

System Identification, State Estimation, and Control of
Unmanned Aerial Robots

Caleb Chamberlain

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

Randal W. Beard, Chair
Brian Mazzeo
Timothy W. McLain

Department of Electrical and Computer Engineering
Brigham Young University
April 2011

Copyright © 2011 Caleb Chamberlain
All Rights Reserved

ABSTRACT

System Identification, State Estimation, and Control of Unmanned Aerial Robots

Caleb Chamberlain

Department of Electrical and Computer Engineering

Master of Science

This thesis describes work in a variety of topics related to aerial robotics, including system identification, state estimation, control, and path planning.

The path planners described in this thesis are used to guide a fixed-wing UAV along paths that optimize the aircraft's ability to track a ground target. Existing path planners in the literature either ignore occlusions entirely, or they have limited capability to handle different types of paths. The planners described in this thesis are novel in that they specifically account for the effect of occlusions in urban environments, and they can produce a much richer set of paths than existing planners that account for occlusions.

A 3D camera positioning system from Motion Analysis is also described in the context of state estimation, system identification, and control of small unmanned rotorcraft. Specifically, the camera positioning system is integrated inside a control architecture that allows a quadrotor helicopter to fly autonomously using truth data from the positioning system. This thesis describes the system architecture in addition to experiments in state estimation, control, and system identification.

There are subtleties involved in using accelerometers for state estimation onboard flying rotorcraft that are often ignored even by researchers well-acquainted with the UAV field. In this thesis, accelerometer-rotorcraft behavior is described in detail. The consequences of ignoring accelerometer-rotorcraft behavior are evaluated, and an observer is presented that achieves better performance by specifically modeling actual accelerometer behavior. The observer is implemented in hardware and results are presented.

Keywords: quadrotor control, path planning, aerial surveillance, system identification, quadrotor attitude estimation, quadrotor

ACKNOWLEDGMENTS

Many people contributed to the work in this thesis. John Macdonald was heavily involved in the work with the Motion Analysis system. Robert Leishman was also closely involved with the Motion Analysis research in addition to the work on system identification. Matt Argyle and Peter Niedfelt worked closely with me on the path planning research. Finally, Randy Beard helped every step of the way.

Table of Contents

List of Figures	xiii
1 Introduction	1
1.1 UAV Path Planning for Ground Target Observability	1
1.1.1 Overview	1
1.1.2 Contributions	1
1.2 Motion Capture Testbed for Indoor Aircraft Navigation and Control	2
1.2.1 Overview	2
1.2.2 Contributions	3
1.3 System Identification for Miniature Quadrotor Aircraft	3
1.3.1 Overview	3
1.3.2 Contributions	3
1.4 Using Accelerometers for Rotorcraft State Estimation	4
1.4.1 Overview	4
1.4.2 Contributions	4
2 UAV Path Planning for Ground Target Observability	7
2.1 Introduction	7
2.2 Occlusion Model	8
2.3 Parametric Path Planner	10

2.3.1	Orbital Paths	12
2.3.2	Canyon-Following Paths	12
2.4	Chain-based Path Planner	15
2.5	Conclusions and Future Work	19
3	Motion Capture Testbed for Indoor Aircraft Navigation and Control	23
3.1	Introduction	23
3.2	Quadrotor Aircraft	23
3.3	Motion Analysis System	25
3.4	Quadrotor Controller	29
3.5	Conclusion	31
4	System Identification for Quadrotor Aircraft	33
4.1	Introduction	33
4.2	System Architecture	33
4.3	Input Selection	34
4.4	Models for Commanded to Realized Angle	36
4.5	An Expanded Model for Position Hold	37
4.6	Simulation-based Gain Tuning	39
4.7	Conclusion	42
5	Using Accelerometers for Rotorcraft State Estimation	43
5.1	Introduction	43
5.2	Accelerometer Model	44
5.3	Estimator Design	50
5.3.1	Background	50

5.3.2	Naive Estimator Design	52
5.3.3	Estimator Incorporating Low-pass Filter Model	54
5.3.4	Hardware Demonstration of Attitude Estimators	56
5.4	Conclusions	58
6	Conclusions and Future Work	61
6.1	UAV Path Planning for Ground Target Observability	61
6.1.1	Overview and Contributions	61
6.1.2	Future Work	61
6.2	Motion Capture Testbed for Indoor Aircraft Navigation and Control	62
6.2.1	Overview and Contributions	62
6.2.2	Future Work	62
6.3	System Identification for Miniature Quadrotor Aircraft	63
6.3.1	Overview and Contributions	63
6.3.2	Future Work	63
6.4	Using Accelerometers for Rotorcraft State Estimation	63
6.4.1	Overview and Contributions	63
6.4.2	Future Work	64
	Bibliography	65
A	Flying Rotorcraft Using Cortex for State Estimation	67
A.1	Introduction	67
A.2	Architecture Overview	67
A.3	Pre-Flight Checklist	68
A.4	Configuring CortexQuadLink Control Software	74

A.5 Conclusions and Future Work	76
---	----

List of Figures

2.1	The function $P_d(\mathbf{x}_t, \mathbf{x}_s, m)$ with \mathbf{x}_t held constant at position $(0, 0)$ and \mathbf{x}_s varied in one meter increments.	9
2.2	Gaussian Mixture Model of the function shown in Figure 2.1.	11
2.3	Example of an orbital path with center selected using Equation (2.7)	12
2.4	Example of an orbital path with center selected using Equation (2.7)	13
2.5	Example canyon-following path shape. The parameters l , P_0 , and θ are modified to optimize target viewability.	13
2.6	Example of a canyon-following path fit to probability data.	14
2.7	Example of a canyon-following path fit to probability data.	15
2.8	Definition of θ_{max} , the maximum allowable turn angle to approximate a circle using a discrete chain.	17
2.9	Definition of terms for chain-based path planner.	19
2.10	Examples of paths generated by chain-based path planner	20
3.1	Hummingbird quadrotor from Ascending Technologies.	24
3.2	Custom board for wireless communication with Hummingbird quadrotor. . .	24
3.3	Top view of Motion Analysis camera configuration.	26
3.4	Motion Analysis camera configuration	26
3.5	Demonstration of the effect of measurement errors on angle accuracy. The larger the distance d between A and B, the lower the impact of the error e on angle estimate error.	27
3.6	Angle estimates computed by Cortex with the aircraft remaining stationary in the volume.	28

3.7	Hummingbird quadrotor with added communication board, masking to prevent unwanted reflection, and markers for Motion Analysis tracking.	28
3.8	Flowchart for experimental setup.	29
3.9	Block diagram of the controller used for the quadrotor aircraft. While not shown in the diagram, each block does have access to system states provided by Cortex.	30
3.10	Hummingbird flying over a model city to test path planners for ground-target observability	31
4.1	Flowchart describing system identification architecture. Position commands were fed into the existing position hold controller, which generated angle-hold commands. The angle-hold commands were recorded along with the angle outputs measured by Cortex.	34
4.2	Frequency content of swept square wave input used for system identification	35
4.3	Frequency content of sum of sines input used for system identification	36
4.4	Comparison of actual system behavior with predicted system behavior for pitch control on the Hummingbird aircraft. The label 'P1' corresponds to a the output generated by a single-pole transfer function model, while the 'arx111' label corresponds to an ARX model.	37
4.5	Quadrotor control model, including both angle and position dynamics	38
4.6	Free-body diagram of quadrotor aircraft	39
4.7	Simulated step response with original, untuned gains	40
4.8	Simulated step response with tuned gains	41
4.9	Predicted vs. actual step response of the rotorcraft using gains optimized in simulation	41
4.10	System output using actual gains for control and the small-angle approximation in the plant simulation. The system is unstable, but the output is bounded because the control inputs are saturated.	42
5.1	Simplified diagram of an accelerometer. Acceleration is measured by detecting deflections in the lever arm.	45
5.2	Simplified free-body diagram for a quadrotor helicopter	46

5.3	Accelerometer-based angle estimates on the Hummingbird quadrotor	47
5.4	A comparison of actual quadrotor pitch angles compared to accel-based estimates and a low-pass filtered version of the actual angles. This figure demonstrates that a low-pass filter model accurately describes accel-based attitude estimates on rotorcraft.	49
5.5	Actual pitch angle vs. estimated pitch angle using accelerometers for state correction directly.	53
5.6	Angle outputs from the Hummingbird autopilot compared to actual angles obtained using Cortex.	54
5.7	Actual pitch angle vs. estimated pitch angle using LPF accelerometer model	56
5.8	A comparison of estimation methods on a flying rotorcraft. The direct accelerometer method results in poor performance due to the neglected dynamics of the rotorcraft. The estimation method that accounts for aircraft dynamics produces results that are consistent with truth data.	57
5.9	Angle estimation errors using accelerometers directly compared to errors resulting when considering the effect of aerodynamic drag	58
A.1	Flowchart for experimental setup.	69
A.2	Dialog for setting the minimum number of lines (pixels) required for the Motion Analysis system to detect and track a marker. For tracking the Hummingbird and Pelican props, this should be set to two.	71
A.3	Dialog for enabling the SDK for broadcasting states computed by Cortex over the TCP/IP connection.	71
A.4	Main Cortex display window when a prop is being tracked. Lines should be drawn between the markers in the prop as shown in the area highlighted in red.	72
A.5	Output of CortexQuadLink software when it has successfully connected to Cortex and is running control loops. If the XBee modems are properly connected and the aircraft is turned on, then the aircraft should be receiving control commands.	73
A.6	Main Cortex display window when a prop is being tracked. Lines should be drawn between the markers in the prop as shown in the area highlighted in red.	74

Chapter 1

Introduction

This thesis describes work in a variety of topics related to aerial robotics, including system identification, state estimation, control, and path planning. Descriptions of each topic, along with a brief statement of contributions, is given below.

1.1 UAV Path Planning for Ground Target Observability

1.1.1 Overview

If an unmanned aerial vehicle (UAV) is to be used for ground target surveillance, the aircraft should be controlled along a path that optimizes the ability of the aircraft to track the ground target. While a variety of path planners have been developed to facilitate autonomous ground target tracking from the air (See [1], [2], [3], [4], [5]), very little research considers the effect of occlusions. It may be justifiable to ignore the effect of occlusions in some circumstances, but in cluttered environments, occlusions could significantly effect the capability of the aircraft to view the ground target.

In Chapter 2, a model is developed to describe the affect of occlusions on ground target viewability. Two different path planners are then developed that take advantage of information captured by the occlusion model.

1.1.2 Contributions

In existing literature that describes planning UAV paths for ground target tracking, occlusions are generally ignored. In one notable exception [6], Kim restricts UAV paths to orbits with the aircraft's minimum turn radius, and attempts to optimize the center of the orbit so that a ground target is visible as often as possible. While this method is effective, it is restrictive in that it only supports orbital paths.

The path planners described in Chapter 2 are capable of generating a much richer set of paths than existing path planners. In addition, the occlusion model developed in Chapter 2 provides a concise, well-structured method for describing occlusions in urban environments.

1.2 Motion Capture Testbed for Indoor Aircraft Navigation and Control

1.2.1 Overview

Chapter 3 describes the use of a 3D camera positioning system from Motion Analysis for real-time state estimation and control of a quadrotor helicopter. In the system, a 3D camera system tracks the location and attitude of the aircraft. This information is then transmitted to a second computer that logs the data, runs control loops, computes paths, etc.

