative to the walls, ceiling, and floor of the hallway.

### **4.1.3 Pose Estimation and Three-dimensional Tracking Methods**

Many pose estimation methods seek to find feature points in an image and track these features over time. Corners track particularly well, and thus a Harris corner detector often serves as a pre-processing step. Strong corners or feature points are then tracked over consecutive frames of a video stream, and given these correspondences, iterative pose estimation schemes are employed to estimate the pose of the camera with respect to viewed world points. Several such schemes exist in which an error function is derived and minimized to find the correct pose. One of these methods, proposed by Lu et al. [59], seeks to minimize a cost function in world space iteratively and is proven to be globally convergent. This convergence does not guar-

antee that the algorithm will escape local minima or that it will find the true pose. Many such algorithms are sensitive to false correspondences, so much work seeks to improve robustness of these types of algorithms. One example is the work of Wunsch and Hirzinger[60] in which occlusions are handled robustly in a Kalman-filtering framework which models the dynamic movement of the tracked object.

Drummond and Cipolla [61] propose an effective method for tracking rigid-body objects using a Lie Group representing the space of rigid-body rotations and translations. They use a method of iteratively re-weighted least squares to robustly track objects in real-time. Kemp [13] then implemented a method based upon the rigid-body tracking method to estimate pose on a quadrotor helicopter. The primary insights used in this approach are the use of a multiple hypothesis tracking algorithm which matches line segments in an *a priori* map to line segments observed in a noisy video feed. The video is processed offboard in real-time using parallel processing on a multi-core workstation to maintain a framerate of 50 Hz. The results from Kemp's work suggests that line tracking provides a much more robust tracking system in indoor environments than point tracking. In an indoor setting, point correspondences are frequently unreliable due to poor textures on walls and the aperture problem (only the motion perpendicular to a line is observable). The approach discussed herein of matching hallway lines in image space to lines in 3-D space follows this logic and presents a robust method for determining pose in a hallway.

#### 4.1.4 Visually-Aided Mobile Robot Navigation

A large base of vision-aided navigation and control techniques exist in ground robotics literature. In [62], Desouza and Kak present a survey of techniques for using vision for mobile robot navigation. They explain techniques used for both indoor and outdoor environments in both structured and unstructured environments. One primary citation to note from their survey is the FINALE system of Kosaka and Kak [63]. In summary, the FINALE system projects a known map onto the imaging plane of a mobile ground robot. The system estimates pose by comparing the viewed image with the map and tracking uncertainty in a Kalman-filtering framework. This

allows a ground robot to navigate buildings at substantial speeds (approximately 8-10 meters/min) without being impaired by stationary or slow moving objects. Some features of the FINALE system are similar to approaches attempted here, particularly attempting to track features in a corridor.

The work of Kemp [13] mentioned above falls into this category as well since it applies the 3-D tracking methods conceived by Drummond and Cipolla [61] on a real-world flying quadrotor. His work demonstrates the ability to use vision guidance as the primary sensor for real-time flight navigation. Concepts used in his work apply well here and suggest methods for future work. Kemp also makes it clear that accurate estimates of vehicle velocities are entirely essential to effectively control the position of the quadrotor. This has proved entirely true in attempting to track position commands on the vehicle reliably.

In addition to the map-based navigation methods, Moravec's now famous Stanford Cart experiments [64] made more practical the field of exploring robots. The Stanford Cart was able to explore unknown indoor scenes using stereoscopic imagery in the year 1980, when the field of vision robotics was much less mature. At the time, the cart was able to traverse a 20 meter corridor in approximately five hours. This is mentioned primarily to show that mobile robotics has a long history of innovation and that great things are possible. Moravec's work has been greatly expanded upon. This thesis work is only a small step toward full vision-reliance, but it demonstrates that computer vision is a capable sensor. For much more detailed summaries of both the Stanford Cart and the FINALE tracking system, the reader is referred to the survey paper by DeSouza and Kak [62].

Call [65] presents two methods for detecting and avoiding obstacles on a small unmanned aerial vehicle. his results are primarily demonstrated in simulation and consist of using a known map to plan a waypoint course through an area. Unmapped obstacles are detected using a Harris corner detector and tracked through multiple video frames transmitted to a ground station from a monocular camera. This work by a labmate was a partial motivation to introduce additional computer vision techniques for flight navigation of UAVs.

Though pose estimation and visual servoing approaches abound, the work discussed in this thesis is relevant because it seeks to bring together work from many fields and describe possible solutions to a complex problem. The pose estimation scheme presented here is based upon first establishing rotation by finding the vanishing point representing the point of convergence of the lines that delineate a hallway. Then, to extract position information, a homography is applied to correct for the discovered rotation. Finally, an estimate of position is provided by correlating the ratios of the angles between these lines in the image space. In addition to this method, some early work has been started in 3-D motion tracking, but due to its immaturity, it will be included in Appendix B for the interested reader.

## 4.2 Image Processing Description

The procedure for detecting vanishing points proceeds in four key steps. First, preprocessing is done on the image resulting in an edge image. Using one of several techniques, lines are discovered from the edges. These lines are filtered to find desired sets of lines. Finally, using these resultant lines, vanishing points are estimated. Figure 4.1 shows each of the image processing steps including several alternative methods for each that will be discussed.

### 4.2.1 Edge Detection

A discrete image, such as that from a digital camera, is simply a two-dimensional set of color values which represent the color at each pixel. Many representations of pixel values exist including RGB<sup>1</sup>, HSV<sup>2</sup>, and YUV<sup>3</sup>. Conversions between color representations are straightforward and left to the literature (see [47]). The input to the

---

<sup>1</sup>In the RGB representation of pixels, each pixel is represented by its components of red, green, and blue intensity.

<sup>2</sup>HSV stands for hue, saturation, value. The hue component corresponds to a location on the color wheel. The saturation is a rough approximation of how pure the color is. The value is a measure of how light the color is.

<sup>3</sup>In the YUV space, pixels are represented by one luma (Y) value and two chrominance (UV) values. The luma image is thus a monochrome image and all of the color information is specified by the chrominance channels.

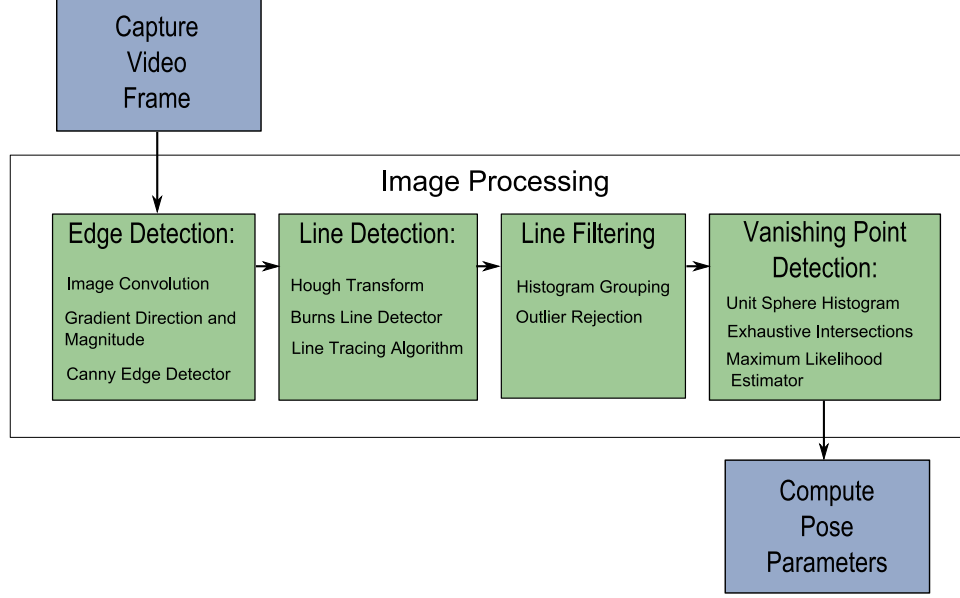


Figure 4.1: **Computer Vision Algorithm Flow Chart:** This flow chart shows the flow of the vanishing point finding algorithm used to determine pose of the aircraft. The image processing steps are shown along with several alternative methods that will be discussed in this section.

edge detection algorithms are intensity values which are created by converting a color image to grayscale (or simply using the Y channel of the YUV color space).

An edge in an image is defined as a region where intensity changes significantly. The principle used in detecting image edges is to compute the gradient of all pixels resulting in a gradient image which represents how much the pixel intensity changes at each given pixel location.

Edge detection is a standard image pre-processing method used to locate changes in the intensity function of an image and is described at length in image processing texts such as Sonka et al.[47]. Edges are detected where the intensity (brightness) changes abruptly (the image gradient is high). Sonka et al. define an edge as a vector variable attached to a pixel with both a magnitude and a direction. The edge direction,  $\phi_e$ , is located at a right angle to the gradient direction,  $\psi_e$  (rotated by  $-90^\circ$ ). The gradient magnitude and direction of an image are computed according to,

$$|\text{grad } g(x, y)| = \sqrt{\left(\frac{\delta g}{\delta x}\right)^2 + \left(\frac{\delta g}{\delta y}\right)^2}, \quad (4.1)$$



Figure 4.2: **Gradient Magnitude Image From the Sobel Edge Detector:** The image gradient magnitude brings out strong edges. This image is the result of using the Sobel kernel to estimate the pixel gradient.

$$\phi_e = \tan^{-1} \left( \frac{\delta g}{\delta x}, \frac{\delta g}{\delta y} \right). \quad (4.2)$$

Since digital images are discrete in nature, gradients are often computed based on differences between a given pixel and its neighbors. Image processing operators are therefore developed which are simply convolution kernels that are passed over an image. Edge detection is very important as a low-level task contributing to many higher level image processing objectives, and therefore remains an active area of research. The sheer number of detection methods makes picking the right edge detector a bit of a challenge. Therefore, methods which have shown promise in other applications have been chosen.

Some of the common operators used for edge detection are the Laplace operator, the Prewitt operator, and the Sobel operator, all shown in Table 4.1. The Laplace operator approximates the second derivative and gives only the gradient magnitude. A  $3 \times 3$  mask representing a convolution sum is often used, which is shown in the table. The Prewitt and Sobel operators both approximate the first derivative. They can be created in several different directions as required.

The gradient direction provides a great deal of interesting data. The principal benefit of it is that lines stand out as connected regions with a similar gradient

Table 4.1: **Common Image Operators for Edge Detection**

Operator	Kernel		
Laplace	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	, $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	, $\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$
Prewitt	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	, $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$	, $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	, $\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$	, $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

direction. This lends itself to line detection methods through line tracing as discussed in Section 4.2.2.

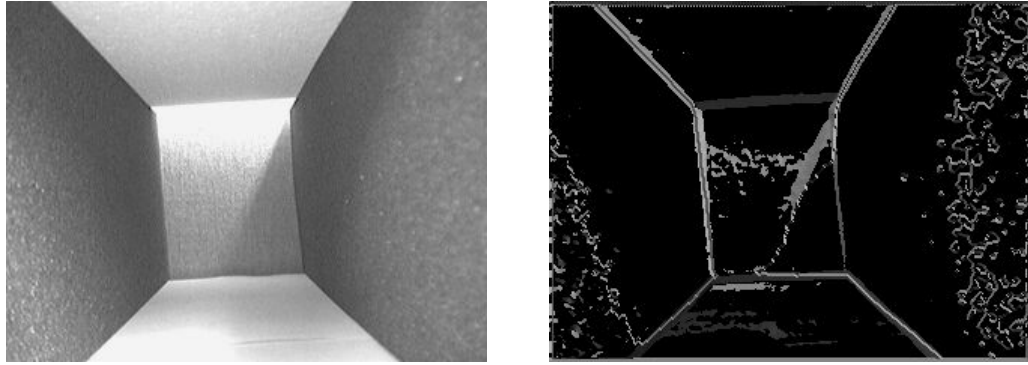


Figure 4.3: **Gradient Direction Image:** The left image is a simple hallway imaged in a video camera. The gradient direction image on the right preserves lines by representing them as regions with similar direction vectors. Lines stand out as being contiguous pixel regions which are easily traced to yield lines.

To speed up the preprocessing stage, the Sobel edge detector can be run very quickly on embedded processors using only add and bitshift operations. Also, CORDIC rotations speed up the calculation of gradient magnitude and direction by using bitshift operations to convert from Euclidean to Polar coordinates.

An optimal edge detector that has gained respect in the computer vision community was designed by Canny in 1983 [66]. The Canny edge detector starts by computing the gradient magnitude of the image. Then, instead of applying a static

threshold to select edge pixels (edgels), thresholding with hysteresis is used. This means that two thresholds are selected. Any pixels with gradient magnitude above the upper threshold are selected as edgels. Then, other pixels adjacent to discovered edgels are also classified as edgels if their gradient magnitude is above the lower threshold. After applying this threshold, non-maximal suppression is used to turn off edgels that are not the maximum magnitude in directions normal to discovered edges.

The Canny edge detector, along with many common operators discussed above, have been incorporated in an open computer vision library created by Intel Corporation entitled OpenCV [29]. Algorithms for edge detection, line detection, morphological operations, and a great variety of other functions have been included making computer vision much more accessible in the programming community. The majority of the programs implemented in this work use OpenCV to some degree, but custom implementations of the algorithms often had to be developed to increase speed or accessibility of variables.

In summary, edges in an image can be detected using a convolution operator which computes changes in intensity around each pixel. The gradient magnitude and direction can then be computed. Methods such as thresholding with hysteresis and non-maxima suppression can also be used as aids for determining true edges. These preprocessing steps are a critical part of the vanishing point detection system.

#### 4.2.2 Line Detection

The next step in the overall image processing algorithm is to group edge pixels into lines. Two main methods from the literature were tested based on the Burns line detector and the Hough transform. In addition to these, a third algorithm was developed which sped up operations from the Burns line detector by tracing adjacent gradient direction edges.

***Method 1: Burns Line Detection*** The Burns line detector [49] is a multi-phase line detection algorithm which groups adjacent regions which share a similar gradient

orientation into line-support regions. The full algorithm consists of the following steps:

1. Group pixels into line support regions based on similarity of gradient direction
2. Fit a planar surface to the intensity surface and intersect this with a vertical plane to determine the position of lines
3. Extract attributes from the lines such as length, contrast, width, location, straightness, etc.
4. Filter lines according to their attributes for various purposes (according to orientation, position, contrast, etc).

The first two steps are critical steps for line detection. To complete the first step, the gradient direction image is thresholded in several overlapping orientation regions, i.e. from  $0^\circ$  to  $45^\circ$ , from  $22.5^\circ$  to  $67.5^\circ$ , etc. Regions sharing pixel orientation are then grouped using connected components and a voting procedure determines which pixels belong to which lines.

The second step conceptually views the intensity image as a surface where high intensity values are peaks and low intensity values are valleys. The algorithm fits a plane to each connected component and intersects this plane with a vertical plane yielding a line location estimate.

The full Burns line detector algorithm requires substantial computing power if performed over the entire image due to the connected component stage of the algorithm. However, it was found that performing an image “AND” operation to the thresholded gradient image and the Canny edge image yielded good lines. This reduced the size of connected components in the image and significantly increased the speed of the algorithm. Results based on this method are demonstrated in Section 4.4.1.

**Method 2: *Hough Transform*** The Hough transform is a very useful method derived from the duality of points and lines. In summary, the Hough transform takes

a binary input image and searches for patterns by sorting all points in an image according to some metric. Two parameters are needed to describe a line, for example the slope and intercept. However, this parameterization has a singularity at a vertical slope (which occurs commonly in images). So, instead, a parameterization using polar coordinates is recommended, replacing the slope-intercept form with a  $\rho, \theta$  form as shown in Equation 4.3. For each edgel in the image, values of  $\rho$  are calculated as a function of  $\theta$  and plotted in an accumulator where they show up as a sinusoid. This type of plot is called the Hough accumulator. Peaks in the accumulator correspond to lines in image space.

The following formula is used to represent a line in polar coordinates:

$$\rho = x \cos \theta + y \sin \theta \quad (4.3)$$

A custom implementation of the Hough transform was developed which used non-maxima suppression to detect peaks and eliminate multiple line discoveries near each peak. Figure 4.4 demonstrates the results of performing the Hough transform on a hallway image. The hallway has very well-defined black walls and a white ceiling. The Hough accumulator is shown in subfigure (b) where peaks are circled in colors corresponding to the discovered lines in (a). Vertical lines are drawn in blue and horizontal lines in green. The vanishing point, discovered using a method discussed in Section 4.2.4 is shown as a blue dot.