There are a variety of motivations for building this type of system. First, many experiments (particularly those involving state estimation and/or mapping) require ground truth to evaluate performance. For example, to evaluate the performance of an algorithm designed to estimate pitch and roll angles on a rotorcraft, the actual pitch and roll angles must be known. The camera positioning system provides this information.

A second motivation for building the system is for control purposes. If you want a robotic platform to follow a specific path, or maintain a particular attitude, the controller must know the current state of the system. GPS is often used in conjunction with other systems to provide system states for control, but FAA restrictions make it difficult to fly autonomous aircraft outdoors. There are no such restrictions for indoor flight, but GPS unfortunately does not work indoors, and even if it did, it is not accurate enough for use in confined spaces. A 3D camera positioning system provides an ideal alternative.

A third motivation is that truth data can be used to characterize physical systems in a way that simplifies control design. In Chapter 4, system identification techniques are used in conjunction with data collected by the motion capture system to develop controllers for quadrotor aircraft.

While there are other possible uses of the system, Chapter 3 focuses exclusively on the architecture that was designed to allow a quadrotor to fly using information from the camera system.

1.2.2 Contributions

The architecture described in Chapter 3 can be used to run flight tests for a variety of experiments. To-date, it has been used to test the path planner developed in Chapter 2, to test cooperative ground-target tracking using a UAV and a UGV, to develop a system model to describe the Hummingbird and Pelican quadrotors (Chapter 4), and to perform a variety of other flight experiments. In the future, it will serve as a starting-point for experiments in indoor navigation, landing-site identification, and cooperative multi-agent state estimation, among other things.

1.3 System Identification for Miniature Quadrotor Aircraft

1.3.1 Overview

In the MAGICC Lab, experiments are often performed related to state estimation, control, and path planning of unmanned rotorcraft. To facilitate these experiments, it is useful to have an accurate model of the flight performance of the aircraft used in experiments. In Chapter 4, system identification techniques are used to characterize the flight behavior of the Hummingbird and Pelican quadrotors from Ascending Technologies. Specifically, system identification techniques are used to develop transfer functions relating commanded to realized angles on both aircraft.

System identification is a well-developed field, and Chapter 4 is not intended to present new research. Rather, Chapter 4 documents the processes used to characterize aircraft flight performance using the testbed described in Chapter 3.

1.3.2 Contributions

While Chapter 4 does not present new research, it provides useful models to describe the behavior of rotorcraft commonly used in the MAGICC Lab. It also documents the procedure used to obtain the models so that other physical systems can be characterized in the future.

1.4 Using Accelerometers for Rotorcraft State Estimation

1.4.1 Overview

Accelerometers are often used in conjunction with rate gyros for attitude estimation on moving platforms. The fundamental idea is that accelerometers can indirectly measure the gravity vector and determine pitch and roll angles. Rate gyros are used to complement the accelerometer measurements and prevent non-gravity accelerations from distorting the angle estimates.

This simple estimation method runs into problems when external non-gravity accelerations are significant enough that the gravity vector cannot be measured accurately. One such case is on flying rotorcraft, where any short-term change in pitch and roll angles produces lateral acceleration that effectively “cancels out” the gravity measurement on the x and y accelerometer axes, making direct measurement of pitch and roll angles impossible. Nevertheless, many quadrotor-based angle estimators use accelerometers directly without obvious problems (see [7], [8], [9], [10], [11], and [12], for example).

In [13], Martin, et al. describes the flight dynamics of a quadrotor aircraft and how those dynamics affect accelerometer measurements. A model is developed to explain why estimators that ignore flight dynamics tend to work anyway, but the specific errors produced by such estimators are not quantified. In Chapter 5, the nature of the errors are quantified, and an estimator is developed and tested that produces much better accuracy by accounting for rotorcraft flight dynamics.

1.4.2 Contributions

While attitude estimation is generally a well-understood topic, there are some counter-intuitive results associated with the use of accelerometers on rotorcraft. On quadrotor aircraft in particular, the effect of the aircraft dynamics on accelerometer measurements is often ignored, causing significant attitude estimation errors. These errors often go unnoticed when angle estimates are not compared with actual ground truth.

In Chapter 5, the effect of rotorcraft dynamics on accelerometer measurements is described in detail. An estimator that ignores rotorcraft dynamics is implemented and

tested both in simulation and on hardware to demonstrate the types of errors that can be expected if accelerometers are not used correctly. A simple model is developed to describe the expected accelerometer output, and a new estimator is presented that uses the model to improve attitude estimates. In simulation and in hardware, the new estimator is shown to produce results with much higher accuracy than an estimator that ignores rotorcraft flight dynamics.

The problems described in Chapter 5 are also described by Martin, et al. in [13], but the paper focuses mainly on the derivation of rotorcraft flight dynamics, and does not provide quantifiable data to highlight the significance of the estimation errors that can be expected from the misuse of accelerometers on rotorcraft. Chapter 5 provides details about the nature of possible attitude estimation errors, along with an intuitive model to explain accelerometer behavior, and a new estimator with flight-test results compared with ground truth.

Chapter 2

UAV Path Planning for Ground Target Observability

2.1 Introduction

Unmanned aerial vehicles (UAVs) equipped with vision sensors have proved to be useful information gathering tools; as a result, many path planners designed specifically with ground target tracking in mind have been developed (See [1], [2], [3], [4], [5]). But even with all the recent attention on UAV-based ground target tracking, few consider the effect of occlusions. Ignoring occlusions may be justifiable in many situations, but in dense urban environments, occlusions can be particularly problematic. In Section 2.2, a model is described that efficiently captures the essence of the urban environment, taking into account the effect of occlusions on the UAV's ability to detect the target, and the model is demonstrated in a simulated city.

As with many path-planning problems, one of the largest challenges inherent in UAV path planning is the size of the configuration space. The number of possible paths is so large that, even with modern computers, we could not hope to search the entire space to find the best path in a reasonable amount of time. For non-holonomic vehicles like a UAV, the problem is made even more interesting because motion constraints (such as minimum turn radius) must not be violated. The result is that, even if we can easily identify a set of positions where we would like the UAV to be, planning the best way to visit them, and in what order, is a non-trivial problem.

One obvious simplification is to restrict the configuration space to a small subset of flyable paths. For example, in [6], Kim restricts UAV paths to orbits with the aircraft's minimum turn radius, and attempts to optimize the center of the orbit so that a ground target is visible as often as possible. Kim discretizes each orbit into a set of positions and tests viewability of the target at each position; the orbit center is adjusted using a simplified

hill-climbing algorithm. While Kim’s algorithm is reasonably fast, it is restrictive in that it can only plan orbits. In Section 2.3, a method is presented for planning a larger set of flyable paths while still maintaining low computational overhead. In this method, basic path shapes are represented using continuous, parametric functions, and performance is optimized by modifying a small set of parameters. A fitness function tests the path shape against the occlusion model developed in Section 2.2, so that parameters can be chosen to maximize target viewability.

The parametric planner described in Section 2.3 is most effective when the UAV position does not need to be coordinated in time. However, in some cases it might be important to know where the UAV will be at specific times in the future. For example, when multiple UAVs are used to cooperatively track a ground target, individual UAV paths should be complementary: when one UAV cannot see the target, the other should be positioned to compensate. It would be difficult to plan cooperative paths without knowledge of each UAV’s expected future position.

Since the parametric planner described in this chapter does not account for the current UAV position, time considerations cannot be handled elegantly. To solve this problem, Section 2.4 introduces a planner that models the UAV path using a series of connected waypoints. The waypoints serve as links in a simulated chain. The chain is placed inside a force-field that is generated using the gradient of the occlusion model, so that each link in the chain tends to move toward good regions for detecting the ground target. Straightening forces are applied to prevent UAV flight constraints from being violated. Since the path is represented using waypoints that are a fixed distance apart, it is easy to determine roughly where the UAV will be at any given time.

2.2 Occlusion Model

In this chapter, it is assumed that the UAV uses vision as its primary sensor. It is also assumed that the target identification problem has been solved, and that the sensor’s ability to detect the target is modeled by the function $P_d(\mathbf{x}_t, \mathbf{x}_s)$, where \mathbf{x}_t is the target location, and \mathbf{x}_s is the sensor location; here it is assumed that the sensor is gimballed and pointing toward the target. Note that $P_d(\mathbf{x}_t, \mathbf{x}_s)$ is not a probability distribution. Rather,

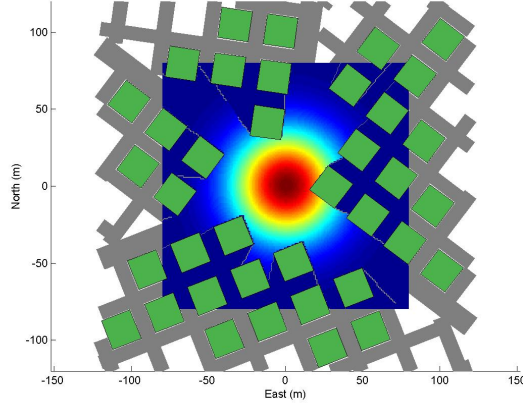


Figure 2.1: The function $P_d(\mathbf{x}_t, \mathbf{x}_s, m)$ with \mathbf{x}_t held constant at position $(0, 0)$ and \mathbf{x}_s varied in one meter increments.

it describes the probability that the sensor will detect the target given the positions of the target and the sensor. Depending on the sensor being used, $P_d(\mathbf{x}_t, \mathbf{x}_s)$ could conceivably be close to one everywhere that the line of sight is not occluded.

While a number of sensor models could be used, it is assumed without loss of generality that $P_d(\mathbf{x}_t, \mathbf{x}_s)$ resembles a Gaussian function of the distance between the target and the sensor:

$$P_d(\mathbf{x}_t, \mathbf{x}_s) \triangleq \eta \exp \left(-\frac{1}{2} (\mathbf{x}_s - \mathbf{x}_t)^\top \Sigma^{-1} (\mathbf{x}_s - \mathbf{x}_t) \right), \quad (2.1)$$

where Σ shapes the curve in the same fashion that the covariance shapes a Gaussian probability distribution, and η is a scale factor. In effect, η describes the probability that the sensor will detect the target when the distance between the sensor and the target is small.

The effects of occlusions also need to be considered, since the sensor and target are assumed to be in an urban environment. Let $P_d(\mathbf{x}_t, \mathbf{x}_s, m)$ be the probability that the sensor will detect the target conditioned on the target location, the sensor location, and a known city map, m . Then

$$P_d(\mathbf{x}_t, \mathbf{x}_s, m) = \begin{cases} P_d(\mathbf{x}_t, \mathbf{x}_s) & \text{if LOS is not occluded,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

Figure 2.1 shows an example probability function in an urban environment (all figures in this chapter are best viewed in color). In this example, the target position, \mathbf{x}_t , is held constant at $(0,0)$, while the sensor position, \mathbf{x}_s , is varied in one meter increments on both axes. As shown, when a building occludes the line of sight, the probability of detection is zero. Occlusions are detected by testing each line of sight vector for intersections with each building polygon in the map.

In practice, it can be useful to represent the detection probabilities using continuous functions rather than a large set of discretized data points. To this end, a mixture of Gaussians is fit to the discrete probability data. Assume that the target position remains constant. Then $P_d(\mathbf{x}_t, \mathbf{x}_s, m)$ can be approximated as

$$\hat{P}_d(\mathbf{x}_t, \mathbf{x}_s, m) = \sum_{i=1}^n \alpha_i \exp \left(-\frac{1}{2} (\mathbf{x}_s - \mu_i)^\top \Sigma_i^{-1} (\mathbf{x}_s - \mu_i) \right), \quad (2.3)$$

where n is the number of Gaussians used to approximate $P_d(\cdot)$. The parameters α_i , μ_i , and Σ_i^{-1} describe the weight, mean, and covariance matrix of each Gaussian used in the fit, and are unique to the target location (i.e. a different target location will yield a different mixture of Gaussians to describe the probability). The fit is performed using expectation maximization. Figure 2.2 shows a Gaussian mixture (GM) fit to the probability data shown in Figure 2.1. While the GM fit is not perfect, it captures the essence of the environment where occluded regions have close-to-zero detection probabilities, and high-probability areas are accurately represented.

2.3 Parametric Path Planner

With a model to describe good positions of the UAV in terms of ground target tracking, paths can be planned that optimize target viewability. Let $X_a(\gamma, \Theta)$, $\gamma \in [0, T]$ be a parameterization of the UAV's path with respect to parameter γ . The entire UAV path will be traced as γ ranges from 0 to T , but γ does not necessarily represent the UAV's position in time. The vector Θ contains all the parameters that affect the shape, orientation, and position of the path. Define the path fitness, $F(\Theta)$, as the average detection probability over

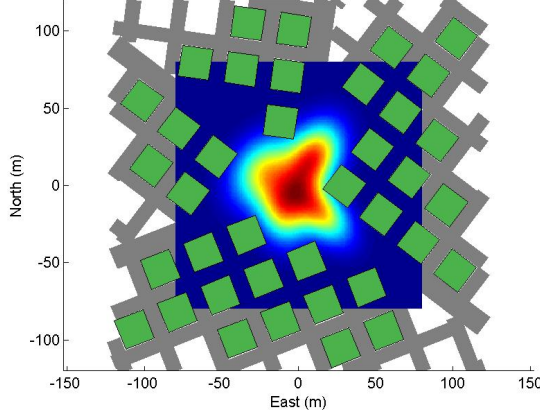


Figure 2.2: Gaussian Mixture Model of the function shown in Figure 2.1.

the path length $L(\Theta)$, or

$$F(\Theta) = \frac{1}{L(\Theta)} \int_0^T P_d(\mathbf{x}_t, X_a(\gamma, \Theta), m) d\gamma. \quad (2.4)$$

If the sensor model function, $P_d(\mathbf{x}_t, \mathbf{x}_s, m)$, is modeled using a Gaussian mixture as demonstrated in Section 2.2, then Equation (2.3) can be substituted into Equation (2.4) to obtain

$$F(\Theta) = \frac{1}{L(\Theta)} \int_0^T \sum_{i=1}^n \alpha_i \exp \left(-\frac{1}{2} (X_a(\gamma, \Theta) - \mu_i)^\top \Sigma_i^{-1} (X_a(\gamma, \Theta) - \mu_i) \right) d\gamma. \quad (2.5)$$

Equation (2.5) can be used to describe the fitness of any path that can be parameterized using a single continuous function. The parameters that yield the optimal path are given by

$$\Theta^* \triangleq \arg \max_{\Theta} \{F(\Theta)\}. \quad (2.6)$$

Since the fitness equation is continuous, an efficient gradient ascent method can be used to arrive at a local maximum. If Θ^- be the most recent estimate of the parameter vector, then the next estimate, Θ^+ , is given by

$$\Theta^+ = \Theta^- + \kappa \frac{\partial F(\Theta)}{\partial \Theta}, \quad (2.7)$$

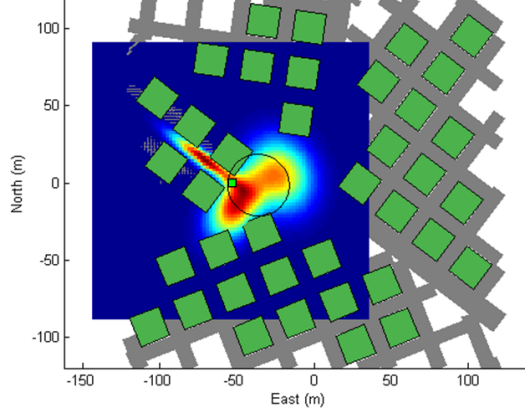


Figure 2.3: Example of an orbital path with center selected using Equation (2.7)

where κ is a small, positive constant. If constraints on Θ are needed to keep the parameterized path flyable, then other methods may be used.

2.3.1 Orbital Paths

With a framework to measure the fitness of a path, specific path shapes can be investigated. A simple parameterized path is an orbit of fixed radius, where only the orbit center is optimized. The parameterization is given by

$$X_a(\gamma, \Theta) = P_0 + \begin{pmatrix} r_{min} \cos(\gamma) \\ r_{min} \sin(\gamma) \end{pmatrix}, \quad \gamma \in [0, T], \quad T = 2\pi, \quad (2.8)$$

where r_{min} is the minimum turn radius of the aircraft, and $\Theta = P_0$.

Figures 2.3 and 2.4 provide examples of orbital paths computed using Equation (2.7). The green square indicates the position of the target, while the gradient surrounding the target represents the ability of the UAV to detect the target at various positions. The computed orbit is represented using a black line placed on the city map.

2.3.2 Canyon-Following Paths

In many cases, simple circular orbits may be less than optimal. Consider, for example, the case where the target is traveling down a narrow urban canyon. If the buildings on either

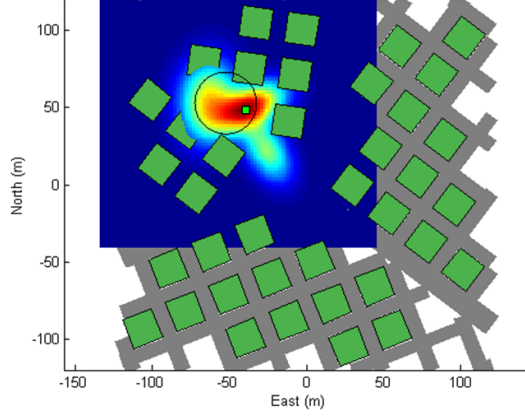


Figure 2.4: Example of an orbital path with center selected using Equation (2.7)

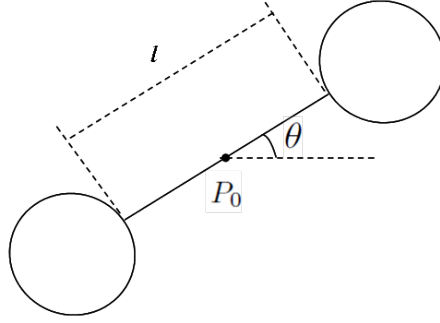


Figure 2.5: Example canyon-following path shape. The parameters l , P_0 , and θ are modified to optimize target visibility.

side of the canyon are sufficiently high compared to the UAV's altitude, then the target is only viewable inside a narrow region running along the urban canyon. In this situation, it might make more sense to plan a path that follows the canyon. Accordingly, the second parameterized path is a straight line with circular orbits at either end of the line. Figure 2.5 shows an example of the canyon-following path shape. The point P_0 is the center of the line segment connecting the two circles, θ is the rotation of the path about the z-axis, and l is the length of the line segment connecting the two circles. All three parameters are adjusted to optimize the sensor's ability to detect the target. Letting $\Theta = (P_0, \theta, l)^T$, the parameterized

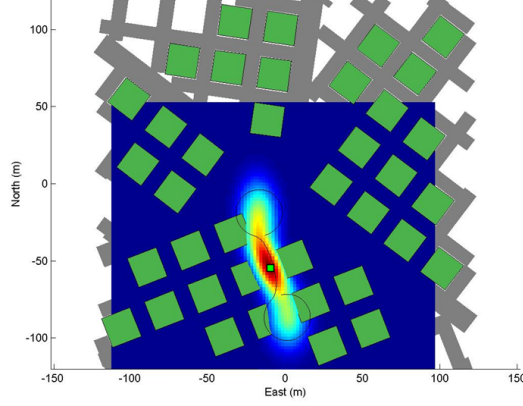


Figure 2.6: Example of a canyon-following path fit to probability data.

path is given by

$$X_a(\gamma, \Theta) = P_0 + R(\theta) \begin{pmatrix} (1 - 2L_{2\pi}(\gamma))(r \cos(\gamma) - \frac{l}{2}) \\ (1 - 2L_{2\pi}(\gamma))r \sin(\gamma) \end{pmatrix}, \quad (2.9)$$

where $\gamma \in [0, T]$, $T = 4\pi$, r is the radius of the two circular orbits, $L_a(\gamma)$ is the logistic function given by

$$L_a(\gamma) = \frac{1}{1 + e^{-k(\gamma-a)}},$$

and $R(\theta)$ is the rotation matrix

$$R(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$

Equation (2.9) is simply the combination of two circle parameterizations, with logistic functions used to specify at which time each individual circle is traced. A relatively straight line is traced between the first circle and the second because the logistic functions do not perfectly model the step function. Figures 2.6 and 2.7 provide examples of canyon-following paths computed using Equation (2.7).

Using Matlab on a 2.4 Ghz Pentium 4 with 1 GB of RAM, both the canyon-following and orbital path planners consistently generate optimized paths in well under one second.

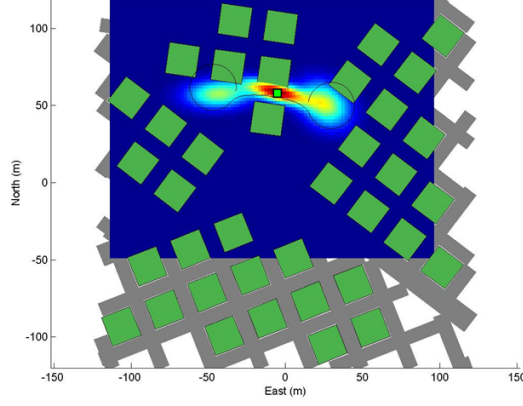


Figure 2.7: Example of a canyon-following path fit to probability data.

Since paths can be generated quickly, it is possible to plan both types of paths, and then select the one with the highest fitness value.

2.4 Chain-based Path Planner

To generate more detailed, informative UAV paths, a chain is placed in a simulated force-field, where angles between links are constrained to enforce the minimum turn radius of the UAV. The chain model is specifically designed so that good paths can be discovered quickly, and existing paths can be modified in real-time as the target moves.