**Method 3: Line Tracing** A second, faster, method was developed for use on an embedded processor without floating point hardware. This method starts off in a similar fashion to the Burns line detection method by thresholding the gradient direction image in regions of interest. Only pixels above a certain gradient magnitude are searched. Starting at the outside of the image and working inward, lines are traced, looking for contiguous regions. Lines found using this method appear in green in Figure 4.5.

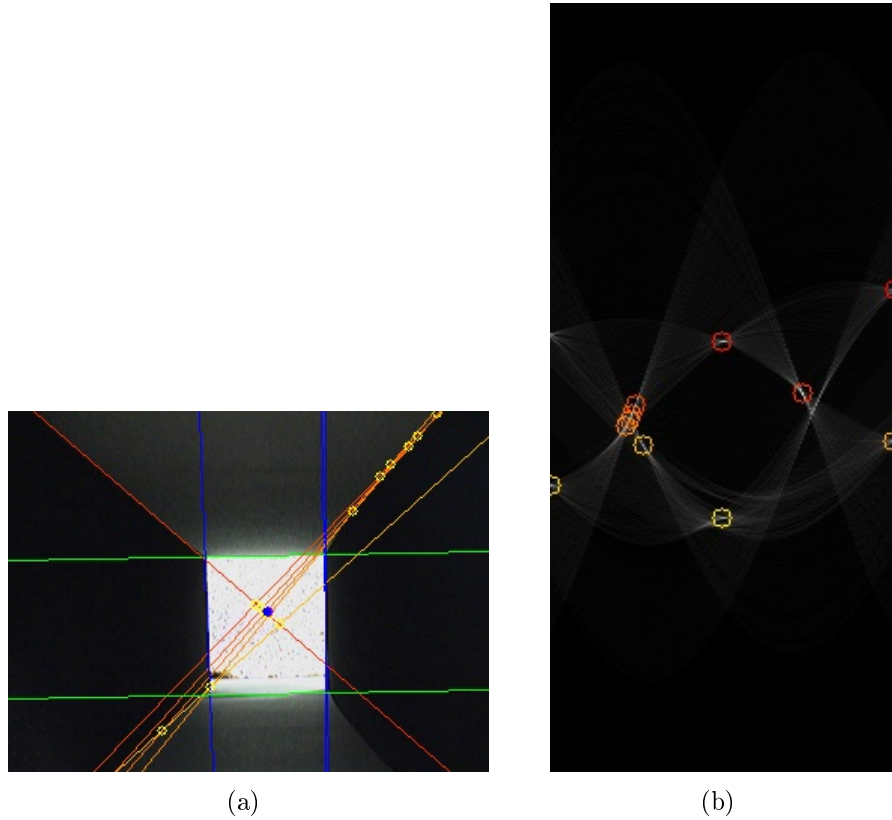


Figure 4.4: **Hough Transform Image:** Lines in image space map to peaks in Hough space. (a) Discovered lines are shown sorted by orientation. Horizontal lines are green, vertical are blue, and diagonal are colored to match the corresponding peak in Hough space. The blue point in the center represents the estimated vanishing point. (b) The Hough accumulator image shows individual peaks in Hough space circled in a color corresponding to the line in the image.

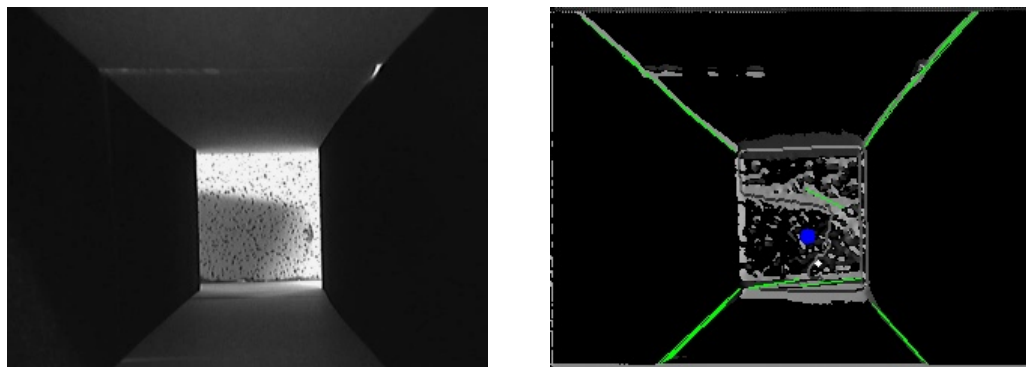


Figure 4.5: **Line Tracing Algorithm Results:** The Line tracing algorithm find the prominent lines rather effectively by following patches of pixels with similar gradient orientations. Once again, the blue dot represents the vanishing point.

### 4.2.3 Line Grouping/Filtering

Having found many lines in the image, it then becomes necessary to decide which lines support different vanishing point locations. In a typical scene, there are several vanishing points, so all lines do not necessarily converge to the same point. We wish to estimate the vanishing point at the end of the hallway, but other vanishing points are also of interest. For example, vertical lines converge to a vanishing point near infinity in the positive z-direction. If this vanishing point is found, it will give information useful for determining the roll angle of the vehicle. We attempt to partition the lines into groups that converge at different vanishing points.

The vanishing point of the horizontal lines between the floor, walls, and ceiling is the primary vanishing point of interest because it will give us a measurement of our heading and pitch angle. Therefore, we initialize the vehicle with an orientation towards this vanishing point.

As discussed in the related work section, Barnard suggested grouping lines together using a histogram over the projection of the image plane onto the Gaussian sphere centered at the camera focal point. Lines in the image then form great circles which intersect at the vanishing points. This reduces the line sorting problem into a histogram and peak detection problem. However, the accuracy of the estimate depends on the spacing of the histogram. It is desired to find a region near the last vanishing point where several lines pass. Thus a good approach is to create a histogram over the image plane near the last vanishing point. Using a coarse histogram, lines converging to a similar point are grouped. The area with the largest support of lines is chosen as the best estimate of the vanishing point and the set of lines through this area is selected and passed into the final portion of the algorithm where the vanishing point estimate is further refined.

### 4.2.4 Vanishing Point Detection

Imaging of three-dimensional scenes through a perspective camera produces images with lines that converge to vanishing points. The locations of vanishing points

are invariant to position and thus provide vector measurements similar to those estimated by accelerometers, magnetometers or star-tracking. Stated a different way, vanishing points constrain the attitude of the vehicle with respect to the hallway providing an efficient way of estimating attitude regardless of position.

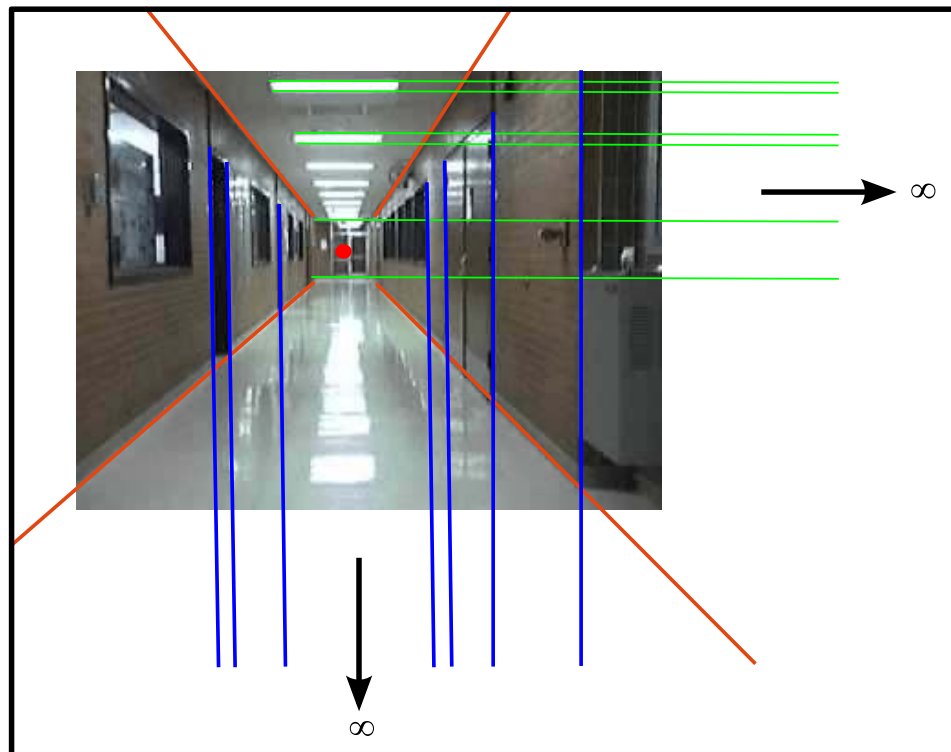


Figure 4.6: **Example Hallway Image:** Parallel lines in three dimensional space converge to vanishing points in the image plane. Lines that intersect the image plane appear as points such as the red dot at the end of the hallway. Lines that are parallel to the image plane “converge” to points at infinity in the image plane which can be defined by a two-dimensional vector.

A line in three dimensional space can be defined by a point  $\mathbf{X}_0^1$  and a direction  $\mathbf{V}$ . Let  $\mathbf{X}_0^1 = \begin{bmatrix} X_0^1 & Y_0^1 & Z_0^1 & 1 \end{bmatrix}^T$  be a point on a line  $L^1$  and  $\mathbf{V} = \begin{bmatrix} V_1 & V_2 & V_3 & 0 \end{bmatrix}^T$  be the direction of the line. All other points on the line  $L^1$  can be expressed as

$$\mathbf{X}^1 = \mathbf{X}_0^1 + \mu \mathbf{V} \quad (4.4)$$

where the scalar  $\mu$  represents the distance from  $\mathbf{X}_0^1$  along the line. Two parallel lines in three dimensional space, then, can be viewed as intersecting at a point at infinity with coordinates  $\mathbf{V}$ , and the image of this intersection is a vanishing point in the image plane. The intersection of parallel lines is expressed mathematically by the fact that parallel lines meet at points with a homogeneous image point with a third coordinate of zero.

Figure 4.6 shows how various types of world lines in a hallway are projected onto the image plane. The parallel lines marking the intersection of the walls with the ceiling, for example, lie orthogonal to the image plane and thus have a common vanishing point near the center of the image. The horizontal lines do not appear to intersect, but can be thought of as intersecting at infinity. The same can be said of vertical lines. Each of the vanishing points in the image provides information useful for determining the attitude of the vehicle relative to its surroundings. The vanishing point near the center of the image, for example, indicates the camera's orientation relative to the hallway and can be used to determine pitch and yaw angles. The vanishing point at infinity to which the vertical lines converge indicates the roll angle of the vehicle. Thus, by finding the various vanishing points in the image, the full attitude of the vehicle can be determined.

***Histogram Method for Vanishing Point Detection*** As in the line grouping algorithm, a histogram can be used to estimate the best vanishing point. If  $n$  lines are found in an image, there exist  $n(n-1)$  intersections of these various lines. Using a line of thought similar to that proposed in Magee and Aggarwal [53], all intersections of the discovered lines can be accumulated over the image plane. Vanishing points will typically have a great deal of line support, and thus many line intersections will pass through them. Peaks in the intersection accumulator space, then, represent likely candidates for vanishing points. Once again, using a coarse histogram, peaks in the accumulator space are chosen as likely vanishing points and lines near these candidates are chosen as support for selecting the best vanishing point in the final step of the algorithm.

**Maximum Likelihood Estimation of Vanishing Points** The set of convergent lines discovered in the first several steps of the image processing is prone to error. Pixelation of light intensities and inaccuracies in the Hough transform contribute to this error. In effect, though a set of lines may pass through a single point in real life, the discovered lines in the image typically do not. Many vanishing point estimation methods simply choose the point closest to the set of lines that converge to a single vanishing point. In this work, a maximum likelihood estimator described in [50] is used to find the best vanishing point iteratively while allowing the line slopes to change slightly. Thus, the point which minimizes how much the line slopes change is chosen as the best estimate of the vanishing point. The error metric is the perpendicular distance between the endpoint of the image line and the line which passes through the image line centerpoint and proposed vanishing point as illustrated in Figure 4.7.

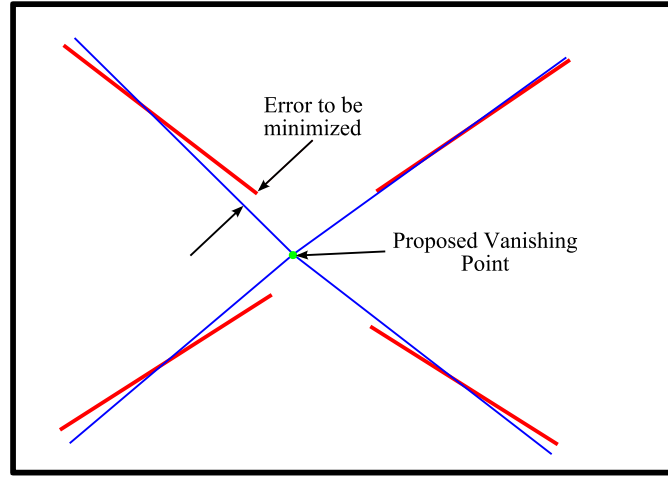


Figure 4.7: **Vanishing Point Error Definition:** The error associated with a proposed vanishing point is defined as the perpendicular distance between the image line and the line that goes through the center of the image line and the proposed vanishing point.

Mathematically, the error associated with a candidate vanishing point is represented by an error vector  $\mathbf{d}$  where each element is computed according to

$$d_i = \frac{\mathbf{v} - \mathbf{c}_i}{\|\mathbf{v} - \mathbf{c}_i\|} \times (\mathbf{p}_i - \mathbf{c}_i) \quad (4.5)$$

where  $\mathbf{v} = (v_x, v_y)$  is the location of the vanishing point,  $\mathbf{c}_i = (c_x, c_y)$  is the centerpoint of the line, and  $\mathbf{p}_i = (p_x, p_y)$  is an endpoint of the line.

To iteratively determine the best vanishing point, an initial guess is provided (typically the vanishing point from the previous frame) and Levenburg-Marquadt iteration is performed. The Levenburg-Marquadt algorithm is a quasi-Newton method of optimization combining Gauss-Newton iteration with gradient descent depending on whether the previous iteration improved or regressed. This relies on having an explicit formula for the gradient of the error function which we seek to minimize. Therefore, we need equations for how the error varies with the selection of the vanishing point. These quantities, termed  $\frac{\partial d}{\partial v_x}$  and  $\frac{\partial d}{\partial v_y}$ , are found to be

$$\frac{\partial \mathbf{d}}{\partial v_x} = \frac{(p_y - c_y)h^{\frac{1}{2}} - h^{-\frac{1}{2}}(v_x - c_x)f}{h}, \quad (4.6)$$

$$\frac{\partial \mathbf{d}}{\partial v_y} = \frac{-(p_x - c_x)h^{\frac{1}{2}} - h^{-\frac{1}{2}}(v_y - c_y)f}{h}, \quad (4.7)$$

where  $f = (v_x - c_x)(p_y - c_y) - (v_y - c_y)(p_x - c_x)$  and  $h = (v_x - c_x)^2 + (v_y - c_y)^2$ . We denote the gradient function

$$\epsilon_p = \begin{bmatrix} \frac{\partial \mathbf{d}}{\partial v_x} \\ \frac{\partial \mathbf{d}}{\partial v_y} \end{bmatrix} \quad (4.8)$$

and seek to minimize the sum of the squared error represented by the inner product  $\mathbf{d}^T \mathbf{d}$ . The Levenburg-Marquadt optimization algorithm approximates the Hessian of the function as

$$g_{pp} = \epsilon_p^T \epsilon_p + \epsilon_{pp}^T \mathbf{d} \quad (4.9)$$

and seeks to find an iterative solution. The solution is reached by computing an increment  $\Delta$  to the vanishing point estimate at each iteration according to

$$\Delta = (\epsilon_p^T \epsilon_p + \lambda I)^{-1} \epsilon_p^T \mathbf{d} \quad (4.10)$$

where  $\lambda$  changes depending on whether the error increased or decreased. If the error grows,  $\lambda$  is increased by a large factor, making the solution method resemble gradient

descent. If the error diminishes,  $\lambda$  is decreased making the method approach Gauss-Newton iteration.

This maximum likelihood algorithm provides a fast, iterative algorithm for determining the best vanishing point given somewhat noisy measurements. The algorithm converges quickly when line estimates are good. If outliers are present and very few lines are found, the estimate may diverge or converge to a false solution. The approach used here is to attempt to remove outliers using the grouping methods mentioned previously and to tune the Hough transform to find many lines to decrease the effect of single outliers. The convergence of the algorithm is pictured in Figure 4.8 where the initial guess is very few from the true vanishing point. The algorithm converges in just five steps to the optimum.