Similar to [14], the chain is modeled as a collection of unit-mass points constrained to the 2-D plane. Letting $\mathbf{z}_i = (x_i, y_i)$ be the position of the i th element in the chain, then an N -link chain is represented by

$$\mathbf{c} = [z_1 \ z_2 \ \dots \ z_N]^T.$$

The magnitude of the force applied to link i is proportional to $(1 - \hat{P}_d(\mathbf{x}_t, \mathbf{z}_i, m))$, while the direction of the force is given by the gradient of $\hat{P}_d(\mathbf{x}_t, \mathbf{z}_i, m)$. Let \mathbf{g}_i be the unit vector pointing in the same direction as the gradient of $\hat{P}_d(\mathbf{x}_t, \mathbf{z}_i, m)$, or

$$\mathbf{g}_i = \frac{\partial \hat{P}_d(\mathbf{x}_t, \mathbf{z}_i, m)}{\partial z_i} / \left\| \frac{\partial \hat{P}_d(\mathbf{x}_t, \mathbf{z}_i, m)}{\partial z_i} \right\|. \quad (2.10)$$

The unconstrained dynamics for link i are then

$$\ddot{\mathbf{z}}_i = \gamma_1(1 - \hat{P}_d(\mathbf{x}_t, \mathbf{z}_i))\mathbf{g}_i, \quad (2.11)$$

where γ_1 is a positive constant. The unconstrained dynamics of the entire chain are therefore given by

$$\begin{aligned} \ddot{\mathbf{c}} &= \gamma_1 \begin{bmatrix} (1 - \hat{P}_d(\mathbf{x}_t, \mathbf{z}_1))\mathbf{g}_1 \\ (1 - \hat{P}_d(\mathbf{x}_t, \mathbf{z}_2))\mathbf{g}_2 \\ \vdots \\ (1 - \hat{P}_d(\mathbf{x}_t, \mathbf{z}_N))\mathbf{g}_N \end{bmatrix} \\ &= \gamma_1 \mathbf{u}. \end{aligned} \quad (2.12)$$

Link motion is also constrained to keep the distance between adjacent links constant. Let L be the desired distance between each adjacent chain link, and define the constraint vector $\phi(\mathbf{c})$ as

$$\phi(\mathbf{c}) \triangleq \begin{bmatrix} \|\mathbf{z}_2 - \mathbf{z}_1\|^2 - L^2 \\ \|\mathbf{z}_3 - \mathbf{z}_2\|^2 - L^2 \\ \vdots \\ \|\mathbf{z}_N - \mathbf{z}_{N-1}\|^2 - L^2 \end{bmatrix}. \quad (2.13)$$

As long as each element of $\phi(\mathbf{c})$ is zero, the distances between each link in the chain will be correct. To push each element of $\phi(\mathbf{c})$ toward zero, a restoring term is added to Equation (2.12) to get

$$\ddot{\mathbf{c}} = \gamma_1 \mathbf{u} - \gamma_2 \frac{\partial \phi}{\partial \mathbf{c}}, \quad (2.14)$$

where γ_2 is a positive constant. Large values of γ_2 force adjacent links to remain a distance L apart. On the other hand, as γ_2 gets large, the differential equation stiffens, requiring more computational resources to simulate the model accurately. A tradeoff must be made between maintaining exact distances, and modeling the chain movement in a computationally efficient manner.

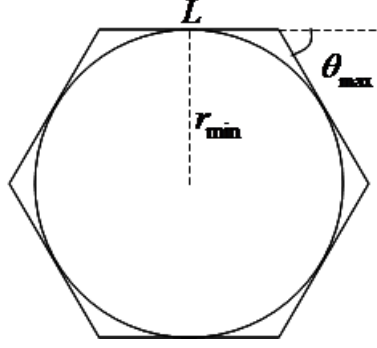


Figure 2.8: Definition of θ_{max} , the maximum allowable turn angle to approximate a circle using a discrete chain.

To ensure that the minimum turn radius of the UAV is not violated, straightening forces are applied to each link in the chain. Let r_{min} be the minimum turn radius of the UAV and let θ_{max} be the maximum allowable angle formed by the vectors between three adjacent links (See Figure 2.8). Then the minimum number of links required to complete a full circle is given by

$$n = \frac{2\pi}{\theta_{max}}. \quad (2.15)$$

As shown in Figure 2.8, the length of the approximate circular path is at least as long as a circle with radius r_{min} , or

$$nL \geq 2\pi r_{min}. \quad (2.16)$$

Combining Equations (2.15) and (2.16), we get

$$\theta_{max} \leq \frac{L}{r_{min}}. \quad (2.17)$$

To guarantee that turn radius constraints aren't violated, Equation (2.17) is treated as a strict equality.

Ideally, the straightening force applied to enforce turn radius constraints would only be applied when the maximum angle constraint is violated, and then only enough to correct the violation. As shown in Figure 2.9, the straightening force applied to link i is designed to

ensure that $|\theta_i| < \theta_{max}$, where θ_i is the angle between \mathbf{v}_i^1 and \mathbf{v}_i^2 , defined as

$$\begin{aligned}\mathbf{v}_i^1 &= (\mathbf{z}_i - \mathbf{z}_{i-1}) / \|\mathbf{z}_i - \mathbf{z}_{i-1}\|, \\ \mathbf{v}_i^2 &= (\mathbf{z}_{i-1} - \mathbf{z}_{i-2}) / \|\mathbf{z}_{i-1} - \mathbf{z}_{i-2}\|.\end{aligned}\tag{2.18}$$

It follows that θ_i is given by

$$\theta_i = \arccos(\mathbf{v}_i^1 \cdot \mathbf{v}_i^2).\tag{2.19}$$

The straightening force for link i is given by

$$\mathbf{f}_i = \frac{\lambda_i}{1 + \exp(k(\theta_{max} - \theta_i))} (\mathbf{v}_i^1)^\perp,\tag{2.20}$$

where k is a positive constant that defines how closely the logistic function approximates a step function, λ_i is the upper limit of the straightening force for link i , and where

$$(\mathbf{v}_i^1)^\perp = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{v}_i^1.\tag{2.21}$$

For each link, λ_i must be large enough to at least match the sum of all possible forces on all subsequent links in the chain; otherwise, there may be cases where the straightening force might not be high enough to prevent minimum turn radius constraints from being violated. If there are N links in the chain, and $i = 1$ corresponds to the first link, then define λ_i as

$$\lambda_i = \gamma_1(N + 1 - i).\tag{2.22}$$

Recall that γ_1 is the largest possible magnitude of the unconstrained force applied by the force field to each link.

Let \mathbf{f}_s be the vector of all straightening forces applied to the chain, defined as

$$\mathbf{f}_s \triangleq \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_N \end{bmatrix}.\tag{2.23}$$

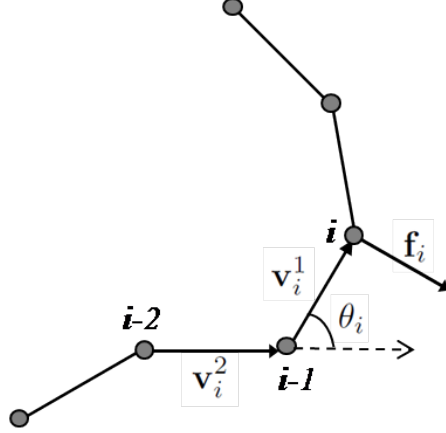


Figure 2.9: Definition of terms for chain-based path planner.

Then the chain dynamics with straightening forces are given by

$$\ddot{\mathbf{c}} = \gamma_1 \mathbf{u} - \gamma_2 \frac{\partial \phi}{\partial \mathbf{c}} \phi + \mathbf{f}_s - \gamma_3 \dot{\mathbf{c}}, \quad (2.24)$$

where the last term is a damping term, and where γ_3 is a positive constant.

When using the chain to plan paths in real-time, the first two links serve as waypoints for the UAV to follow and are not allowed to move. When the UAV nears the end of the line segment formed by the first two links, the first link is removed from the chain and added to the end, and the links that then comprise the beginning of the chain are fixed as new waypoints. So, while the UAV always has two unchanging waypoints to follow, the remainder of the chain continuously adapts to changing target conditions. Figure 2.10 shows a number of paths generated by the chain-based planner. Each path is represented by a black line over the city map, while the UAV is represented by a small white triangle at the beginning of the path.

2.5 Conclusions and Future Work

In this chapter, a continuous-time model was presented for describing the effect of occlusions on a vision-based sensor in urban terrain. Using the occlusion model, two path

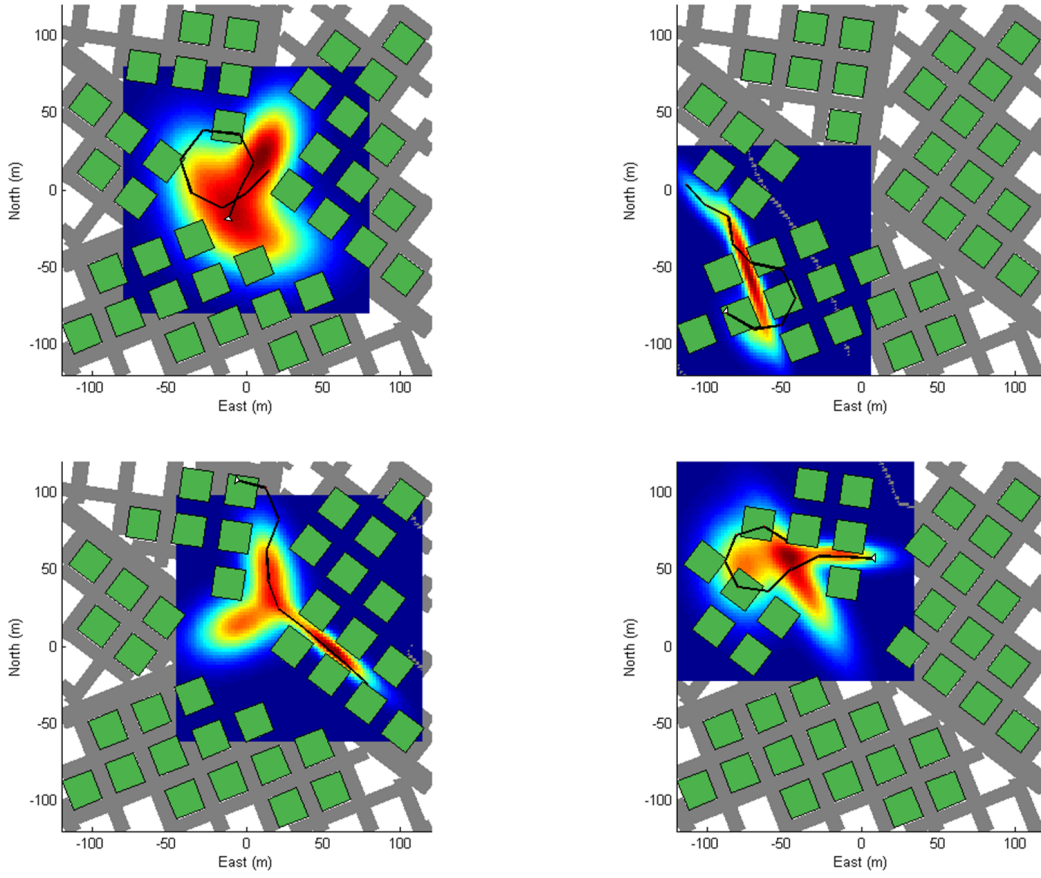


Figure 2.10: Examples of paths generated by chain-based path planner

planners were described that use the model to plan UAV paths that maximize the ability of the sensor to view a ground target, a “parametric” planner, and a “chain-based” planner.

The parametric path planner can optimize any path shape that can be represented using a single, continuous parametric function. We showed two such paths: an orbit planner, and a canyon-following planner. In both cases, the parametric planner consistently computes optimized paths in well under one second.

The chain-based planner is advantageous because it runs quickly, paths can be modified continuously as the ground target moves, and there are fewer restrictions on the shape of the generated path. In the future, it may be desirable to use multiple air vehicles to cooperatively track a ground target. In this case, the chain-based planner could be useful because it encodes the expected UAV position with respect to time, allowing each UAV to

determine when it expects its line of sight to be occluded. By weighting the forces on each chain link properly, multiple UAVs can “hand-off” target tracking responsibility so that at least one sensor is positioned to view the ground target at all times.