Figure 4.8: **Example of Vanishing Point Detection:** Here, the vanishing point is detected using the Levenburg-Marquadt algorithm. The approximation to the Hessian quickly drives the estimate of the vanishing point to the optimum location.

### 4.3 Pose Estimation From Vanishing Points

Once the best vanishing points are determined from the image, they can be used to determine an estimate of the attitude angles. The vanishing point at the center of the hallway is used to determine yaw and pitch while the vertical lines determine the roll of the vehicle.

### 4.3.1 Attitude Estimation

Estimating the vanishing point locations is as simple as reversing the projection from the image plane onto the Gaussian sphere centered at the camera center:

$$\psi = \tan^{-1} \left( \frac{\gamma_x}{f} \right), \quad (4.11)$$

$$\theta = \tan^{-1} \left( \frac{\gamma_y}{f} \right) \cos \psi, \quad (4.12)$$

where  $\gamma_x$  is the distance in the  $x$ -direction of the vanishing point from the center of the image,  $\gamma_y$  is the vertical distance to the center, and  $f$  is the focal length of the camera (which can be discovered through calibration).

This measurement of yaw angle can be combined with the yaw rate gyro in a simple one-state Kalman filter to provide a reliable estimate of heading with bounded error. The yaw rate gyro is capable of estimating the yaw angle through integration, but after several seconds, the estimate becomes unreliable. But, with visual feedback of finding the vanishing point in the image, this error is bounded. This is demonstrated in the results section by placing the quadrotor on a yaw-plate where it is constrained to only move in the yaw direction. Onboard computing using the Pico-ITX processor to discover the vanishing point. The quadrotor is thus able to consistently find the correct heading and point itself straight down the hallway.

The roll angle can also be determined by determining which lines are vertical. Figure 4.4 shows an image where horizontal, vertical, and diagonal lines are grouped separately. The vanishing point to which the vertical lines converge is in a downward direction. An estimate of the vector pointing towards this vanishing point is computed by finding the homogeneous intersections of all vertical lines. A single intersection  $\bar{\mathbf{X}} = [\bar{x}, \bar{y}, \bar{z}]$  indicates a roll angle of

$$\phi_v = -\tan^{-1} \frac{\bar{y}}{\bar{x}} \quad (4.13)$$

where it is assumed that  $\bar{z}$  is small (since the intersection is near infinity). Thus, the estimate of roll is estimated by averaging the roll estimate from all of the vertical line intersections. This provides a roll estimate that does not drift and is independent of position in the hallway.

#### 4.3.2 Position Estimation

If the four main lines outlining the hallway can be found in a consistent and reliable manner, an estimate of the vehicle's position in the hallway can be estimated. To correctly estimate position, the discovered lines in the image must be rectified by compensating for the attitude angles  $\psi$  and  $\theta$ . Using a homography matrix based on the rotation matrix of the vehicle discovered in the previous steps, a correction is thus applied that moves the discovered vanishing point to the center of the image. Effectively, the line projection due to yaw and pitch is removed, and the vehicle's position relative to the four lines is estimated.

The homography which removes the yawing and pitching motion can be computed as

$$H = KRK^{-1} \quad (4.14)$$

where  $K$  is the intrinsic camera calibration matrix and

$$R = \begin{bmatrix} \cos \psi & 0 & \sin \psi \\ \sin \theta \sin \psi & \cos \theta & -\sin \theta \cos \psi \\ -\cos \theta \sin \psi & \sin \theta & \cos \theta \cos \psi \end{bmatrix}. \quad (4.15)$$

Applying this homography to the vanishing point places it in the exact center of the image. Then, looking at the four lines outlining the hallway and assuming we know the aspect ratio of the hallway, we can determine the ratios of the arcs between the lines as shown in Figure 4.9. The vehicle's height and lateral position in the hallway can then be estimated from these ratios according to

$$y = m_y (\tan^{-1} (ratio_{LR})) + b_y, \quad (4.16)$$

and

$$z = -m_z \left( \tan^{-1} (ratio_{UD}) \right) + b_z \quad (4.17)$$

where

$$ratio_{LR} = \frac{\theta_{UL} + \theta_{LL}}{\theta_{UR} + \theta_{LR}}, \quad (4.18)$$

and

$$ratio_{UD} = \frac{\pi - \theta_{UL} - \theta_{UR}}{\pi - \theta_{LL} - \theta_{LR}}. \quad (4.19)$$

The calibration coefficients ( $m_y$ ,  $b_y$ ,  $m_z$ , and  $b_z$ ) were determined using the hallway simulation model described in Appendix B.2.

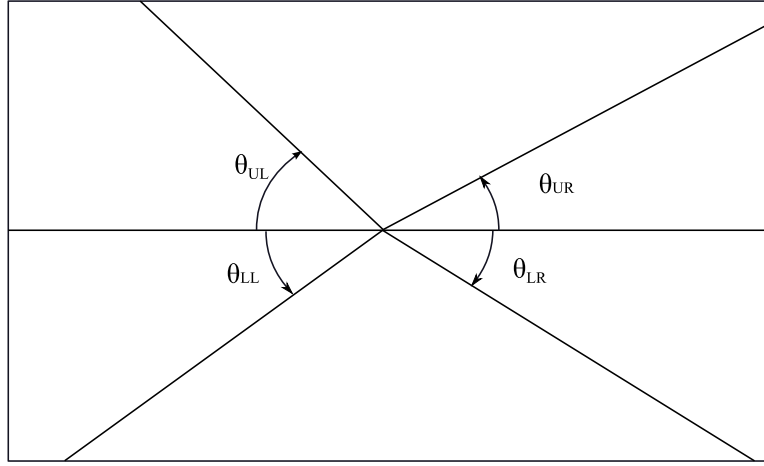


Figure 4.9: **Line Angle Definitions:** Definition of line angles used to estimate position.

#### 4.4 Results

Several methods of detecting vanishing points and hallway position were attempted, with varying levels of success. The two main methods attempted were (1) using a Burns Line Detector and line clustering using a histogram to cluster line segments and (2) using a Hough Transform method and intersecting all lines exhaustively. The Burns Line Detection method showed promise and was implemented on the onboard gumstix processor. The second method proved to be the most consistent

and reliable. Both were used to demonstrate heading hold on a yaw plate. Visual results of these two methods are presented.

#### 4.4.1 Burns Line Detection

First, an application was written in the Windows operating system. This application, called HeliVisionApp is a great testbed for developing algorithms and consists of a GUI frontend which uses OpenCV's HighGUI[29] to capture and display images. A screenshot of the application is shown in Figure 4.10.

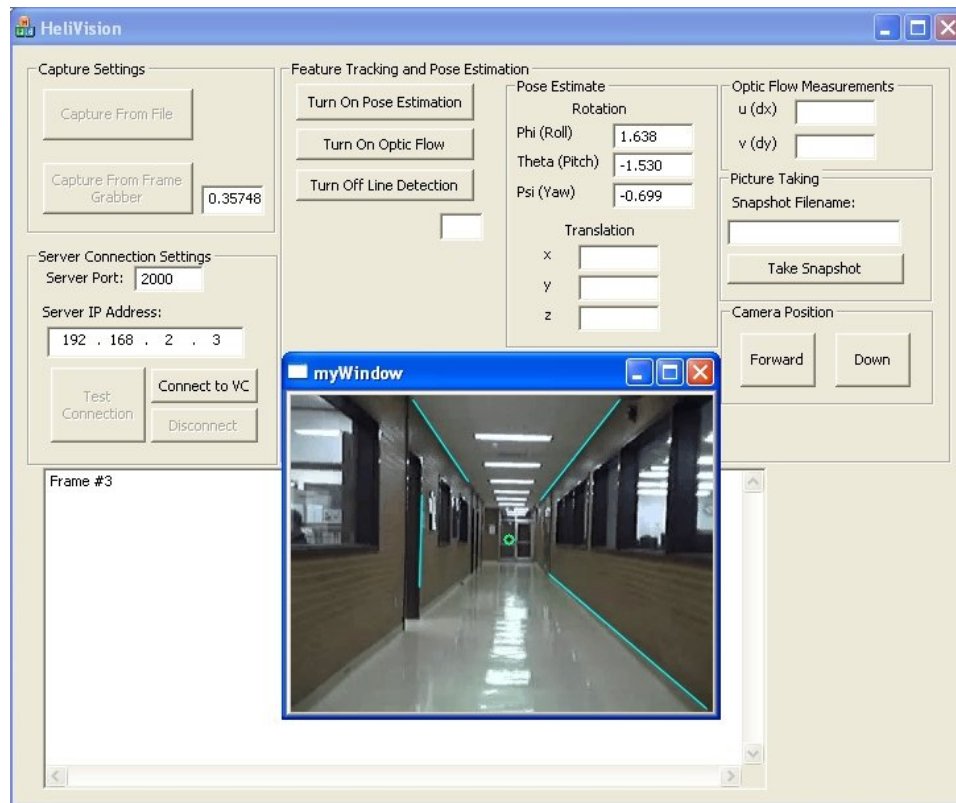


Figure 4.10: **HeliVisionApp Application**

This application was intended to be used on a ground station and therefore endured some changes when implemented onboard the helicopter. It demonstrated an ability to find lines in each frame using a Burns line detector customized to the hallway at hand. The line detector performs a search for lines by thresholding the gradient direction image and looking for connected components. The discovered lines

are then passed into the Levenburg-Marquadt optimization method which finds the “best” vanishing point considering all of the lines.

In the majority of video frames, this algorithm reliably detected the prominent lines on the edges of the hallway. Figure 4.11 shows several successful frames in which the vanishing point is correctly identified. As long as two diagonal lines were correctly detected and no outliers were present the algorithm performed well. Two problems with this algorithm were encountered, however. First, during some frames, blur due to rapid camera motion prevented the line finding algorithm from detecting any lines. In these cases, no vanishing point was detected and the prior estimate was retained as shown in frame 125 of Figure 4.12. Remarkably, even in these cases, the algorithm was able to reacquire the correct vanishing point in successive frames since a global line search is performed when each frame is received. So, though the algorithm occasionally failed, estimates did not drift and thus were still effective in correcting biases due to rate gyro drift. The second problem in the algorithm was associated with outliers. If few lines were discovered, outlying lines greatly affected the vanishing point estimate, as shown in frames 115 and 127 in Figure 4.12. The Levenburg-Marquadt algorithm converges to a false estimate which could be detrimental if used for attitude estimation. Thus, outlier rejection algorithms were implemented which threw out estimates that varied too greatly from Kalman-filtered estimates from the inertial sensors.

This same algorithm was implemented on a Gumstix micro-computer and used to track and control heading on a quadrotor vehicle. Results using this configuration are included in Chapter 5. Many changes were made to speed up execution time on the Gumstix, which does not have a hardware floating point unit and which has a considerably slower processor speed than a typical desktop PC. With these improvements to the algorithm, a frame rate of 7-10 Hz was achieved which was effective at correcting heading measurements in real-time onboard the quadrotor.

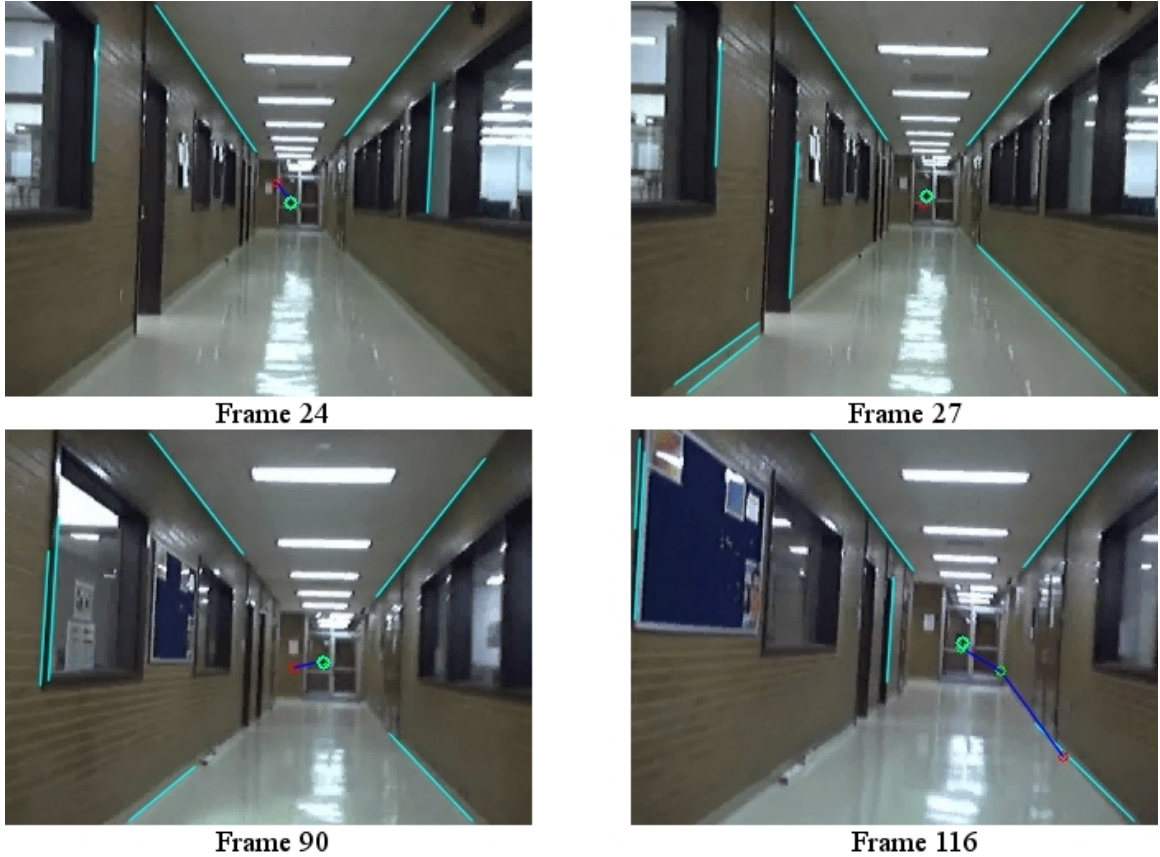


Figure 4.11: **Burns Line Detector Successes:** The Burns Line Detector was able to detect the vanishing point in many of the frames, even if the prior vanishing point discovered erroneously, as shown in Frame 116.

#### 4.4.2 Hough Transform Method

Hough transforms generally require a large amount of processing because for each point in the edge image, a series of lines are drawn in the Hough-space accumulator. However, on a faster processor like the Pico-ITX, this was reasonable, so an algorithm was written using the Hough transform. In this method, all peaks in the accumulator space above a given threshold were selected. Clusters of lines were found by assuming that vertical and horizontal lines converge to a different vanishing point than do diagonal lines. A histogram was made in image space by sampling evenly over the unit sphere. Peaks in this space were compared and the best peak near the center of the image was selected as the optimal estimate of the vanishing point.



Figure 4.12: **Burns Line Detector Failures:** If the number of lines found from the Burns Line Detector was insufficient, the Levenberg-Marquadt algorithm converged incorrectly.

This method proved superior to the Burns line detector method for several reasons. First, real-time visual feedback from the onboard camera allowed the camera settings to be tuned prior to flight. Thus, the camera settings and algorithm settings were selected to find lines that reliably converged to vanishing points. Second, the algorithm did not rely on any predetermined parameters of the hallway itself. The Burns line detector presupposed a knowledge of the gradient direction of each line. The Hough transform effectively found all lines in the image and grouped them properly. Frame rates were somewhat slow in this algorithm due to an exhaustive computation of the Hough transform. In addition, some outlier rejection was needed, but reliable estimates were consistent. The vehicle was tested once again on

a yaw-plate and showed the ability to hold heading for over three minutes without any noticeable drift.

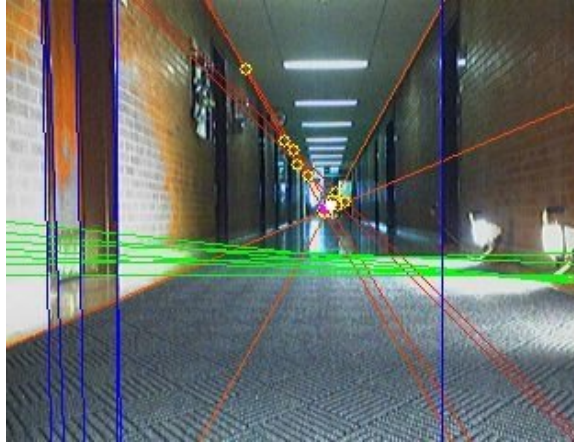


Figure 4.13: **Finding All Vanishing Points:** Hough Transform Method of Finding all vanishing points

## 4.5 Summary

In this chapter, vanishing point detection was discussed including related work and actual implementations on a quadrotor helicopter. Edge detection, line detection, line clustering, and vanishing point estimation all play crucial roles in determining pose on the aircraft. Methods for determining attitude and position from discovered vanishing points were demonstrated. Image processing results for two particular methods were compared, and the best algorithm based on a Hough transform was elaborated upon. Flight results using these methods are further discussed in the next chapter.



## Chapter 5

### Results and Discussion of Results

To culminate the work presented in this thesis, each of the elements from control, estimation, and computer vision discussed in earlier chapters were combined. Difficulties such as varying trim angles and inconsistent response to battery voltage made real-life demonstrations significantly more challenging than simulations and predictions, but occasionally Murphy’s Law abated enough to collect valuable data and demonstrate useful abilities on the quadrotor. These capabilities include hands-off operation, hallway guidance, visual detection of vanishing points, and heading correction during flight.

It is important to note some of the difficulties encountered in demonstrating the capabilities developed here. Therefore, in addition to the positive results, some notes on what made things difficult will also be presented. Emphasis will be placed on developments that have not been published elsewhere such as the use of the optic flow sensor for positioning and vanishing point detection for attitude estimation. The results are divided into three main areas: position hold, heading estimation from vanishing points, and hallway following.

#### 5.1 Reliable Position Hold

The ability to reliably maintain position on a six degree-of-freedom vehicle depends primarily on two things: fast, reliable, non-drifting position estimates and the ability to control position by tracking attitude angles. Using the control loop structure developed in Section 2.5.3, the quadrotor was able to hold its position

within a one meter radius circle for a seven minute flight using the optic flow sensor as the only position sensor.

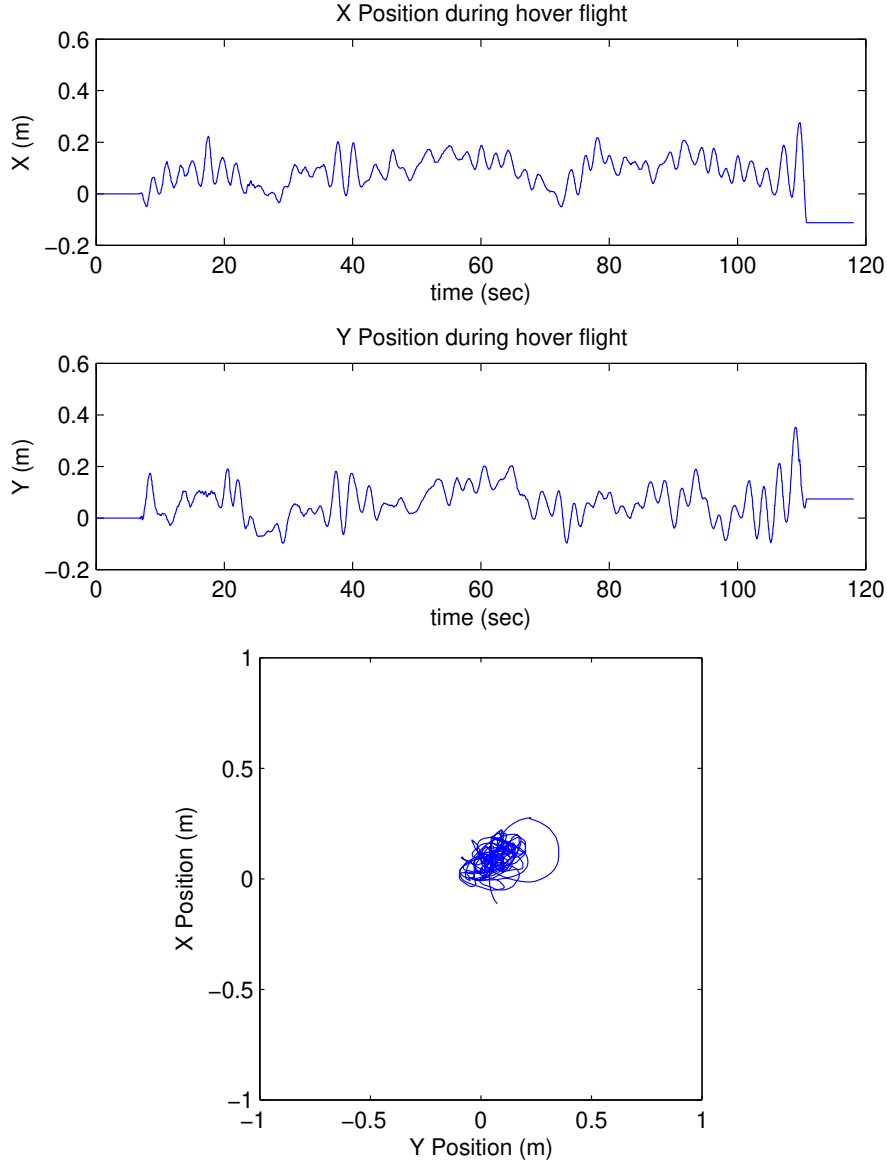


Figure 5.1: **Position Hold Results:** Complete hands-off hover flight is demonstrated in this data plot of actual position measured by the optic flow sensor. This demonstrates the ability of the helicopter to hover well provided a good position/velocity measurement. True position measurements are not available, but a video recording shows very little drift, estimated to be less than one meter over a period of 7 minutes.

Three hover flights were conducted, each demonstrating a reliable position hold. The first lasted approximately 90 seconds in duration and a small amount of

drift occurred. The second flight lasted approximately two minutes, at which time the batteries were exhausted and replaced. After a small amount of retuning due to a change in response to new batteries, a seven minute flight without manual position corrections was performed. A portion of this third flight is shown in Figure 5.1. Due to limitations in memory onboard the autopilot, telemetry data is only available for a portion of the flight, but this data shows an ability to track position effectively. The optic flow sensor readings of position error are always less than 0.4 meters, showing that the helicopter tracks position very reliably. Truth values are not available, but a video of the flight shows that very little drift occurred.

The full sensor suite needed to demonstrate this ability was remarkably small. Only a set of rate gyros, accelerometers, a sonar, and the optic flow sensor were used to estimate the full pose. The total cost of this sensor suite is rather small when compared to sensor suites on larger vehicles with similar capabilities. Despite vulnerabilities to drift in the rate gyros, the attitude estimate remained well-centered and the position estimate drifted remarkably little during the seven minute flight. The heading estimate, relying only on the integral of the rate gyro, also drifted very little, estimated to be less than  $30^\circ$  over seven minutes of time.

Results such as these rely on certain special conditions. First, the optic flow sensor must have adequate lighting to maintain a high surface quality reading during position-hold flight. Otherwise, some amount of position drift will occur as the attitude estimates are imperfect and lead to errors in position estimates. Lighting was enhanced using bright flood lights which contained light in the specific part of the spectrum to which the OFS is sensitive. Second, the ground surface must be adequately texture-rich and of a complimentary brightness for the OFS to read pixel flow across its imaging array. The hover flights mentioned here took place in a room with a finely-textured carpet. The small field-of-view lens was able to pick up texture change well in this situation, making the estimates reliable. However, surface quality readings over tile floors and floors with reflective coatings were poor and made position and velocity estimates unreliable. This makes the optic flow sensor effective

only in certain conditions, limiting its effectiveness. However, when the conditions are right, no offboard cameras or GPS information are required for long-term flight.

Additional tests were completed using onboard lighting. First, red LED lights were attached to the underside of the vehicle providing light in the desired area of the spectrum. However, the LED lighting was insufficient to provide usable optic flow readings. A bright xenon bulb provided surface quality readings comparable to those provided by flood lights and was chosen as a superior lighting method. Hover flight results similar to those mentioned earlier were demonstrated using this spotlight. This capability greatly increases the sensor’s utility in real world situations such as rooms lit by fluorescent light.

## 5.2 Heading Hold using Computer Vision

To demonstrate the effectiveness of computer vision algorithms, the quadrotor was first tested on a yaw plate - a device intended to constrain motion of the vehicle to be around the  $z$ -axis. This “lazy Susan” device allowed testing of the heading estimate from computer vision independent of all other measurements.

A simple experiment was carried out in which the quadrotor was placed on the yaw plate and commanded to a heading of zero which corresponded to facing directly down the hallway. First, an algorithm based on the Burns line detector was implemented on the gumstix processor. This algorithm computed heading estimates at rates of approximately 4 Hz, comparable to a GPS sensor. A single-state Kalman filter was implemented on the autopilot which was tuned to trust the computer vision estimate very highly and integrate the yaw rate gyro in between vision updates. The quadrotor was able to maintain heading for over 30 seconds despite biases on the rate gyro. The quadrotor was manually pulled away from its forward facing position and the heading control quickly returned it to the desired forward-facing heading.

A second experiment demonstrated even better results using the superior Hough transform method of detecting lines implemented on the Pico-ITX. Figure 5.2 compares the results from simply integrating the yaw rate gyro signal with the superior estimate which came from fusing gyro data with vanishing point detection.

An EKF was used to combine these measurements. Several times, the plate was turned, moving the quadrotor as much as  $20^\circ$  away from the desired heading (noted as “Disturbances” in the figure). The helicopter consistently returned to the desired angle facing itself straight down the hallway. The estimate from integrating gyros drifted a total of  $80^\circ$  over the four minute test. The vision estimate is capable of effectively correcting this drift.

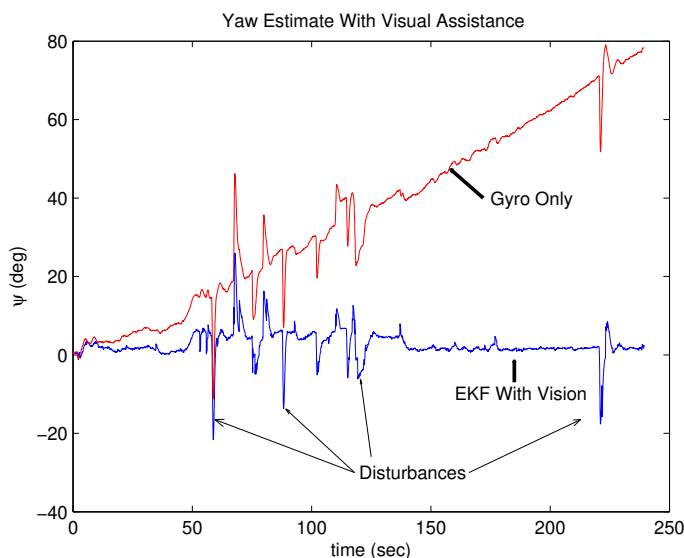


Figure 5.2: **Yaw Estimation Using Vanishing Point Detection:** Over a period of four minutes, the vanishing point detection algorithm was used to correct the heading estimate. Here, the estimate from integrating the yaw rate gyro signal is compared with the estimate which results by fusing the gyro measurement with computer vision. The vanishing point detection corrects the yaw estimate and the quadrotor is able to maintain heading down the hallway for the entire time. The quadrotor was disturbed several times, but was always able to correct itself.

Figure 5.3 compares the yaw estimate from computer vision with the fused estimate of yaw in the EKF. Vanishing point detection produces estimates that do not drift, but occasionally outlying estimates occur. A simple outlier rejection algorithm threw out vision estimates that were more than five degrees away from the EKF estimate. This allowed the good high-frequency estimate of the gyros to be combined with the low-frequency vision update, taking the best of both worlds.

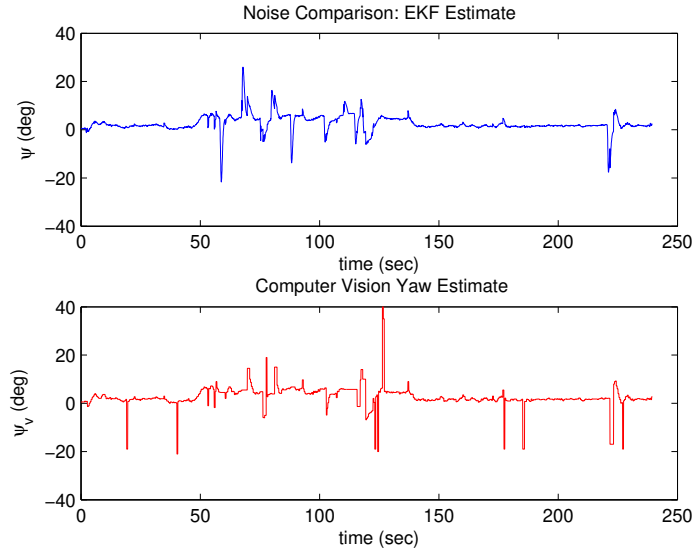


Figure 5.3: **Comparison of Estimation Noise:** The yaw estimate from vanishing point detection contains outliers that must be rejected.

### 5.3 Hallway Following Using The Optic Flow Sensor

The original goal in this work was to fly a helicopter through a corridor using a combination of inertial and visual sensors. This has been achieved by using a combination of a sensor suite and an onboard vision system. The best set of results is demonstrated in Figure 5.4 where the quadrotor's path is shown in blue. The quadrotor started centered between the walls of a four-meter-wide hallway. The red lines denote the positions of the walls. Using velocity control, the quadrotor was steered down the hallway with only occasional corrections to lateral position. In this test, flood lights provided lighting for the optic flow sensor. This method provided the capability of following the hallway in several successive flights.

Figure 5.5 shows telemetry data accompanying the hallway flight. The HAG (in plot (a)) was rather low to provide good surface quality readings on the optic flow sensor (shown in (b)). The quadrotor following a steady trajectory from the starting location to a point approximately six meters down the hallway. The lateral position error was less than a third of a meter throughout the flight. Several hallway following flights were completed demonstrating a consistent ability to navigate the quadrotor in a confined area.

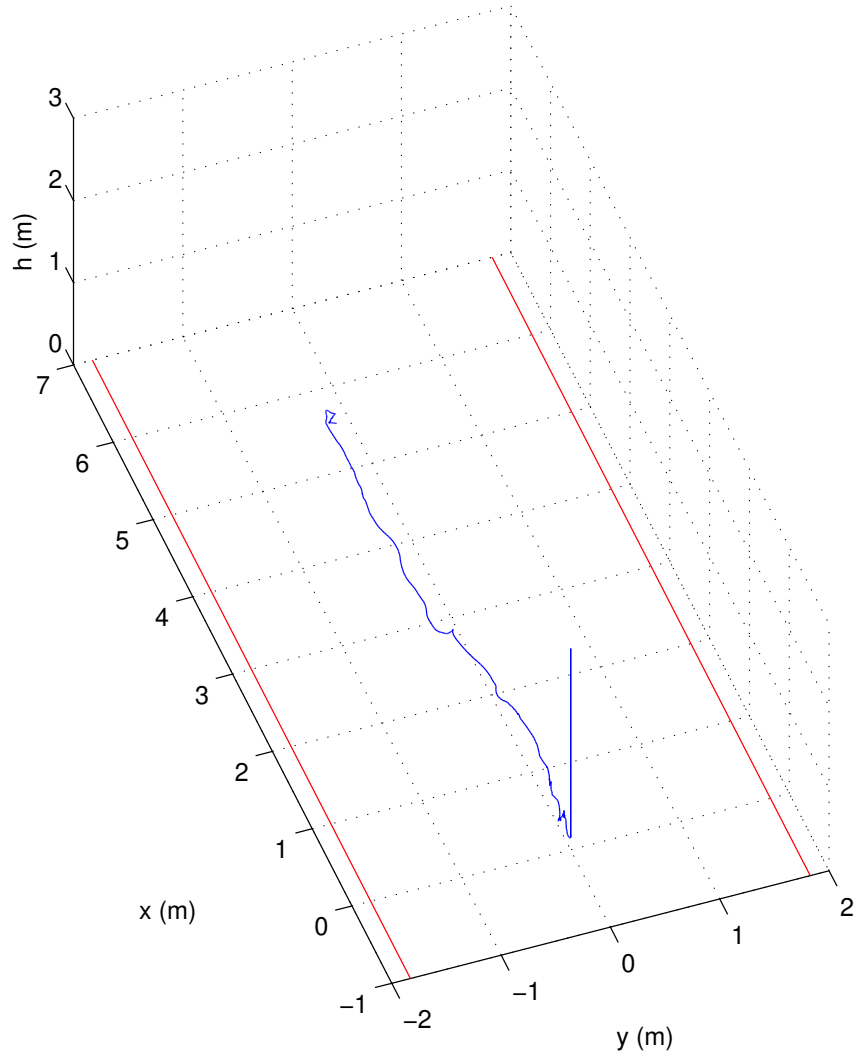


Figure 5.4: **Hallway Following Telemetry Plot:** The quadrotor’s position as reported by the optic flow sensor is shown in blue. An accompanying video displays the capability of navigating the hallway using desired velocity commands. The spike at the beginning is due to a spurious sonar reading.