Chapter 3

Motion Capture Testbed for Indoor Aircraft Navigation and Control

3.1 Introduction

This chapter describes the use of a 3D camera positioning system from Motion Analysis for real-time state estimation and control of a quadrotor helicopter. In the system, a 3D camera system tracks the location and attitude of the aircraft. This information is then transmitted to a second computer that logs the data, runs control loops, computes paths, etc. While there are a variety of possible uses of the system, this chapter focuses exclusively on the architecture that was designed to allow a quadrotor to fly using information from the camera system. In Chapter 4, the system is used to collect data for system identification. In Chapter 5, the positioning system is used to develop more accurate attitude estimation methods for flying rotorcraft.

3.2 Quadrotor Aircraft

The Hummingbird quadrotor from Ascending Technologies (see Figure 3.1) was used for this thesis. Quadrotors are ideal test platforms because they are mechanically simple and therefore robust to crashes, relatively inexpensive, and easy to repair. Traditional helicopters, on the other hand, have complex mechanical linkages that are expensive, easily damaged, and difficult to repair. Larger, more dangerous propellers are also required on a helicopter to obtain the same amount of thrust as a quadrotor.

The Hummingbird quadrotor comes equipped with an autopilot that can be configured to run either rate-hold or angle-hold control loops at 1000 Hz. Custom motor controllers by Ascending Technologies allow high-bandwidth control of the aircraft, making the Hummingbird one of the most stable quadrotors available. The onboard autopilot supports



Figure 3.1: Hummingbird quadrotor from Ascending Technologies.



Figure 3.2: Custom board for wireless communication with Hummingbird quadrotor.

manual flight using an RC transmitter, but any combination of channels (thrust, pitch, roll, etc.) can be controlled via a TTL UART. This capability makes the Hummingbird ideal for controller development, since one channel can be placed under computer control while all others can be left under human control. The TTL UART onboard the autopilot can also stream telemetry information.

For communication, a custom board was designed to connect XBee radios from Digi directly to the Hummingbird autopilot (see Figure 3.2). On the ground, a matching XBee radio was connected to a desktop computer that ran control algorithms.

3.3 Motion Analysis System

As mentioned previously, a camera-based 3D positioning system from Motion Analysis was used as a replacement for GPS. The particular setup for this experiment used 8 cameras situated around the edge of a 9 meter by 5 meter room as shown in Figures 3.3 and 3.4. In the setup shown in the aforementioned figures, the usable volume (i.e., the volume in which two or more cameras can always see an object) is $4 \times 3 \times 1.5$ meters.

Each camera in the positioning system uses an array of near-infrared LEDs to illuminate the room. Synchronized images from all cameras are returned to a central computer. If at least two cameras detect the same reflective object, the 3D position of the object is automatically computed by Cortex, which is the positioning software provided with the cameras. To test the accuracy of the positioning system, a reflective marker was attached to a pair of calipers with measurement accuracy to within one thousandth of an inch. The calipers were then adjusted inside the volume so that the exact deviation of the marker could be measured and compared to the estimates made by Cortex. Relative position estimate errors were zero-mean with a standard deviation of 0.55 mm. Note that the calipers were limited to measuring distances of 150 mm, so larger errors may possibly manifest themselves across the entire volume.

Quadrotor state estimation is simplified by Cortex. A set of reflective points can be attached to a rigid body in a known configuration, called a prop, and Cortex will automatically try to fit the prop to image data received from the cameras. If the prop is found, then the 3D position and orientation of the object is computed.

Prop states (position, orientation) can then be streamed over a TCP/IP connection at up to 200 Hz. Figure 3.7 shows the prop used to define the Hummingbird quadrotor. The reflective markers on top are made out of foam and retro-reflective tape. The accuracy of the angle estimates returned by Cortex depend on the spacing between markers in the prop: the farther apart the markers, the better the angle estimates. Motion Analysis does

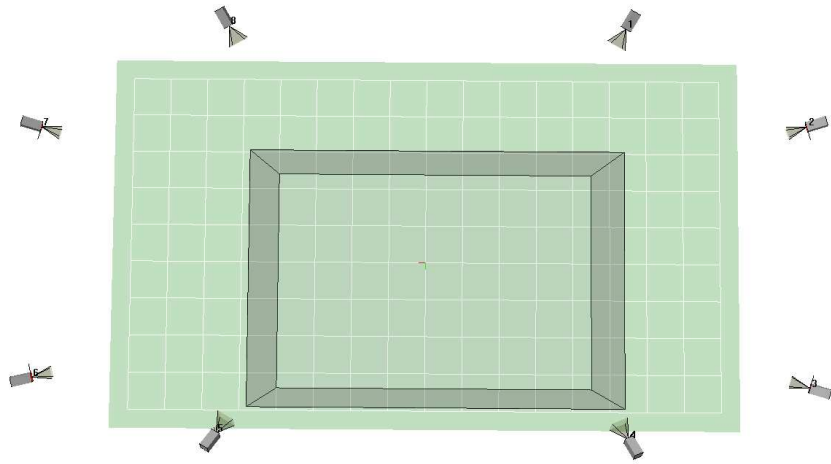


Figure 3.3: Top view of Motion Analysis camera configuration.

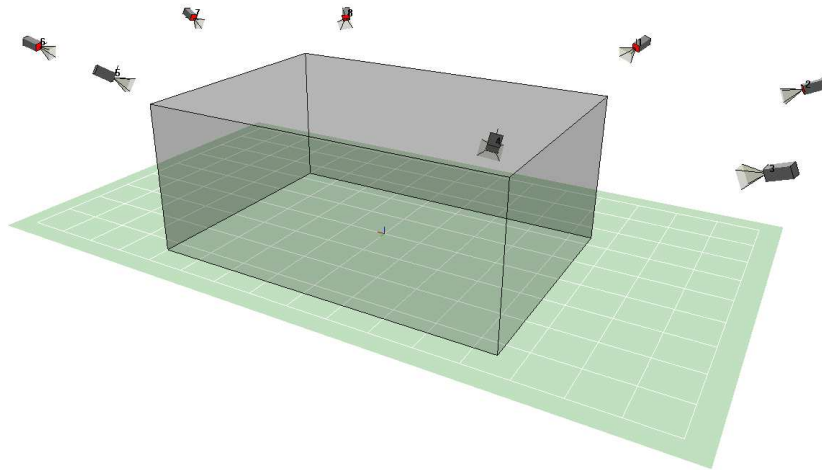


Figure 3.4: Motion Analysis camera configuration

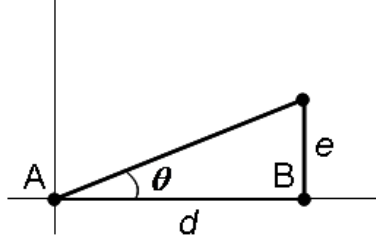


Figure 3.5: Demonstration of the effect of measurement errors on angle accuracy. The larger the distance d between A and B, the lower the impact of the error e on angle estimate error.

not provide details about the algorithm used to compute angle estimates, but the effect of marker spacing on angle accuracy can be shown intuitively by considering the measured angle between two points as shown in Figure 3.5.

Points A and B lie a distance d apart along the x-axis. The line between A and B forms an angle of $\theta = 0$ degrees with the horizontal axis. Assume as shown that the position of point B is measured with additional error e along the y-axis as shown. The angle θ is then measured as

$$\theta = \arctan(e/d). \quad (3.1)$$

Clearly, as d increases, angular error decreases. The full 3D estimator is more complicated than this example shows, but the noise in the angle estimates can be easily measured. The Hummingbird was placed inside the volume and the pitch angle was measured with Cortex while the aircraft remained stationary. The resulting angle estimates are shown in Figure 3.6. As shown, the estimate noise is well within one degree of the actual angle. More spacing between markers would result in less angle estimate error.

While no tests were performed to determine the delay in the positioning system state outputs, Motion Analysis claims that the delays are well under 10 milliseconds. There were no problems with excessively long delays while running experiments with the positioning system.

With reliable state estimates from Cortex, path planners running in Matlab, and a means for communicating with the aircraft, a “glue” application was needed to tie everything together. A central application was written to

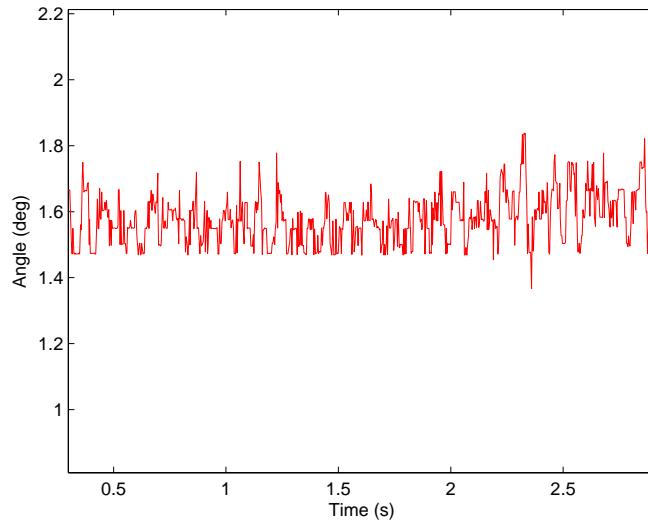


Figure 3.6: Angle estimates computed by Cortex with the aircraft remaining stationary in the volume.

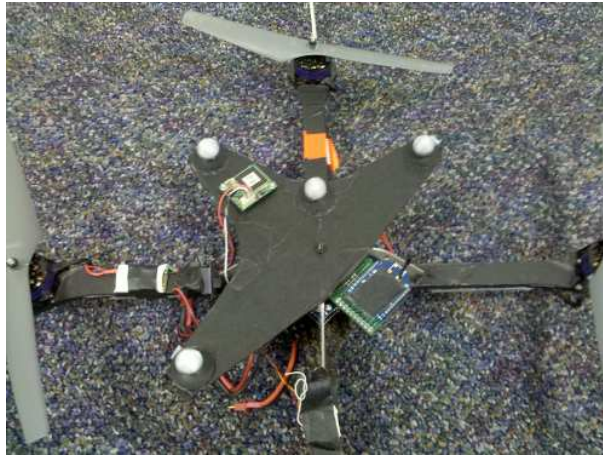


Figure 3.7: Hummingbird quadrotor with added communication board, masking to prevent unwanted reflection, and markers for Motion Analysis tracking.

1. Receive and log quadrotor states from Cortex over a TCP/IP connection,
2. Receive high-level control commands from a Matlab path planner,
3. Run position, velocity, and heading control loops,
4. Receive and log telemetry from the quadrotor, and

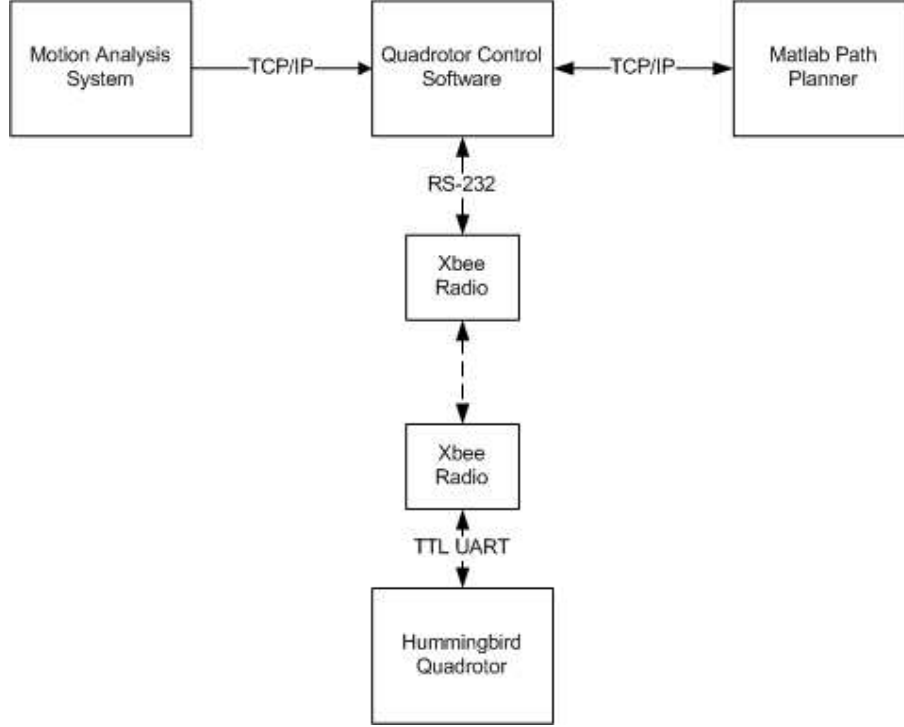


Figure 3.8: Flowchart for experimental setup.