#### 5.4 Discussion of Results

Quadrotors have traditionally been troublesome devices to fly reliably (see [19] and [17]). High accuracy of attitude and position estimates along with high estimation frequencies are necessary to provide reliable control. The results presented above demonstrate an ability to deal with these attitude and position estimation

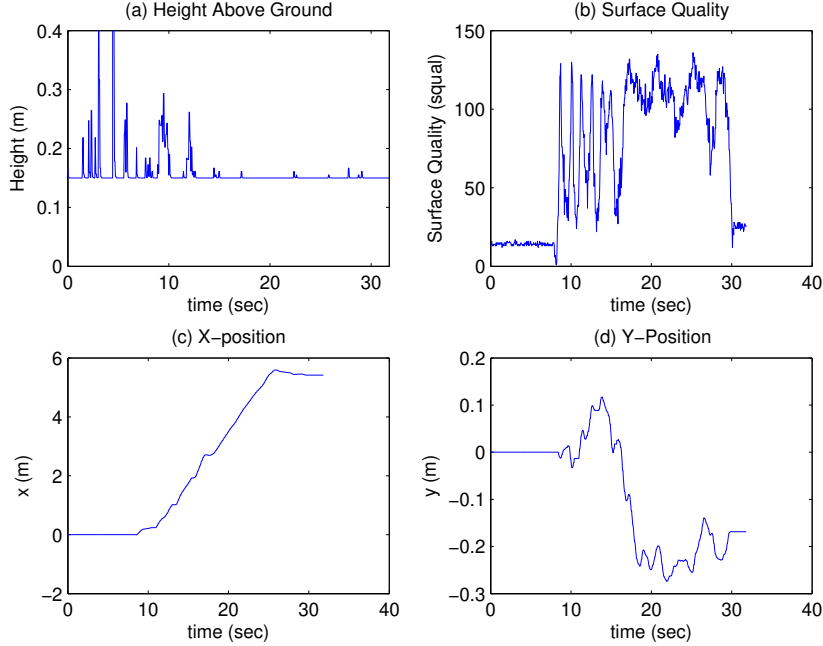


Figure 5.5: **Hallway Following Telemetry Data:** Flying at a low height, shown in (a), the optic flow sensor’s surface quality reading was fairly consistent after take-off. Plots (c) and (d) demonstrate trajectory following with very little error over the corridor flight.

challenges by using a combination of onboard sensors and computer vision. Full pose can be determined using the sensor suite of rate gyros, accelerometers, sonar, optic flow sensors, and computer vision. The ability to combine the capabilities of each individual sensor has been demonstrated.

While hover flight and indoor navigation have been demonstrated to be possible, several factors make quadrotor testing a tedious process. First, quadrotor dynamics change substantially when minor mechanical changes are made. For example, occasional rough landings often caused one of the motor shafts to rotate slightly, changing the attitude trim position substantially. This meant that trim settings for each motor had to consistently be adjusted even when nothing obvious changed on the physical quadrotor. It was not uncommon for trim biases to change from  $+5^\circ$  to  $-5^\circ$  pitch trim angle during one battery set’s lifespan. These changes had to be discovered empirically by observing the behavior of the system and tediously tuning

out such biases. Often, by the time the biases were compensated for, they would change or the set of batteries would be exhausted.

Second, the quadrotor responded unpredictably to different sets of batteries. All of the batteries used on the system were pairs of three-cell Lithium-Polymer batteries with power ratings of 2100 mAh and nominal voltages of 11.1 V. However, some brands had different qualities of current consumption capability than others, and even within batteries of the same brand, the quadrotor seemed to behave differently when a new set of batteries were installed. During the time a battery drained from 12 V to 9.5 V, the altitude controller required frequent adjustment. The final solution to this problem was to set an upper limit of PWM value which the altitude controller could command and adjust it according to the battery voltage. It is admitted that this is a non-optimal solution, but it allowed results to be collected.

These first two problems have to do with mechanical and electrical characteristics of the system. But, they slowed testing progress and made it very difficult to complete additional tests with vision onboard. In addition to these problems caused by physical characteristics of the vehicle platform, many problems contributed to difficulties in using onboard vision. Some of these problems were solved while others remain troublesome.

One problem imposed by computer vision was the tuning of the web camera. Settings such as white balancing, shutter speed, and gain control settings can be programmed to be automatic or set to particular values through a Linux API. This meant that given different lighting conditions, a human could tune the video settings to match the needs of the scene. However, this was often a tedious procedure because the automatic settings often did a poor job of preserving edges in the images in dark lighting conditions. The ability to hand tune the camera was provided, but a set of dials had to be tuned, and even doing so did not guarantee that the vision program would find appreciable lines throughout the flight sequence. This led to making the algorithms often perform poorly, though they worked well in simulation and in slow movements of a hand-held camera.

In addition to tuning the camera, the vision algorithms themselves require some degree of tuning to perform properly. For example, the vanishing point detection algorithm using a Hough transform required four additional settings to be tuned in addition to the camera settings already mentioned. First, the Canny edge detector required an input of upper and lower thresholds. The upper threshold determined which edgels would be automatically selected and the lower threshold specified the minimum edge magnitude that could be accepted. Typical values were 150 and 50, but both needed to change depending on how well the edges showed up in the grayscale image. Then, the Hough transform required an input parameter for the peak selection threshold. Peaks in Hough space above this threshold would be considered lines in the image. This value, however, depended on how well the Canny edge image preserved true lines. If the Canny edge detector allowed short line segments to be preserved, they resulted in multiple noisy lines in the Hough accumulator. So, a tedious process resulted in which first the camera settings were adjusted to show good edges. Then, the Canny edge detector was tuned to preserve key lines without detecting too many short line segments. Then, the Hough transform was tuned to select lines that led to the vanishing point desired.

In general, a great deal of human intervention was necessary to tune the vision algorithms. During the tuning process, then, attitude and position estimates came strictly from onboard sensors and computer vision estimates were entirely rejected. Then, once computer vision estimates were determined to be somewhat reliable, EKF gains in the autopilot were adjusted to trust the vision estimates more. However, in order to reject outlying vision estimates, some simple outlier rejection schemes were used. For example, the heading estimate from vision was only trusted if it was within five degrees of the EKF estimate.

In summary, a great deal of tuning is necessary in all aspects of quadrotor flight. Trim angles, controller gains, estimator covariances, camera settings, and vision settings all had to be tuned and retuned to collect results such as those presented here. However, given adequate dedication to these tuning procedures, good results

were collected demonstrating reliable hover flight, heading estimation using computer vision, and hallway following using velocity tracking.



## Chapter 6

### Conclusion and Future Work

Pose estimation for hovering flight is a much sought-after capability which can be acquired using onboard sensors and visual guidance. Several conclusions based on the methods and results in this thesis will be discussed and recommendations for future work will be made.

#### 6.1 Conclusions

The research conducted on the quadrotor platform leads to several conclusions that will be discussed relating to pose estimation, quadrotor modeling and control, and the use of computer vision on the quadrotor platform.

First, modeling methods based on first principles were presented. The quadrotor has fast, unstable angular dynamics in the hover flight regime. Proportional-derivative control methods were demonstrated to sufficiently track angular commands. Outer loops based on position and velocity tracking were then added to maintain position in a GPS-denied environment with no external cameras. Positive flight results demonstrated the ability to hold position for extended periods of time without user intervention.

Second, a unique combination of sensors was introduced to determine attitude and position on the quadrotor. The optic flow sensor plays a key role in estimating velocity and position in this work. It has the capability of delivering reliable estimates at fast rates and has been explored as a possible sensor in a GPS-denied environment. Results demonstrating the hover flight and hallway following using the optic flow sensor are the first of their kind.

Attitude angles can be reliably estimated in the short run by integrating the angular rates from MEMS gyros, but noise on the signal leads to drift which renders the measurement unsuitable to attitude estimation in the long run. Typical methods of providing vector attitude corrections such as accelerometers and magnetometers have inherent weaknesses on hovering vehicles. Thus, an additional vector measurement is necessary to correct attitude readings for long-term flights. Two methods of using image processing to determine vanishing points in a hallway have been demonstrated. The more promising of the two uses a Hough transform to detect lines in the image and forms a histogram of the intersections to detect likely vanishing point candidates. Once the vanishing point is detected, it has been demonstrated to effectively act as a vector measurement to correct attitude estimates on the quadrotor vehicle on a test stand.

All of these elements improve the flight capabilities of quadrotor helicopters. In addition, experimentation with onboard vision systems has increased productivity and provided higher quality video data for image processing than transmitted video typically provides.

To reiterate, the overall contributions of this thesis are the following:

- The first example to our knowledge of using an optic flow sensor on a quadrotor helicopter with reliable position estimation demonstrated in position hold. This was demonstrated with a seven-minute flight and several other long flights with no pilot corrections.
- The first example to our knowledge of computing heading from vanishing points onboard a helicopter and using that estimate to maintain a desired heading in real-time.
- Several methods for determining vanishing points are used for attitude estimation. A functional vanishing point detection algorithm was developed and implemented on the quadrotor with flight results.
- A reliable code base has been developed for estimating pose and controlling quadrotors in hover.

- Methods for modeling the projection of a hallway onto the imaging plane of a monocular camera have been explored and modeled.
- Flight down a hallway has been demonstrated using velocity commands from a human pilot.
- Onboard vision has been developed as a usable sensor with real-time visualization on a remote workstation. This greatly aids computer vision algorithm development and testing.

## 6.2 Future Work

The results obtained in this work suggest future projects which could improve hover capability on the quadrotor platform. Some of the future projects might include:

1. Auto-trim capabilities: Finding trim values automatically through an optimization method would greatly speed up flight testing. Suggested methods might include recursive least-squares estimation of trim values from flight data or gradient descent.
2. Adaptive control: Altitude control presented substantial challenges due to correlations between battery voltage and motor output. This suggests that adaptive control might be used for improved altitude hold. In addition, adaptive control methods might correct the problems having to do with slight vehicle characteristic changes which demanding meticulous tuning for flight testing.
3. Improvements to optic flow sensor: The optic flow sensor only operates reliably given certain external conditions such as adequate lighting and texture quality. These needs stem primarily from the fact that a chip intended for a computer optical mouse is being used for an entirely different purpose. If this chip were designed to respond better to other lighting conditions, it's utility would greatly improve. Thus, it is suggested that research be conducted on improving the chip design itself and that a chip be designed from ground-up with the intention of being used on hovering vehicles.

4. Outdoor Testing: The optic flow sensor might be combined with GPS for very accurate position sensing outdoors. This could provide accurate velocity readings from optic flow while relying on GPS to prevent position drift.
5. Complete reliance on computer vision: Experiment has shown that the specific computer vision algorithms implemented herein require a great deal of fine-tuning in order to provide reliable estimates. Automatic tuning and thresholding could greatly improve the reliability of these algorithms.
6. Hallway Rigid-body Tracking: Formulating the hallway tracking problem differently could greatly speed up the vision algorithms and possibly lead to more reliable pose estimation. Work was initiated by the author in this area (quite late in the process, admittedly) that follows the work of Drummond, Cipolla, and Kemp ([61], [13]). This method uses a tracking algorithm rather than a global line search to track the hallway.
7. Disturbance rejection: the controllers derived in this work do not directly account for disturbances to the system such as wind. The controllers could be improved to reject such disturbances and provide more robust control.

## References

- [1] R. S. Christiansen, “Design of an Autopilot for Small Unmanned Aerial Vehicles,” Master’s thesis, Brigham Young University, August 2004. 3
- [2] N. Knoebel, “Adaptive Quaternion Control of a Miniature Tailsitter UAV,” Master’s thesis, Brigham Young University, 2007. 3
- [3] B. Mettler, T. Kanade, and M. B. Tischler, “System Identification Modeling of a Model-Scale Helicopter,” Carnegie Mellon University Robotics Institute, Tech. Rep., 2000. 8
- [4] G. Hoffman, 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. 8, 11
- [5] K. Sprague, V. Gavrillets, D. Dugai, B. Mettler, and E. Feron, “Design and Applications of an Avionics System for a Miniature Acrobatic Helicopter,” in *Digital Avionics Systems*, 2001. 8
- [6] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, “Vision-based Autonomous Landing of an Unmanned Aerial Vehicle,” *IEEE International Conference on Robotics and Automation*, May 2002. 8
- [7] J. Hintze, “Autonomous Landing of a Rotary Unmanned Aerial Vehicle in a Non-Cooperative Environment Using Machine Vision,” Master’s thesis, Brigham Young University, 2004. 8
- [8] S. Saripalli, J. Roberts, P. Corke, G. Buskey, and G. Sukhatme, “A Tale of Two Helicopters,” in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, 2003, pp. 805–810 vol.1. 8
- [9] O. Amidi, T. Kanade, and K. Fujita, “A Visual Odometer for Autonomous Helicopter Flight,” *Journal of Robotics and Autonomous Systems*, vol. 28, pp. 185 – 193, August 1999. 8
- [10] J. Kelly, S. Saripalli, and G. S. Sukhatme, “Combined Visual and Inertial Navigation for an Unmanned Aerial Vehicle,” in *Proceedings of the International Conference on Field and Service Robotics*, Jul 2007. [Online]. Available: [http://cres.usc.edu/cgi-bin/print\\_pub\\_details.pl?pubid=540](http://cres.usc.edu/cgi-bin/print_pub_details.pl?pubid=540) 8
- [11] A. Wu, E. Johnson, and A. Proctor, “Vision-Aided Inertial Navigation for Flight Control,” in *2005 AIAA Guidance, Navigation, and Control*

- Conference and Exhibit*, 2005, pp. 1–13. [Online]. Available: <http://scholar.google.com/scholar?hl=en&lr=&cluster=3514948030657029360> 9
- [12] L. Mejías, S. Saripalli, P. Campoy, and G. S. Sukhatme, “Visual Servoing of an Autonomous Helicopter in Urban Areas Using Feature Tracking,” *Journal of Field Robotics*, vol. 23, pp. 185–199, 2006. 9
  - [13] C. Kemp, “Visual Control of a Miniature Quad-Rotor Helicopter,” Ph.D. dissertation, University of Cambridge, February 2006. 9, 58, 59, 98, 130
  - [14] A. Tayebi and S. McGilvray, “Attitude Stabilization of a Four-rotor Aerial Robot,” in *43rd IEEE Conference on Decision and Control*, 2004. 9
  - [15] P. McKerrow, “Modelling the Draganflyer Four-rotor Helicopter,” in *IEEE International Conference on Robotics and Automation*, 2004. 9
  - [16] E. Altug, J. P. Ostrowski, and R. Mahony, “Control of a Quadrotor Helicopter Using Visual Feedback,” in *IEEE International Conference on Robotics and Automation*, 2002. 10
  - [17] E. Altug, J. P. Ostrowski, and C. J. Taylor, “Quadrotor Control Using Dual Camera Visual Feedback,” in *IEEE International Conference on Robotics and Automation*, 2003. 10, 89
  - [18] S. Bouabdallah, P. Murrieri, and R. Siegwert, “Design and Control of an Indoor Micro Quadrotor,” in *IEEE International Conference on Robotics and Automation*, 2004. 10
  - [19] S. Bouabdallah, A. North, and R. Siegwert, “PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004. 10, 89
  - [20] S. Bouabdallah and R. Siegwert, “Backstepping and Sliding Mode Techniques Applied to an Indoor Micro Quadrotor,” in *IEEE Conference on Robotics and Automation*, 2005. 10
  - [21] J. F. Roberts, T. S. Sterling, J.-C. Zufferey, and D. Floreano, “Quadrotor Using Minimal Sensing for Autonomous Indoor Flight,” in *European Micro Air Vehicle Conference and Flight Competition*, 2007. 10
  - [22] G. M. Hoffman, H. Huang, S. L. Waslander, and C. J. Tomlin, “Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment,” in *AIAA Guidance, Navigation, and Control Conference*, 2007. 11
  - [23] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, “Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery,” in *AIAA Guidance, Navigation, and Control Conference*, 2006. 11