5. Send control commands to the quadrotor.

Figure 3.8 shows a diagram of the complete system architecture used for testing.

3.4 Quadrotor Controller

The controller incorporates PID loops to regulate the position and velocity of the aircraft. Since the Hummingbird autopilot already incorporates an angle-hold controller, the position and velocity controllers running on the ground computer only need to compute desired pitch and roll angles. Since the Hummingbird is not equipped with a magnetic sensor, yaw angle (heading) is not automatically controlled. Rather, heading rate (measurable using onboard rate gyros) is controlled. The yaw command sent to the aircraft therefore specifies desired angular rate, while the pitch and roll commands specify desired angles.

A set of rough, “first-pass” gains were developed to keep the quadrotor flying. Later, system identification techniques were used to optimize the gains. The system identification process used to tune gains is described in Chapter 4.

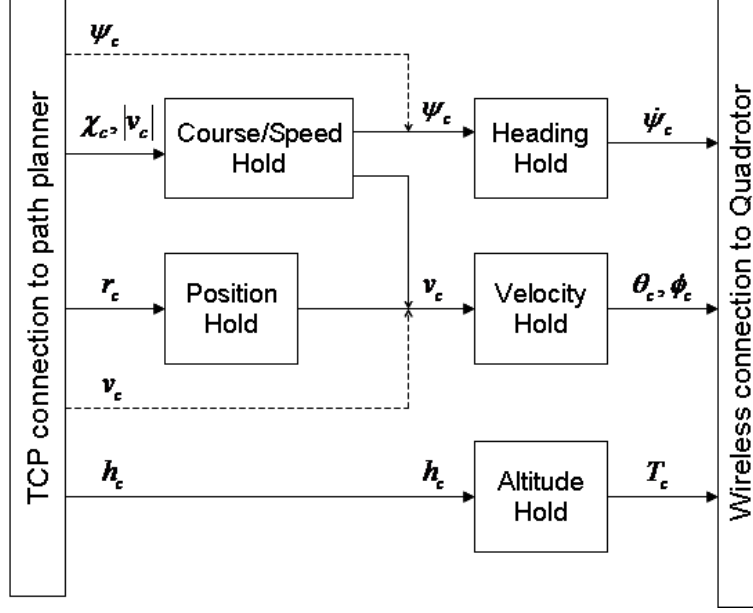


Figure 3.9: Block diagram of the controller used for the quadrotor aircraft. While not shown in the diagram, each block does have access to system states provided by Cortex.

The quadrotor controller can receive a variety of different inputs depending on the desired application, including position-hold, velocity-hold, and course/speed hold. Figure 3.9 provides an overview of the control structure. As shown, high level commands are sent to the controller via a TCP/IP connection. The action of the controller then depends on the type of control commands received. If a position hold command (r_c) is received, the position-hold controller generates a velocity command, which is then used compute desired pitch and roll angles. If a velocity command (v_c) is used, the position-hold controller is bypassed altogether. Finally, if a velocity/speed command ($\chi_c, |v_c|$) is provided, the velocity/speed controller produces a heading and velocity command, which are in turn sent to the velocity and heading controller blocks. The altitude hold loop operates independently from the other control loops. The connections from the path planner TCP block to v_c and ψ_c are dotted because they generally aren't used for control directly, though they can be used if desired. Usually, the position-hold or course/speed hold loops are used. Naturally, each control block has access to truth states provided by Cortex, also over a TCP block.



Figure 3.10: Hummingbird flying over a model city to test path planners for ground-target observability

The salient feature of the control architecture described here is that it is designed to be easy to use - a path planner and path follower that produces course and speed commands can be used integrated to the flight testbed seamlessly, provided that the commanded paths do not move the aircraft outside the observable volume. And because the code was written to establish the TCP connection through Matlab, many planners already used by the MAGICC lab can be implemented in hardware with minimal overhead.

To demonstrate the capabilities of the system, the orbit optimizing path planner described in Chapter 2 was implemented and tested. The course/speed controller was used, and the speed of the aircraft was fixed to simulate the behavior of a fixed-wing aircraft. A hypothetical city map was scaled to fit inside the observable volume, and model buildings were constructed out of foam. Figure 3.10 shows the Hummingbird aircraft flying above the model city.

3.5 Conclusion

This chapter describes an architecture that allow flight experiments to be performed using a 3D camera-based positioning system from Motion Analysis. While no specific theory

is developed, the system described will serve as the basis for a wide variety of experiments in the future. As a practical matter, many of the tools described in this chapter deserve additional explanation. Appendix A provides more specific details about using the various tools described in this chapter.

Chapter 4

System Identification for Quadrotor Aircraft

4.1 Introduction

Control design techniques allow a designer to select feedback gains to optimally control physical systems, but only when good models of the physical systems exist. For example, in Chapter 3 PID control loops were implemented to control the position and velocity of a quadrotor rotorcraft. If, during implementation, a reliable model existed to describe commanded to realized angle, the PID gains could have been selected precisely before flight. No accurate model existed at the time, however, so gains were necessarily tuned on the actual hardware. Tuning gains on a flying platform can be difficult because poor gains yield poor stability characteristics, which can lead to potentially destructive crashes. Once a set of acceptable gains has been found, it is no less difficult to ensure that they are optimal.

One way around this problem is to use system identification techniques to develop a suitable model around which to select control gains. In this chapter, system identification techniques are used to characterize the flight behavior of the Pelican and Hummingbird quadrotor platforms from Ascending Technologies. System identification is a well-developed field (See [15], [16], and [17]), and this chapter is not intended to present new research. Rather, this chapter documents the processes used to characterize aircraft flight performance using the testbed described in Chapter 3.

4.2 System Architecture

The general idea behind system identification is that the properties of a physical system can be discovered by observing its inputs and outputs. For example, the Pelican and the Hummingbird quadrotors have built-in angle-hold controllers. The input to the system is the angle command, and the output is the realized angle. This chapter develops a model

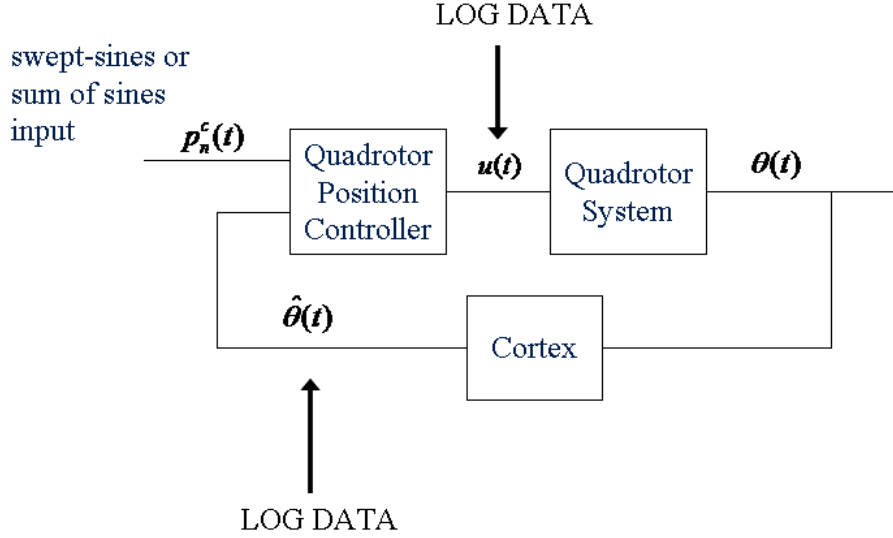


Figure 4.1: Flowchart describing system identification architecture. Position commands were fed into the existing position hold controller, which generated angle-hold commands. The angle-hold commands were recorded along with the angle outputs measured by Cortex.

to describe the relationship between the commanded angle and the realized angle. In this chapter, it is assumed that pitch, roll, and yaw dynamics are uncoupled. Only the pitch angle dynamics are investigated.

The testbed described in Chapter 3 provides a convenient way to collect data needed to do system identification. Pose data from Cortex is collected and logged at 200 Hz during flight, and control commands sent from the ground computer is logged as well. The architecture used for system identification is outlined in Figure 4.1. As shown, the existing position-hold controller is used to produce angle hold commands. This helps ensure that the aircraft does not fly outside the observable volume. The angle-hold commands generated by the position controller are logged along with the realized angle as measured by Cortex.

4.3 Input Selection

The input used for system identification must be sufficiently “rich” to provide enough information to fully characterize the system. In this chapter, two types of input are considered: a swept square wave, and a sum of sinusoids with various amplitudes and frequencies. Figures 4.2 and 4.3 show the frequency content of the swept square wave and sum of sines

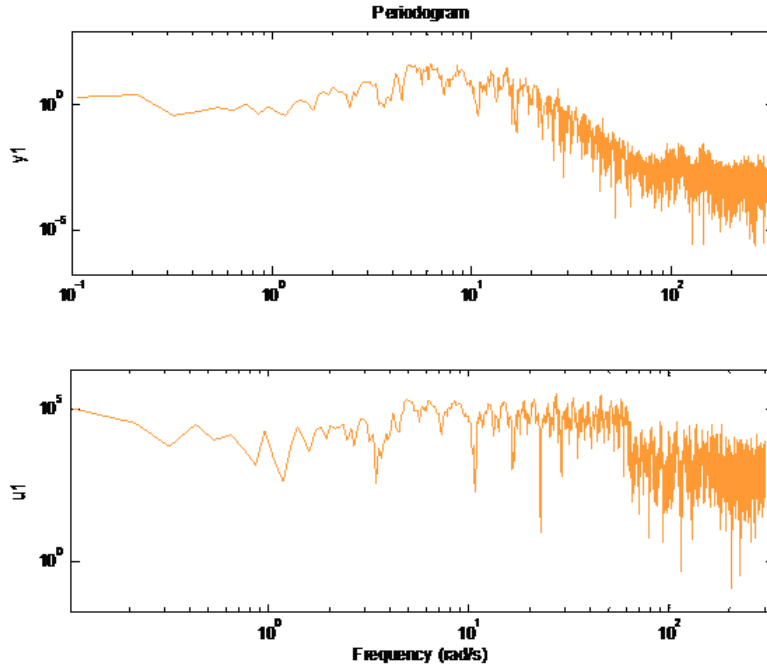


Figure 4.2: Frequency content of swept square wave input used for system identification

inputs, along with the frequency content of the measured output. Note that the selected inputs are not used to drive the commanded angle directly. Rather, the inputs are given to the position hold controller, which in turn generates the angle hold commands that are used for system identification. The system inputs are therefore generated indirectly.

It was thought that the swept square wave input would produce better data for system characterization because it contained a wider selection of input frequencies. However, it was discovered that the sum of sines input consistently yielded better models. One potential reason for this less intuitive result is that regardless of the selected input, high frequency plant behavior was never captured well by any of the modeling methods that were tried (where “high frequency” is, in this case, around just 20 Hz). This failure to characterize high-frequency plant behavior may stem from a number of factors, including potential nonlinearities in the plant, and probable lossy communication between the ground computer and the aircraft: lossy communication would have a larger impact when control signals are changing rapidly. In any event, the swept square-wave input contained much more power in

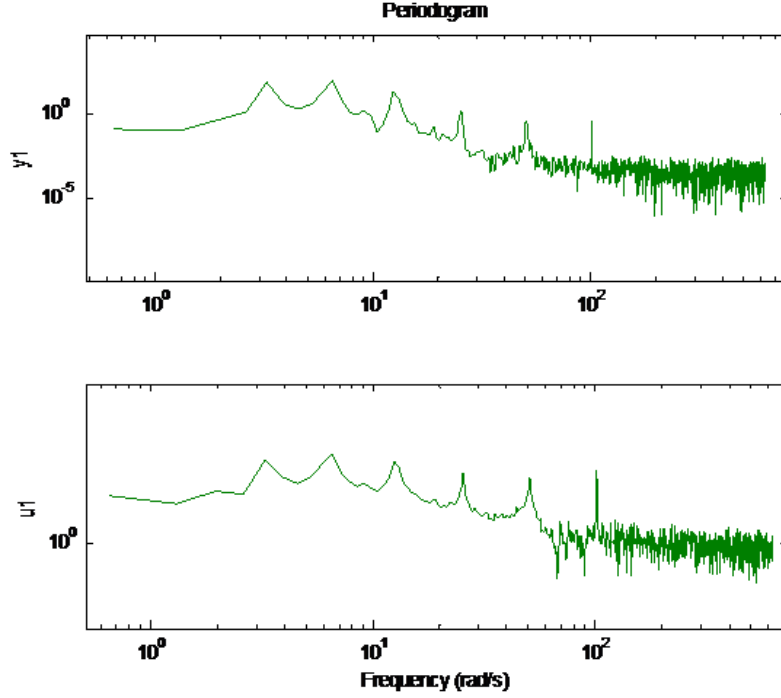


Figure 4.3: Frequency content of sum of sines input used for system identification

the high frequency band than the sum of sines input, so the failure of the swept square-wave input to produce useful models is not at all surprising.

4.4 Models for Commanded to Realized Angle

After system input and output data was collected, Matlab's System Identification Toolbox was used to produce system models for both the Pelican and Hummingbird. A variety of model types were used, including continuous frequency domain models, ARX, and ARMAX models. In general, low-order models fit the plant behavior more accurately than high-order models. A variety of datasets were collected for model identification and verification, and the best fit models depended on the specific datasets used. For most datasets, a single-pole frequency-domain model fit the data most accurately. The best fit model for commanded to realized angle on the Hummingbird is given by

$$H_{\theta}(s) = \frac{1}{0.38143s + 1}, \quad (4.1)$$

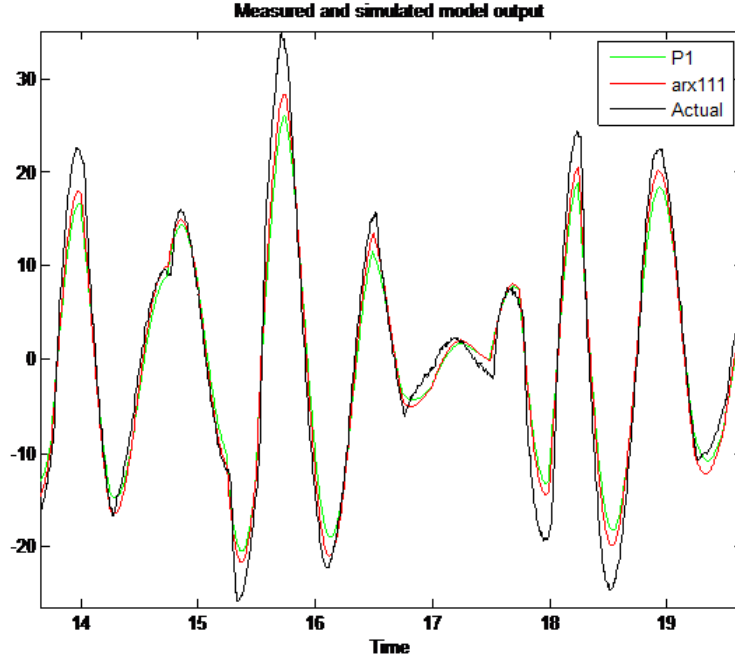


Figure 4.4: Comparison of actual system behavior with predicted system behavior for pitch control on the Hummingbird aircraft. The label 'P1' corresponds to a the output generated by a single-pole transfer function model, while the 'arx111' label corresponds to an ARX model.

and the best fit model for the Pelican is given by

$$H_{\theta}(s) = \frac{1}{0.47911s + 1}. \quad (4.2)$$

Because both aerial platforms are symmetric, the same models can be used for roll. Yaw dynamics were not characterized. A comparison of model output and actual output for the Hummingbird is given in Figure 4.4.

4.5 An Expanded Model for Position Hold

The angle dynamics identified in Section 4.4 are more useful when combined with the position hold dynamics of the aircraft, as shown in Figure 4.5. By incorporating position-hold dynamics and angle-hold dynamics, it is possible to select gains for position and velocity control more intelligently.

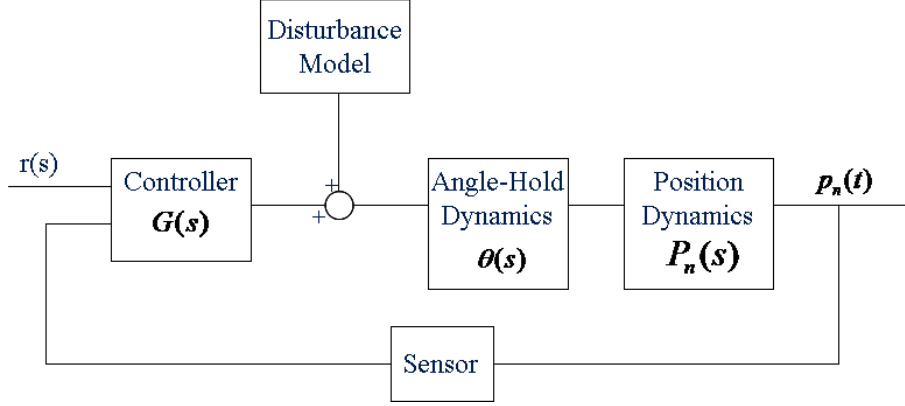


Figure 4.5: Quadrotor control model, including both angle and position dynamics

The derivation of the relationship between angle and position is straightforward. Using Figure 4.6 as a reference, and assuming that the thrust of the aircraft is high enough to maintain constant altitude, we have

$$F_T = mg / \cos(\theta), \quad (4.3)$$

where m is the mass of the aircraft, F_T is the total thrust, and θ is the pitch angle. It follows that

$$F_X = mg \tan(\theta), \quad (4.4)$$

and finally that

$$a_x = g \tan(\theta). \quad (4.5)$$

Using the small angle approximation, we have

$$a_x \approx g\theta, \quad (4.6)$$

which leads to the transfer function

$$P_n(s) = \frac{g}{s^2}. \quad (4.7)$$

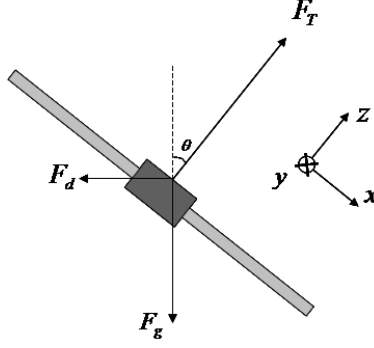


Figure 4.6: Free-body diagram of quadrotor aircraft

The transfer function from commanded pitch angle to quadrotor position is therefore given by

$$\frac{P_n(s)}{\theta_c(s)} = \frac{g}{s^2} H_\theta(s), \quad (4.8)$$

where $H_\theta(s)$ is the transfer function from commanded to realized angle identified in Section 4.4. Note that this model ignores the effects of drag, predicting that a constant non-zero pitch angle will result in unbounded velocity and position. This is, of course, not accurate, but for low velocity flight near hover the assumption is reasonable. Additional consequences of this assumption in the context of state estimation are described in Chapter 5.

In practice, the small-angle approximation is useful for finding control gains, but it is only valid for a small set of angles close to zero. In fact, using Equation 4.8 to describe the plant in simulation resulted in instability using the same gains the actual physical system used. This will be discussed in more detail later.

4.6 Simulation-based Gain Tuning

To facilitate gain tuning, a simulation was developed that included the angle to position dynamics of the quadrotor, and that incorporated a controller written exactly like the one already running on the motion capture testbed. The idea was to produce a simulation that resembled reality close enough that gains could be selected, tested, and tuned without putting the flight platform at risk. The controller running on the physical platform is described in detail in Chapter 3. To summarize briefly, there are two PID loops running on the system,

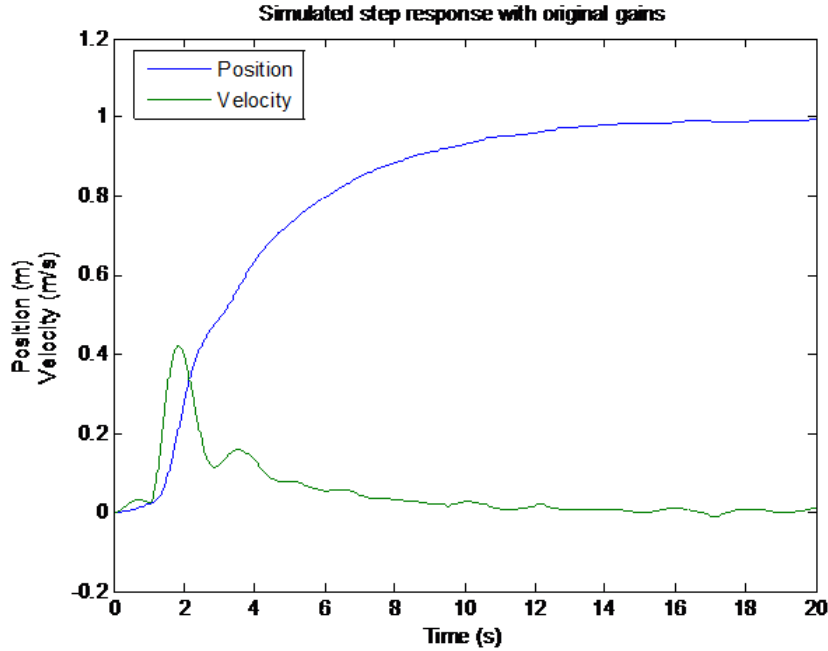


Figure 4.7: Simulated step response with original, untuned gains

one for position hold, and one for velocity hold. The position hold controller accepts a desired position and generates a velocity command. The velocity hold controller accepts a desired velocity and generates an angle command, which is then sent to the aircraft. The controller thus consists of six gains - proportional, integral, and derivative gains for both control loops.