- [24] R. W. Beard, “Quadrotor Dynamics and Control,” Brigham Young University, Tech. Rep., 2008. [Online]. Available: <https://dspace.byu.edu/handle/1877/62412>, 16
- [25] Gumstix, “Computing by Gumstix,” April 2008, <http://www.gumstix.com/>. 27
- [26] Via, “Via Pico-ITX Mainboard Form Factor,” April 2008, <http://www.via.com.tw/en/initiatives/spearhead/pico-itx/>. 27
- [27] Procerus Technologies, “Procerus Technologies: Fly Light With the World’s Smallest UAV Autopilot,” 2006, <http://www.procerusuav.com>. 28
- [28] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, “Autonomous Vehicle Technologies for Small Fixed Wing UAVs,” *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, Jan 2005. 28
- [29] Intel Corporation, “Open source computer vision library,” August 2006, <http://www.intel.com/technology/computing/opencv>. 30, 55, 64, 77
- [30] Golem5, “EmbedCV - An Embeddable Computer Vision Library,” April 2008, <http://golem5.org/embedcv/>. 30
- [31] Via, “VIA Networking VNT6656G6A40 54 MBps Wireless USB Module,” April 2008, <http://www.logicsupply.com/pub/docs/VT6656usb.pdf>. 30
- [32] Nomachine, “Nomachine: The Desktop Virtualization Company,” April 2008, <http://www.nomachine.com/>. 30
- [33] Avago Technologies, “ADNS-3080 High Performance Optical Mouse Sensor,” Specifications Sheet, 2005, data Sheet. 34
- [34] S. R. Griffiths, “Unmanned Air Vehicle Remote Terrain Navigation,” Master’s thesis, Brigham Young University, 2006. 35
- [35] G. L. Barrows and C. Neely, “Mixed-Mode VLSI Optic Flow Sensors For In-Flight Control of a Micro Air Vehicle,” in *Critical Technologies for the Future of Computing, SPIE*, 2000. 35
- [36] D. B. Barber, S. R. Griffiths, T. W. McLain, and R. W. Beard, “Autonomous Landing of Miniature Aerial Vehicles,” in *AIAA Guidance, Navigation, and Control Conference*, 2005. 35
- [37] A. F. Rodriguez, E. Andersen, J. M. Bradley, and C. N. Taylor, “Wind Estimation Using an Optical Flow Sensor on a Miniature Air Vehicle,” in *AIAA Guidance Navigation and Control Conference*, 2007. 35, 37
- [38] N. Franceschini, F. Ruffier, and J. Serres, “A Bio-Inspired Flying Robot Sheds Light on Insect Piloting Abilities,” *Current Biology*, vol. 17, pp. 329–335, 2007. 35

- [39] J.-C. Zufferey and D. Floreano, “Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft,” in *IEEE Transactions on Robotics*, 2006. 35
- [40] F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*. John Wiley & sons, 1986. 43
- [41] F. Chaumette and S. Hutchinson, “Visual Servo Control Part I: Basic Approaches,” *IEEE Robotics & Automation Magazine*, pp. 82–90, 2006. 54
- [42] —, “Visual Servo Control Part II: Advanced Approaches,” *IEEE Robotics & Automation Magazine*, pp. 109–117, 2007. 54
- [43] T. Kanade, O. Amidi, and Q. Ke, “Real-time and 3D Vision for Autonomous Small and Micro Air Vehicles,” in *2004 IEEE Conference on Decision and Control*, vol. 2, 2004, pp. 1655–1662. 54
- [44] D. Nister, O. Naroditsky, and J. Bergen, “Visual Odometry,” in *2004 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004. 54
- [45] H. Durrant-Whyte and T. Bailey, “Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms,” *IEEE Robotics and Automation Magazine*, vol. 13, pp. 99–110, Jun 2006. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1678144](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1678144) 54
- [46] T. Bailey and H. Durrant-Whyte, “Simultaneous Localisation and Mapping (SLAM): Part II State of the Art,” *Robotics and Automation Magazine*, Sep 2006. [Online]. Available: <http://scholar.google.com/scholar?hl=en&lr=&cluster=6188288591783386436> 54
- [47] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Thomson Learning, 2008. 55, 60, 61
- [48] P. V. C. Hough, “Method and Means for Recognizing Complex Patterns,” Patent U.S. Patent 3,069,654, December, 1962. 55
- [49] J. B. Burns, A. R. Hanson, and E. M. Riseman, “Extracting Straight Lines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. Pami-8, pp. 425–455, 1986. 55, 64
- [50] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 55, 71, 121
- [51] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2004. 55, 126
- [52] S. T. Barnard, “Interpreting Perspective Images,” *Artificial Intelligence*, vol. 21, pp. 435–462, 1983. 56

- [53] M. J. Magee and J. K. Aggarwal, "Determining Vanishing Points from Perspective Images," *Computer Vision, Graphics, and Image Processing*, vol. 26, pp. 256–267, 1984. 56, 70
- [54] R. Schuster, N. Ansari, and A. Bani-Hashemi, "Steering a Robot with Vanishing Points," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 491–498, 1993. 56
- [55] T. Tuytelaars, M. Proesman, and L. V. Gool, "The Cascaded Hough Transform," in *International Conference on Image Processing*, 1997. 57
- [56] P. Gamba, A. Mecocci, and U. Salvatore, "Vanishing Point Detection by a Voting Scheme," in *International Conference on Image Processing*, 1996. 57
- [57] H. G. Baltzakis and P. E. Trahanias, "VPLF Method for Vanishing Point Computation," *Image and Vision Computing*, vol. 19, pp. 393–400, 2001. 57
- [58] G. McLean and D. Kotturi, "Vanishing Point Detection by Line Clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 11, pp. 1090–1095, 1995. 57
- [59] C.-P. Lu, G. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 6, pp. 610–622, 2000. 57
- [60] P. Wunsch and G. Hirzinger, "Real-time Visual Tracking of 3-D Objects With Dynamic Handling of Occlusion," in *IEEE International Conference on Robotics and Automation*, 1997. 58
- [61] T. Drummond and R. Cipolla, "Real-Time Visual Tracking of Complex Structures," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 24, pp. 932–946, 2002. 58, 59, 98, 129
- [62] G. Desouza and A. Kak, "Vision for Mobile Robot Navigation: A Survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 2, pp. 237–267, 2002. 58, 59
- [63] A. Kosaka and A. Kak, "Fast Vision-Guided Mobile Robot Navigation using Model-based Reasoning and Prediction of Uncertainties," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992. 58
- [64] H. P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. dissertation, Stanford University, 1980. 59
- [65] B. Call, "Obstacle Avoidance for Unmanned Air Vehicles," Master's thesis, Brigham Young University, December 2006. 59
- [66] J. Canny, "Finding Edges and Lines in Images," Massachusetts Institute of Technology, Tech. Rep., 1983. 63

- [67] J. W. Then, “Bifilar Pendulum—An Experimental Study for the Advanced Laboratory,” *American Journal of Physics*, vol. 33, pp. 545–547, 1965. 107
- [68] W. Johnson, *Helicopter Theory*. Princeton University Press, 1980. 114
- [69] A. R. S. Bramwell, G. Done, and D. Balmford, *Bramwell’s Helicopter Dynamics*, second edition ed. American Institute of Aeronautics and Astronautics, Inc, 2001. 114
- [70] P. Castillo, R. Lozano, and A. E. Dzul, *Modelling and Control of Mini-Flying Machines*. Springer-Verlag, 2005. 115
- [71] A. Dzul, T. Hamel, and R. Lozano, “Modeling and Nonlinear Control for a Coaxial Helicopter,” in *IEEE SMC*, 2002. 115
- [72] B. Mettler, *Identification Model and Characteristics of Miniature Rotorcraft*. Kluwer Academic Publishers, 2003. 119

## **Appendix A**

### **Coaxial Helicopter Dynamics**

#### **A.1 Coaxial Helicopter Platform**

The coaxial helicopter was designed around a Walkera 53#1 coaxial hobby helicopter. The helicopter was heavily modified, leaving only the main shaft and servo assembly as original. Layers of G10 composite material were added below the shaft base to hold all the necessary electronics. A spherical carbon-fiber shroud was built around the whole unit to protect the brittle blades.

The coaxial helicopter provides an easy learning experience for pilots unversed in helicopter flight. Coaxial helicopters have a tendency to remain upright due to the opposing motion of the two rotor assemblies. These helicopters provided a good platform for testing sensors, but presented challenges due to vibration, sensitivity to mechanical changes, and a lack of weight capability. Despite these difficulties, this platform presented a welcome challenge and a good learning experience.

##### **A.1.1 Quadrotor Vehicle Platform**

In contrast to the coaxial helicopter, the quadrotor vehicle demands some level of motion damping in order to fly. But, with relatively simple estimation and control, the quadrotor is a good hovering platform due to its decoupled axes and forgiving weight capacity. The quadrotor is able to carry a great deal of sensing equipment, at the expense of requiring significant battery power which shortens its flight duration to about 10 minutes per pair of batteries. However, as a vehicle testbed, the quadrotor is

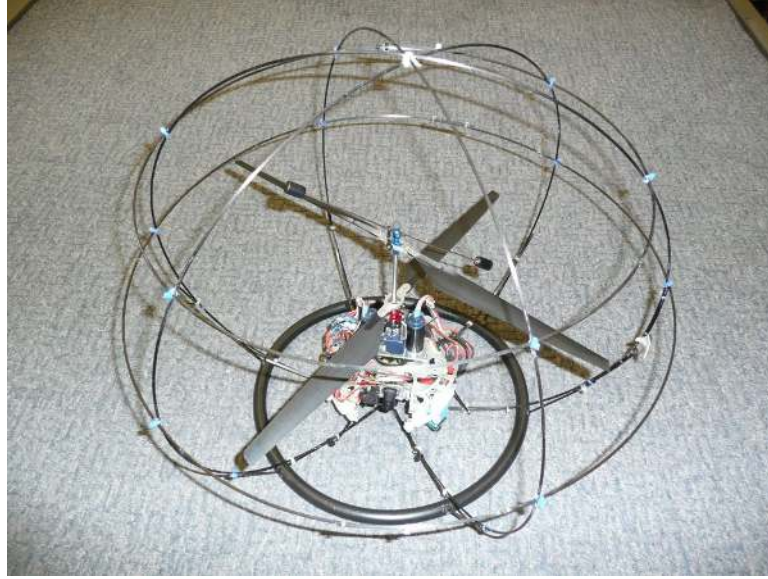


Figure A.1: **Coaxial Helicopter Image**

an excellent choice and is rather forgiving of minor crashes. The quadrotor helicopter pictured in Figure 1.1 is equipped with a four Axi 2212/26 brushless motors and four accompanying speed controllers. Each motor is capable of providing over a pound of thrust, but the nominal hover throttle required is near 50%. The array of sensors onboard will be discussed in section 3.2.

## A.2 Coaxial Helicopter Dynamics

The behavior of the rotor system on helicopters adds additional degrees of freedom to the system. To better understand how rotor dynamics affect the flight characteristics and dynamic behavior of the coaxial helicopter, a model has been developed and tested in Matlab simulation. This model is presented along with the specific properties of the modified coaxial helicopter built and tested in the MAGICC Lab.

### A.2.1 Physical Properties of the Helicopter

Some of the physical properties of the specific helicopter which we have chosen to model are listed in table A.1.

Table A.1: **Coaxial Helicopter Physical Parameters:** Some key parameters measured on the coaxial helicopter

Property	Value	Description
$b$	2	Number of blades per rotor
$\bar{c}$	2.3 cm (0.969 in)	Average blade chord
$R$	22.4 cm (8.875 in)	Blade radius from shaft to tip
$m$	0.805 kg	Total mass of vehicle
$I_{xx}$	0.00892446 $kg \cdot m^2$	Mass moment of inertia about body x-axis
$I_{yy}$	0.00801767 $kg \cdot m^2$	Mass moment of inertia about body y-axis
$I_{zz}$	0.00581 $kg \cdot m^2$	Mass moment of inertia about body z-axis
$R_{fb}$	10.4 cm	Radius of flybar
$I_{fb}$	0.0002366 $kg \cdot m^2$	Moment of inertia of flybar about its hinge

The moments of inertia of the vehicle itself were measured using a bifilar pendulum experiment (see [67]). The off-axis terms were neglected. The moment of inertia of the flybar was calculated from the geometry.

Given accurate measurements of the mass and moments of inertia of the vehicle, we have all the necessary properties to describe the rigid-body motion of the helicopter, assuming that the fuselage does not introduce any significant aerodynamic forces near hover. The forces and moments acting on the helicopter then are primarily due to gravitational forces and the thrust vectors from the two rotors. Both rotors are necessary to provide lift and both influence the translational and rotational motion of the helicopter.

### A.2.2 Rotor Reference Frames and Notation

Several Planes come into play when describing rotor motion. First, the control plane (CP) is the plane through which the swashplate moves and thus induces cyclic

feathering motion of the blades. The Hub Plane (HP) is simply the plane perpendicular to the shaft. The Tip-Path-Plane is the plane in which the tips of the blade move. All of these planes are pictured in Figure A.2.

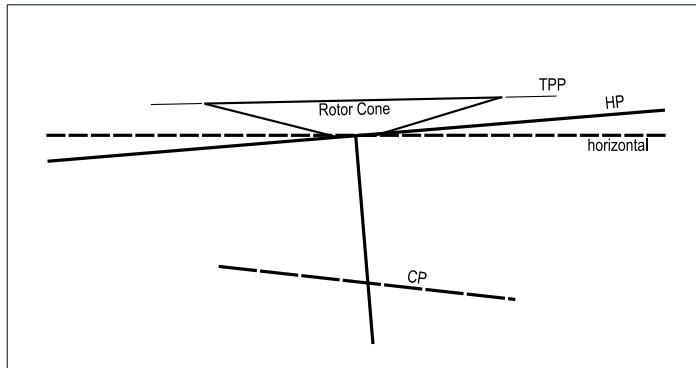


Figure A.2: **Rotor reference planes:** Control Plane (CP), Hub Plane (HP), and Tip-Path-Plane (TPP)

To describe the motion of helicopter rotors and their effective thrust forces and moments, it is necessary to describe the notation and reference frames which will be used. For each rotor, we will define two new reference frames called the rotor hub plane and the rotor tip-path plane. The azimuth angle of each blade will be denoted by  $\psi_x$  where the subscript denotes which rotor we refer to, either  $\psi_b$  for the bottom rotor or  $\psi_t$  for the top rotor. In general, we will refer to  $\psi$  as a general variable denoting the azimuth of a blade from the tail of the aircraft with positive rotation being counter-clockwise as shown in Figure A.3. We will later make a change of variable from azimuth angle to time (using  $\Omega = \dot{\psi}$ ). Then, to account for clockwise rotating rotors (like the bottom rotor on our coaxial helicopter), we will use a variable  $\lambda$  which will be equal to one for counter-clockwise rotors and negative one for clockwise rotors.

The hub plane is defined as the plane which remains perpendicular to the helicopter shaft with its axes centered at the rotor-shaft hinge and with directions parallel to body frame axes. The blade goes through complex motion as it rotates around the shaft at fast angular speeds ( $\text{RPM} > 2500$ ). Blade hubs come in many

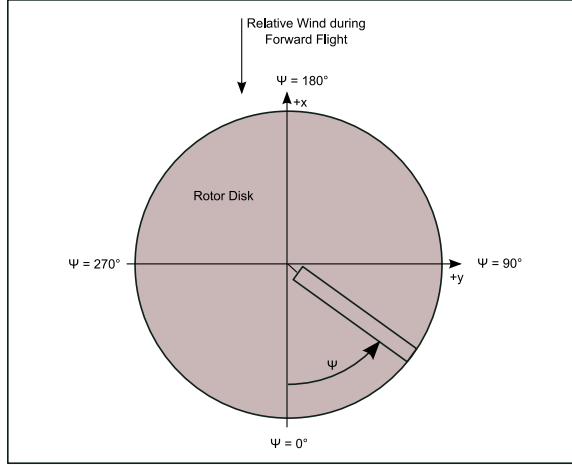


Figure A.3: **Azimuth Direction Definition:** Azimuth direction notation with respect to vehicle frame.

varieties, and generally three hinges are built into the support for a single blade: the feathering hinge, the flapping hinge, and the lead-lag hinge. These correspond to the three degrees of freedom of the blade. These are depicted in Figure A.4.

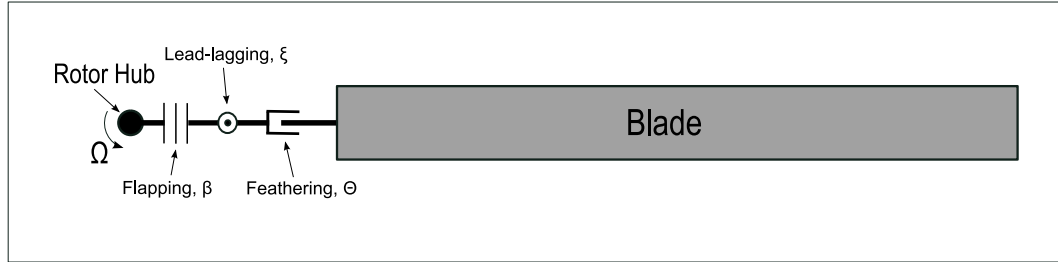


Figure A.4: **Additional degrees of freedom of a moving blade.**

The coaxial helicopter we use has a hingeless blade. This means that there is no flapping hinge, yet the blade will still flap through elastic bending. It will later be shown that on these particular helicopters, the blades are designed with an equivalent hinge offset which results in approximately  $45^\circ$  of phase lag between a control plane deflection (motion of the swashplate) and a subsequent tilt of the tip-path-plane. It is typical for manufacturers to then pre-phase the control plane by  $45^\circ$  to account

for this phase lag in order to create flapping moments as desired by control plane deflections.

The  $45^\circ$  offset results in some strange nonlinear behavior in the rotor system. In summary, there are cross-coupling effects which occur due to angular rates and angular accelerations. A full dynamic model would need to take all of these cross-coupling effects into account. My current understanding is limited to cross-coupling due to effective hinge offset.

### **A.2.3 Blade Element Method**

Blade theory is the common method for integrating aerodynamic forces and moments produced by a blade as it sweeps through a revolution. Elementary blade element methods are used to solve for average thrust produced as a function of rotor RPM. Since there are two thrust-producing rotors with independent speed control, this method will be used independently on the two rotors to solve for lift and drag components in the body-frame of the vehicle.

### **A.2.4 Lift and Drag Calculations**

The lower blade is simple to model because it is similar to regular blades on single-axis helicopters except that it is not subjected to collective pitch changes. The blades on our helicopter are built with a permanent angle of attack which is greatest where the chord is largest, a short distance from the blade root. The blades then slope down to a smaller chord and smaller angle of attack near the tips. Rather than model the blade twist, we have estimated the blade geometry as being of a constant angle-of-attack with a shortened effective radius due to losses at the blade tips and blade root. The lift and drag are calculated by integrating incremental lift and drag over small blade elements. The losses at the blade tips are included by using an effective radius,  $R_{eff}$ , which was found to be  $0.97 * R$ . The root losses are included by changing the

lower limit of integration to an initial  $r$  value which was estimated to be  $0.1 * R$ . The total thrust is determined by integrating the increment of lift according to:

$$\Delta L = \frac{1}{2}\rho(\Omega r)^2 a \bar{c} \Delta r \quad (\text{A.1})$$

and the total thrust is therefore found to be a function of rotor speed (in rad/s) according to

$$L = \frac{1}{2}\rho\Omega^2 a \bar{c} \alpha \int_{r_0}^{R_{eff}} r^2 dr. \quad (\text{A.2})$$

In the above equations,  $\rho$  is the local air density,  $a$  is the lift curve slope (a good general value of 6 is used), and  $\alpha$  is the local angle of attack. It is true that this model does not take into account rotor inflow or changes in angle of attack, but constant inflow is generally a good assumption near hover flight (it is????) and the angle of attack is instead taken into account by later modeling the tilt of the thrust vector created by dynamic flapping.

The incremental torque produced by a rotor is composed of induced drag and profile drag according to

$$\Delta Q = r(\Delta L \phi + \Delta D_0) \quad (\text{A.3})$$

where we assume that induced drag is negligible and that all effective drag is due to profile drag.

The corresponding drag integral for a rotor blade is

$$\Delta D = \frac{1}{2}\rho(\Omega r)^2 c_d \bar{c} \Delta r. \quad (\text{A.4})$$

### A.2.5 Flybar Modeling

The flybar or stabilizer bar acts like a gyroscope and as an effective swashplate with respect to the top rotor. The bottom rotor is attached to a swashplate and there-

fore its feathering and flapping motion are strongly governed by the planar motion of the swashplate. However, due to the small size of the blades and their small amount of inertia (represented by a very small moment of inertia in comparison to full-size helicopters), the top rotor would behave uncontrollably if not directly attached to the flybar. This was verified by running the rotor with the flybar detached and with only the top rotor installed. The motion produced was erratic and produced vibrations too strong to complete the test. The flybar, then, is a completely necessary component of the system due to the small size of the helicopter. Since there is no swashplate attached to the top rotor, so, the flybar is designed to act as an effective swashplate, though not directly actuated by a pilot.

$$\dot{c} = -\frac{c}{\tau_s} - q \quad (\text{A.5})$$

$$\dot{d} = -\frac{d}{\tau_s} - p \quad (\text{A.6})$$

The flybar is simply a shaft with two equal sections hinged about its middle on a hinge which spins with the top rotor shaft. On each end are adjustable weights which create a large amount of rotational inertia in the plane perpendicular to the shaft. If the shaft tilts with respect to the flybar's plane of motion, the flybar tends to continue to stay in plane. This means that it opposes body-frame angles (and angular rates). Testing demonstrated that the flybar has a slow restoring moment which will eventually return the flybar to be perpendicular to the shaft after a given amount of time. This behavior is a first-order system defined by a time constant,  $\tau_s$  (which has a value of about 2.5 seconds for the flybar on our system). If left alone (not attached to a rotor), the equation of motion for the flybar would include two new states, denoted by  $c$  and  $d$ , the respective pitch and roll of flybar tip-path-plane.

The variable  $c$  is defined as positive pitch up and  $d$  is defined as positive roll right to correspond with typical pitch and roll definitions of air vehicles.

The important things to notice about this system are that the time constant is very slow and that the flybar tends to stay in its previous state in an inertial sense which provides a stabilizing effect on the top rotor to which it is attached.

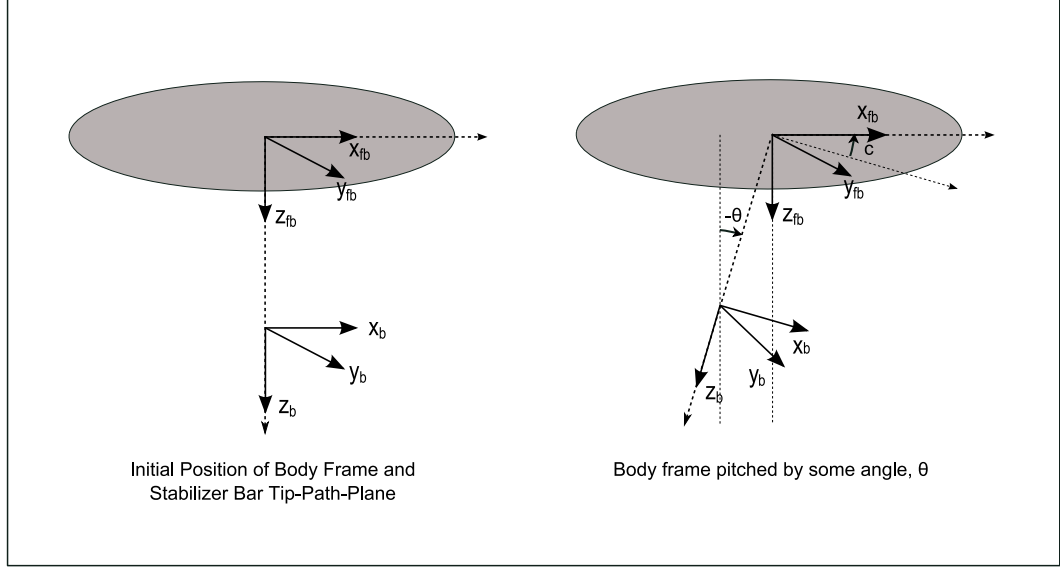


Figure A.5: **Gyroscopic Flybar Behavior:** This diagram illustrates the ideal behavior of a gyroscopic flybar. If no friction or aerodynamics were present, the flybar would continue to spin in the plane in which it was originally rotated.

The flybar is an important part of the overall system because it provides damping to the angular rates of the rigid body and a gyroscopic correction to the top blade tending to keep the helicopter pointed opposite of the gravity vector. The flybar (also referred to as a control rotor or stabilizer bar) is a rotating system with additional degrees of freedom. The primary degree of freedom that we are concerned with is the flybar flapping angle,  $\beta_{fb}$ , which is measured relative to the flybar hub plane. The coaxial helicopter's flybar is a teetering rotor, meaning that it is hinged directly at the shaft with no flapping hinge offset. It is also designed to minimize aerodynamic effects so that its behavior is primarily due to inertia, making it behave

very much like a gyroscope. Conventional helicopter literature ([68],[69]) provide equations of motion of a teetering rotor in terms of  $\beta_{fb}$  as a function of azimuth angle (defined in Figure A.3). The flybar flapping angle can be represented as a periodic function as

$$\beta_{fb} = -\beta_{1c,fb} \cdot \cos(\psi_{fb}) - \beta_{1s,fb} \cdot \sin(\psi_{fb}) - \dots \quad (\text{A.7})$$

where we will drop all higher order terms above the first harmonic. The flybar is constrained from coning, so there is no constant term in the series. Moments around the teetering hinge include inertial moments, aerodynamic moments (denoted  $M_{a,fb}$ ), gyroscopic moments (denoted  $M_{gyro,fb}$ ) and moments due to the bearings or friction at the hinge. For the moment we will neglect the bearing or friction forces and assume that the flybar does not provide any noticeable lift forces (the flybar on most coaxial helicopters are not airfoils in shape so no net forces are present perpendicular to the relative wind). Solving for moment equilibrium about the teetering hinge we get (insert equation here).

### A.2.6 Flapping Equations of motion

A blade spinning about the rotor shaft induces several forces. First, the fast rotation induces centrifugal forces which pull the blades away from the shaft. Second, lift forces make the blade want to flap out of the plane of rotation. Inertia opposes the flapping motion induced by the aerodynamic forces. These forces are pictured in Figure A.6.

### A.2.7 Specific Modeling for a Coaxial Helicopter with One Actuated Rotor

Coaxial helicopters are becoming a very popular hobby toy due to their inherent stability which makes them simple to learn how to pilot. Where conventional

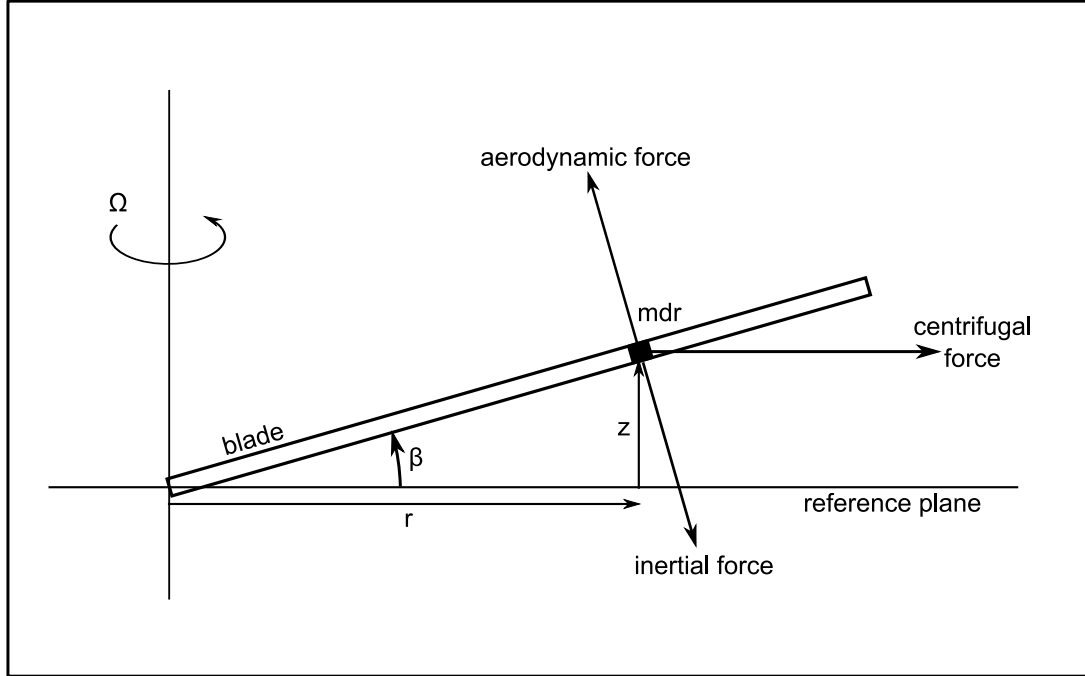


Figure A.6: **Rotor Blade Flapping Moments:** Forces acting on a blade element include aerodynamic forces, centrifugal forces and inertial forces.

single-rotor helicopters tend to require a skilled pilot, coaxial helicopters are much easier to keep upright. However, though their stability is augmented by the coaxial rotation and the stabilizer bar connected to the top rotor blade, they are sluggish in response to pilot inputs and therefore present some controllability challenges. Typical models of coaxial helicopters, like that proposed by [70], assume that flapping angles can be commanded directly and that these result in direct forces and moments on their expected axes (i.e. lateral cyclic pitch results directly in a pure lateral moment, etc.). From our experiments, we have determined that more complex dynamics are commonplace due to the construction of the rotor mechanism on these hobby helicopters. The primary concern is that the swashplate is connected only to the bottom rotor, leaving the top rotor to produce forces and moments which are not directly implied by the cyclic commands. In addition, models such as that proposed by Dzul, Hamel, and Lozano in [71] do not explicitly model the stabilizer bar and assume that the top rotor is actuated by some swashplate mechanism. Unfortunately, we have

found that this model is insufficient to account for the transient behavior of the top rotor which has a great effect on flight handling qualities and position control.

***Modeling Top Blade and Flybar Interaction*** The linkage tying the top blade and flybar together, and the fact that neither is directly actuated by the pilot presents a significant challenge to full authority control. Also, since a large portion of the forces and moments acting on the rigid body are a result of the thrust vector from the top rotor, it is very important to accurately model the dynamic motion which results from the coupled system.

Viewing the helicopter from above, there is an angle offset between the location where the linkage connecting the flybar and top rotor connects to the feathering axis of the top rotor and the flapping axis of the flybar. This angle, denoted  $\theta_{fb}$ , is easily measured to be  $45^\circ$  on the helicopter we have modified. This angle is important because it represents how the angles  $a_t$ ,  $b_t$ ,  $c$ , and  $d$  interrelate. As previously discussed, the flybar behaves similar to a gyroscope with a small moment due to bearing and aerodynamic friction which eventually makes the flybar align itself perpendicular to the shaft. Thus, if left alone, it would behave as a first order system with a time constant,  $\tau_{fb}$ . The top blade responds primarily to two moments: the moment due to the rigid hinge, which acts a like a strong spring making the blade quickly align itself perpendicular to the shaft, and the moment imposed by the flybar which steadies the flapping motion in a manner analogous to a swashplate.

The flybar is not left alone, obviously, and we must model the moment that the top blade imposes on the flybar. The flapping cross coupling equations below attempt to represent both of these interactions. They are set up in a sequential manner in order to allow a differential equation solver to solve them numerically, as is the case in the Matlab/Simulink simulator.

$$\dot{c}_1 = -\frac{c}{\tau_{fb}} - q \quad (\text{A.8})$$

$$\dot{d}_1 = -\frac{d}{\tau_{fb}} - p \quad (\text{A.9})$$

$$\dot{a}_t = \alpha\left(-\frac{a}{\tau_t} - q\right) + (1 - \alpha)\left(\frac{1}{\tau_t}\right)(\cos(\theta_{fb})\dot{c} - \sin(\theta_{fb})\dot{d}) \quad (\text{A.10})$$

$$\dot{b}_t = \alpha\left(-\frac{b}{\tau_b} - p\right) + (1 - \alpha)\left(\frac{1}{\tau_t}\right)(\sin(\theta_{fb})\dot{c} + \cos(\theta_{fb})\dot{d}) \quad (\text{A.11})$$

$$\dot{c} = \dot{c}_1 + K_{fb}\left(\frac{1}{\tau_{fb}}\right)(\cos(\theta_{fb})\dot{a} + \sin(\theta_{fb})\dot{b}) \quad (\text{A.12})$$

$$\dot{d} = \dot{d}_1 + K_{fb}\left(\frac{1}{\tau_{fb}}\right)(-\sin(\theta_{fb})\dot{a} + \cos(\theta_{fb})\dot{b}) \quad (\text{A.13})$$

There are several important terms which have physical meaning and need to be tuned in order to properly represent the true system. The time constants of both systems are very important. The blade's time constant ( $\tau_t$ ) is much faster than the flybar's time constant ( $\tau_{fb}$ ) due to the spring action of the hingeless blade. This means that the system will behave as a hybrid dynamic system responding to the pushing and pulling of the two systems on each other through the linkage.

These equations help to explain the behavior of the system in flight. When the rotor is spun up before and during takeoff, the flybar is naturally oriented horizontally with respect to the ground. Once the helicopter lifts off, it is stabilized by the gyroscopic tendencies of the flybar to stay aligned horizontally. However, once a pilot introduces control moments through cyclic actuation of the lower blade system, the helicopter induces a rolling and pitching motion which affects the top blade. The top blade thus aligns itself perpendicular to the shaft and exerts forces which slowly pull the flybar out of its horizontal inertial plane into an off-axis plane. There is no moment naturally counteracting the flybar's motion. This induces translational motion and couples the flapping angles of the top blade into off-axis moments. A pilot's main goal, then, is to restabilize the aircraft by realigning the flybar horizontally parallel

to the ground, thus making the thrust vector directly oppose gravitational forces. However, even if the pilot can succeed in doing so, the helicopter has by then induced inertial translational accelerations which carry it away indefinitely (if not for drag forces acting on the rotor blades). So the pilot has an additional challenge: predict the accelerations which will be induced by the rigid body motion and directly counter them before (or while) stabilizing the flybar. This proves to be a difficult task due to the great deal of cross-coupling introduced by the  $45^\circ$  offset between the top blade and flybar.

### A.2.8 Coupling Rotors with Rigid Body

Forces and moments come primarily from the thrust vector and the gravity vector when the helicopter is near hover. Drag also enters in, but we typically model drag as being negligible in the hover state (is this a good assumption?). The forces from the motors are found rather easily using the flapping states. For each rotor, we have:

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \frac{T}{\sqrt{1 - \sin^2 a \sin^2 b}} \begin{bmatrix} -\sin a \cos b \\ \sin b \cos a \\ -\cos a \cos b \end{bmatrix} \quad (\text{A.14})$$

Since the thrust vector tilts with respect to the shaft, moments are produced about the center of gravity of the vehicle. In addition, the hingeless rotor acts like a spring wanting the restore the rotors to be perpendicular to the shaft. This also exerts forces on the shaft which result in moments about the center of gravity with a moment arm of length equal to the distance  $l$ , between the rotor hub and the C.G.

These moments are summed up as follows:

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \frac{|T|}{\sqrt{1 - \sin^2 a \sin^2 b}} \begin{bmatrix} -l_y \cos a \cos b - l_z \sin b \cos a \\ l_x \cos a \cos b - l_z \sin a \cos b \\ l_y \sin a \cos b + l_x \sin b \cos a \end{bmatrix} \quad (\text{A.15})$$

Mettler [72] makes small angle assumptions, assumes that  $l_x$  and  $l_y$  are small and takes into account the moments due to spring forces which results in:

$$L = (l_z T + k_\beta) b \quad (\text{A.16})$$

$$M = (l_z T + k_\beta) a \quad (\text{A.17})$$

The moment about the z-axis due to thrust and hub stiffness should be zero. On the coaxial helicopter, we find that a trim value can be found which results in little or no yawing motion in hover flight as long as the rotors stay at constant throttle. When throttle is modulated, moments due to rotor drag become more significant.



## Appendix B

### Perspective Geometry and Hallway Simulation

A background understanding of homogeneous points, rigid-body projections, camera models, etc. are important to understand in using computer vision in a hallway. For those experienced in computer vision, these topics will be well understood, while for those less familiar, this appendix will be important to read first. Even for those with a general knowledge of perspective geometry may gain by reading this section simply to familiarize themselves with the terminology and notation used in this thesis.

First, perspective geometry is described using Euclidean transformation matrices and homogeneous coordinates. The duality of points and lines and reasons for using homogeneous coordinates are briefly described. Rigid-body transformations as well as camera projections are then covered. An application of using perspective transforms to predict motion of the hallway is presented using a Matlab simulation.

#### B.1 Perspective Geometry

Human interpretation of perspective images is something to marvel at. The human mind seemingly converts stereoscopic images from two eyes into a viewable world in which humans are adept at operating. However, computers viewing the world through a monocular camera must be taught to interpret the projective geometry that results. As Hartley and Zisserman [50] and other computer vision texts note, the only thing we can generally say that is preserved in a projective transformation

is straightness. Angles, distances, etc, are not necessarily preserved. This fact leads to the general system of incrementing the Euclidean space  $\mathbb{R}^n$  by one dimension to create a projective space  $\mathbb{P}^n$  by representing points as homogeneous vectors. This section discusses a way of representing points in a homogeneous fashion, how to apply rigid-body transformations to these points, and how to mathematically represent the perspective projection occurring in a camera

### B.1.1 Homogeneous Representation of Points and Lines

In a Euclidean 2-space, we represent a point as a coordinate such as  $(x, y)$ . To make this a projective space, we simply append a third coordinate to represent the same point as  $(x, y, 1)$ . This allows us to represent regular points as we always have, but it also allows us to represent infinite points such as the intersections of parallel lines. These lines will simply be denoted as an ordered triple where the third coordinate is equal to 0. The convenience of working with homogeneous coordinates quickly becomes apparent when finding lines as intersections of points, finding vanishing points as intersections of lines, etc. These operations become simple cross-products, greatly reducing computational complexity (and making the math easier to follow). In fact, this simple fact (that the intersection of two points is a line and the intersection of two lines is a point) leads to a general duality of lines and points. This comes into play when we discuss localization methods, like those implemented in SLAM. Many SLAM methods use feature point tracking, but the transition to line-tracking is rather simple due to the duality in the homogeneous representation.

Following is a description of how to represent rigid-body transformations which include both rotations and translations using homogeneous three-space. Then, we will discuss how to compute projections of points or lines from a 3-D representation to a 2-D representation which will allow us to see how points and lines are mapped onto an imaging plane. In general, homogeneous points will be represented using a bar

notation so that a nonhomogeneous point,  $P_b$ , will have the homogeneous equivalent  $\bar{P}_b$ .

### B.1.2 Rigid-body Transformations in Homogeneous Coordinates

Rigid-body representations are elegantly represented using a Euclidean projection matrix,  $E$ . Any such matrix is composed of a 3x3 rotation matrix,  $R$ , and a 3x1 translation vector,  $T$  as follows:

$$E = \begin{bmatrix} R & T \\ \mathbf{0} & 1 \end{bmatrix} \quad (\text{B.1})$$

where  $R$  is an orthonormal rotation matrix such that  $RR^T = I$  and  $|R| = 1$ . Figure B.1 shows two coordinate frames, the world frame centered at the world origin,  $\mathbf{W}$ , and the body frame, centered at  $\mathbf{B}$ . To represent a world point  $P_w$  in the body frame as a point  $P_b$ , we would apply the rotation and translation as follows:

$$P_b = R_{bw}P_w + T_{bw} \quad (\text{B.2})$$

where  $R_{bw}$  is the rotation matrix which changes points from the world frame to the body frame and  $T_{bw}$  is the needed translation. This is mathematically identical to applying the Euclidean transformation  $E_{bw}$  which represents a homogeneous world point  $\bar{P}_w$  as a point in the body frame  $\bar{P}_b$ :

$$\bar{P}_b = \begin{bmatrix} P_b \\ 1 \end{bmatrix} = \begin{bmatrix} R_{bw} & T_{bw} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P_w \\ 1 \end{bmatrix} = E_{bw}\bar{P}_w. \quad (\text{B.3})$$

Often, we do not keep track of the vector  $T_{bw}$  expressed in the body frame, but, rather, we keep track of the position of the body frame relative to the world frame,

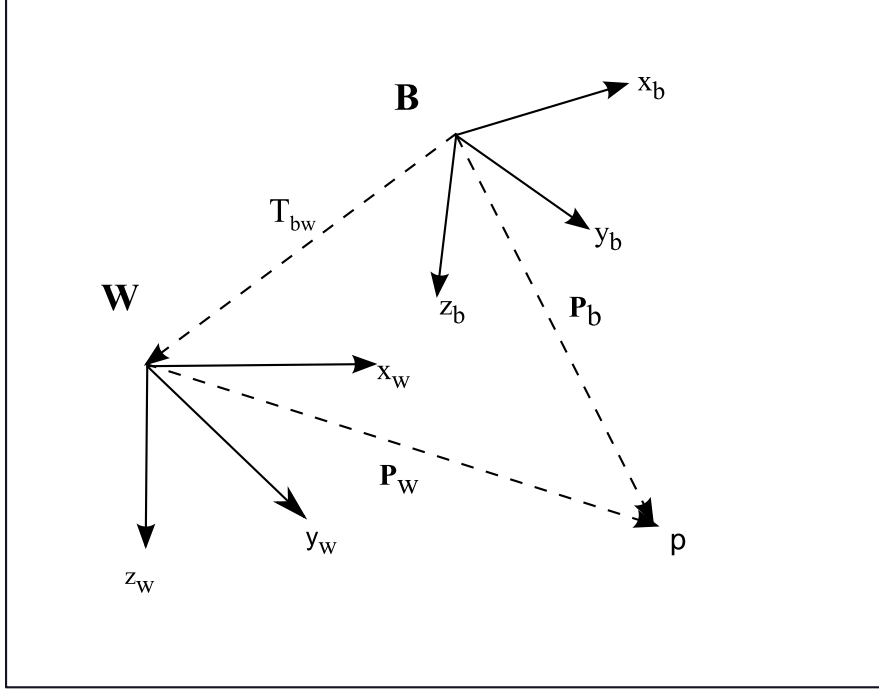


Figure B.1: **Rigid-body transformation:** A point with known coordinates in a world frame represented in the body-frame of the vehicle through a rigid-body representation as depicted.

which we denote  $C_{bw}$ . The desired vector can be found using the following rule:

$$T_{bw} = -R_{bw}C_{bw}. \quad (\text{B.4})$$

With this rather simple model for representing rigid-body motion, we can effectively calculate how to render a set of points in the world frame into the camera frame by representing the entire transformation as the product of two Euclidean transformations:  $E_{bw}$ , as already described, and  $E_{cb}$ , the projection of body frame points onto the camera frame. Thus, to go from points in the world frame to points in the camera frame, we use the following equation

$$\bar{P}_c = E_{cb}E_{bw}\bar{P}_w. \quad (\text{B.5})$$

### B.1.3 Camera Projection Model

Finally, the act of perspective projection requires knowledge of camera parameters which we represent using a camera calibration matrix,  $K$ . The camera calibration matrix effectively shifts the coordinates from world units (such as meters or feet) into pixel units and offsets the origin to the edge of the image. This allows the common representation of pixel points as an array beginning at the upper left of the screen. The camera calibration matrix is a way of representing the intrinsic parameters of the camera and is composed of the following elements:

- $s_x$ - scale factor in the x-direction representing the conversion from metric units to pixel units
- $s_y$ - scale factor in the y-direction, typically  $s_x = s_y$  which means the pixels are square
- $f$  - the camera focal length
- $c_x$ - offset of the origin in the x-direction from the center
- $c_y$ - offset of the origin in the y-direction from the center

Additional intrinsic parameters are sometimes introduced to account for angular skew, spherical distortion of the lens, etc. However, in the case of the web cameras employed on our vehicle, spherical distortion is negligible and calibration yielded no angular skew. The camera calibration matrix is then an upper-triangular matrix of the following form:

$$K = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (\text{B.6})$$

In the case of the Logitech QuickCam employed on the vehicle, the  $K$  matrix is

$$K_{logitech} = \begin{bmatrix} 395.5 & 0 & 160 \\ 0 & 395.5 & 120 \\ 0 & 0 & 1 \end{bmatrix}. \quad (\text{B.7})$$

Knowing the camera's intrinsic parameters allows us to effectively project known world points into the camera frame to predict where lines would show up in the image. We can later use this knowledge to limit our search for lines to be close to predicted lines. Homogeneous image point locations, denoted  $\bar{P}_i$ , can be represented as a three-vector computed from the camera frame point locations:

$$\bar{P}_i = K\Pi_0\bar{P}_c = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (\text{B.8})$$

where  $\Pi_0$  is an identity matrix with an extra column of zeroes which represents the perspective projection [51]. We go from a homogeneous four-vector to a homogeneous three-vector. Finally, the actual image points are found by dividing out the scale factor introduced by the homogeneous points:

$$P_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \frac{1}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (\text{B.9})$$

Combining the rigid-body transformation with the projective camera model results in the following set of operations to convert from world points to image points:

$$\bar{P}_i = K\Pi_0 E_{cb} E_{bw} \bar{P}_w \quad (\text{B.10})$$

## B.2 Hallway Projection Simulation

In order to understand the perspective projection and test the correct modeling of the camera, a model was built in Matlab which performs this projection based on a given pose of the vehicle and known transformation of the camera frame from the body frame. This projection model is based on the above transformation in Equation B.10 with the following quantities defined:

$$R_{cb} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad (\text{B.11})$$

$$T_{cb} = \begin{bmatrix} 0 \\ 0 \\ -0.10 \end{bmatrix}, \quad (\text{B.12})$$

which convert from the body frame to the camera frame. The body frame of the vehicle is defined by the position of the vehicle in the world frame and the rotation parameterized by Euler angles. The Euler angles can be used to find the rotation matrix from the body to world frame as follows:

$$R_{wb} = R_{\phi} R_{\theta} R_{\psi} \quad (\text{B.13})$$

where the intermediate rotations are defined with respect to the inertial frame, vehicle-1 frame and vehicle-2 frame and are defined as follows:

$$R_{\psi} = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (\text{B.14})$$

$$R_\theta = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix}, \quad (\text{B.15})$$

and

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix}. \quad (\text{B.16})$$

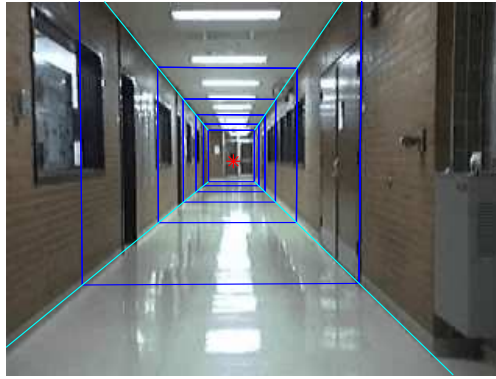
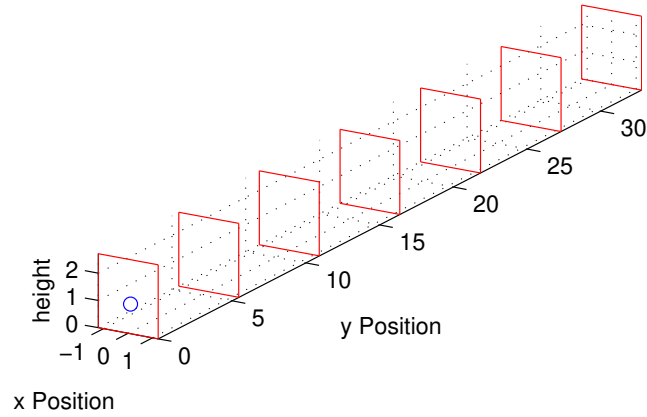
In the camera projection, we desire the rotation of the world frame with respect to the body frame,  $R_{bw}$  which is the inverse of  $R_{wb}$ , or simply the transpose due to the nature of orthogonal matrices. Performing the necessary operations, the desired matrix is written in closed form as:

$$R_{bw} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (\text{B.17})$$

where a  $c$  denotes a cosine and an  $s$  denotes a sine.

Finally, the translation of points in the world frame with respect to the body frame, denoted  $T_{bw}$ , can be found from the known location of the vehicle in the world frame,  $C_{bw}$ , according to Equation B.4. Performing these operations, we can transform a known hallway onto an image of the hallway and compare, as shown in Figure B.2.

This hallway simulation method can be extended to any rigid-body in the field-of-view of the camera. A line tracking procedure will be discussed in section B.2.1 which provides a fast algorithm for tracking the motion of a rigid-body.



$\psi: 2.000$   $\theta: -2.500$   $y: 0.09$   $z: 0.39$

Figure B.2: **Hallway Simulation:** The first diagram shows the vehicle's position with respect to the corridor. The second image shows a captured image from the camera with the projected edges overlaid in cyan.

### B.2.1 Line Tracking Method

My favorite method involves projecting a predicted pose based on prior measurements and sensor data onto the imaging plane. This rendered pose gives an initial guess of where lines will be located in the image plane. Then, a line search is performed perpendicular to the predicted lines. Using the method proposed by Drummond and Cipolla [61], a least-squares method is used to determine the most

likely combination of pose changes which would generate the observed image-space motion. This method shows great promise and has been shown by Kemp [13] to be capable of tracking 3-D corridors in real-time.