Figure 4.7 shows the simulated position/velocity response of the system using the original, untuned gains running on the position-hold controller. Figure 4.8 shows the step response after the gains were tuned in simulation. The tuned gains from simulation were applied to the actual position controller and tested without modification. The results are shown in Figure 4.9. As shown, the actual system response closely mirrors the predicted response.

At the end of Section 4.5, it was mentioned that the small-angle approximation failed to produce results comparable to the physical system. This was true even when the input was simply a one meter step in position. The nonlinear system model used to produce Figure 4.9 produced accurate results. However, simply replacing the nonlinear model with

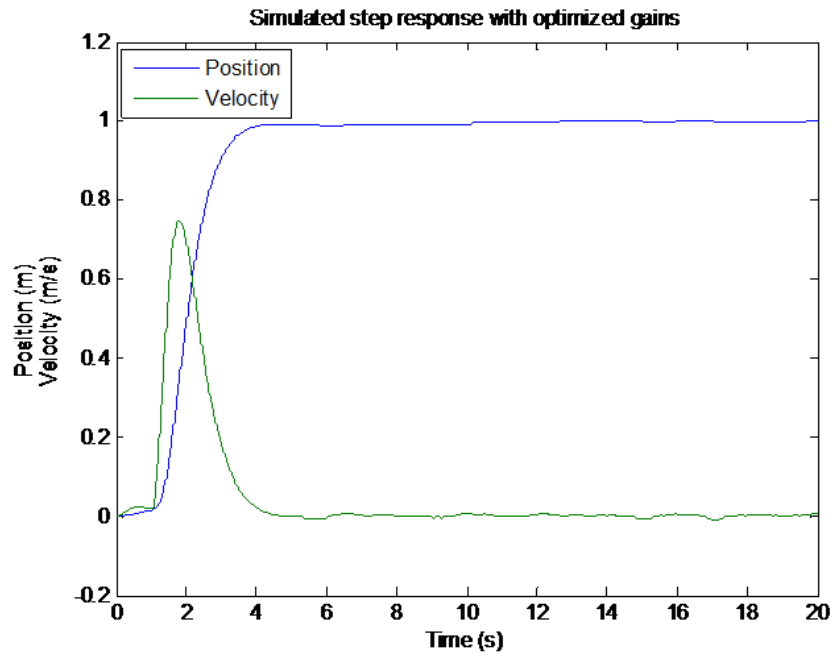


Figure 4.8: Simulated step response with tuned gains

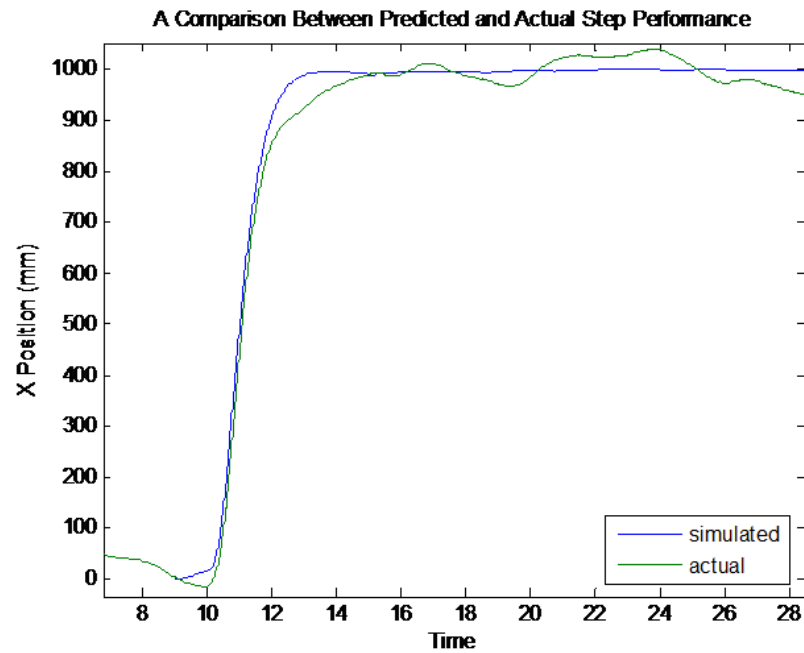


Figure 4.9: Predicted vs. actual step response of the rotorcraft using gains optimized in simulation

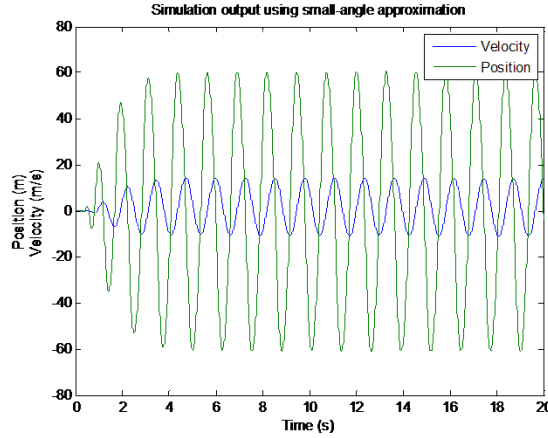


Figure 4.10: System output using actual gains for control and the small-angle approximation in the plant simulation. The system is unstable, but the output is bounded because the control inputs are saturated.

the linear model obtained using the small-angle approximation caused the simulated response to become unstable. This is shown in Figure 4.10. This suggests that the small-angle approximation is useful only as a starting point for gain selection, and that more iteration is required.

4.7 Conclusion

Without accurate knowledge of a physical system, it is difficult to select good control gains without iteratively implementing different sets of gains on the actual system. For a flying platform, this can be problematic because poor gains can cause the system to lose stability and crash, potentially damaging the aircraft. Even when stable gains have been implemented, it is still a time-consuming task to optimize the gains because it involves testing and retesting new gains on the aircraft.

By characterizing the dynamics of the quadrotor, the iterative guess-work in selecting control gains can be completely removed. In this chapter, quadrotor dynamics were identified and a simulation was developed to allow gains to be selected, implemented, and tuned without putting the aircraft at risk.

Chapter 5

Using Accelerometers for Rotorcraft State Estimation

5.1 Introduction

Before a controller can perform any useful function, the physical states (or some subset of the physical states) of the system to be controlled must be estimated from measurements. For example, a stabilizing control loop for a quadrotor must know the attitude of the aircraft if the attitude is to be regulated. Typically, the attitude of the aircraft is not directly measureable. An observer, such as a Kalman Filter, is usually used to combine data from a variety of sensors to produce attitude estimates.

A common and intuitive approach for pitch and roll estimation is to combine data from rate gyros and accelerometers. The fundamental idea is that in the absence of other accelerations, the accelerometers can measure the gravity vector, which can, in turn, be used to back out pitch and roll angles. In the short term, however, other accelerations interfere with measurement of the gravity vector so that angle estimates based solely on accelerometers are unreliable. The usual solution is to complement the accelerometer data with data from rate gyros. By integrating measured angular rates, angle estimates can be produced that are less susceptible to errors from transient accelerations. While gyro-based angle estimates do drift over time, the combination of accelerometers and rate gyros ideally provides the best of both worlds. Rate gyros are more immune to the effect of transient accelerations, while accelerometer-based angle estimates do not typically get worse as time progresses.

In practice, rate gyro and accelerometer data can be combined to estimate attitude using a variety of methods, from a simple complementary filter [18], to a full Extended Kalman Filter ([19], [20]). Regardless of the method used for estimation, however, the intuitive description of the estimation method given above still applies if it is assumed that accelerometers can measure the gravity vector directly.

This intuitive approach to attitude estimation is only valid when the accelerometers usually measure something close to the gravity vector. On rotorcraft, however, this is not the case. In fact, for a rotorcraft near hover, the expected accelerometer-based angle estimate should always be close to zero regardless of the actual attitude [13]. Despite this fact, many quadrotor aircraft use an accelerometer/gyro attitude estimation scheme without obvious problems (see [7], [8], [9], [10], [11], and [12], for example). In [13], Martin et al. describe why these erroneous assumptions about accelerometer behavior on rotorcraft produce satisfactory results. A modified controller is also presented that improves performance by accounting for the actual behavior of accelerometers on rotorcraft.

In this chapter, an alternative method of improving the attitude estimate is presented that models accelerometer-based attitude estimates as low-pass filtered versions of the actual angles. Experimental data demonstrates that the model captures the essential information, and a modified EKF is developed that specifically incorporates the model to improve accuracy.

5.2 Accelerometer Model

The output of a three-axis accelerometer mounted to a rigid body can be modeled as

$$\hat{\mathbf{a}} = \frac{1}{m} (\mathbf{F} - \mathbf{F}_{\mathbf{g}}), \quad (5.1)$$

where $\hat{\mathbf{a}}$ is the measured acceleration, m is the mass of the body, \mathbf{F} is the sum of all forces on the body (including gravity), and $\mathbf{F}_{\mathbf{g}}$ is the force due to gravity.

The accelerometer model in Equation (5.1) can be intuitively explained using the accelerometer diagram shown in Figure 5.1. As shown, an accelerometer can be constructed by attaching a proof-mass to a lever-arm. Any upward or downward acceleration of the sensor in Figure 5.1 causes deflection of the proof mass, which can be measured to determine acceleration (below the natural frequency of the lever/mass system, deflection is proportional to acceleration). This accounts for the first force term, \mathbf{F} , in Equation (5.1): the sum of all forces on the body, including gravity, produces physical acceleration of the sensor, which in turn causes deflection of the proof mass.

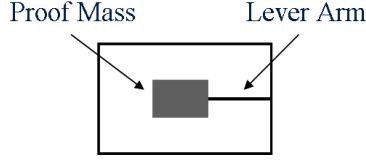


Figure 5.1: Simplified diagram of an accelerometer. Acceleration is measured by detecting deflections in the lever arm.

The second force term, \mathbf{F}_g , is present in the accelerometer model because the force of gravity not only accelerates the sensor body, it also causes deflection of the proof-mass itself. If the accelerometer is not accelerating ($\mathbf{F} = 0$), then gravity produces a downward deflection of the proof mass that appears equivalent to upward acceleration of the sensor at the acceleration of gravity. Similarly, if the accelerometer is in free-fall ($\mathbf{F} = \mathbf{F}_g$) there is no deflection and hence no measured acceleration.

The accelerometer model in Equation (5.1) suggests that the expected accelerometer measurement can be obtained by drawing a free-body diagram that includes all forces except the force of gravity: the first term, \mathbf{F} , includes gravity, while the second term, \mathbf{F}_g , removes it. From this perspective, accelerometers never measure gravity directly - they measure the forces that prevent the sensor from accelerating toward the center of the Earth (in addition to other external forces).

If $\mathbf{F} \approx 0$, then the predicted accelerometer output is given by

$$\hat{\mathbf{a}} = -\frac{1}{m}\mathbf{F}_g, \quad (5.2)$$

from which it is easy to determine pitch and roll angles. The assumption that $\mathbf{F} \approx 0$ is widely used for state estimation on quadrotor platforms. Interestingly, on a rotorcraft, this assumption is not valid. Consider the free-body diagram of a single axis of a quadrotor helicopter shown in Figure 5.2. Thrust, which is always in line with the body frame z axis, is given by F_T . The force of gravity and aerodynamic drag are represented by F_g and F_d , respectively. In [13], Martin and Salaun show that the aerodynamic drag acting on a

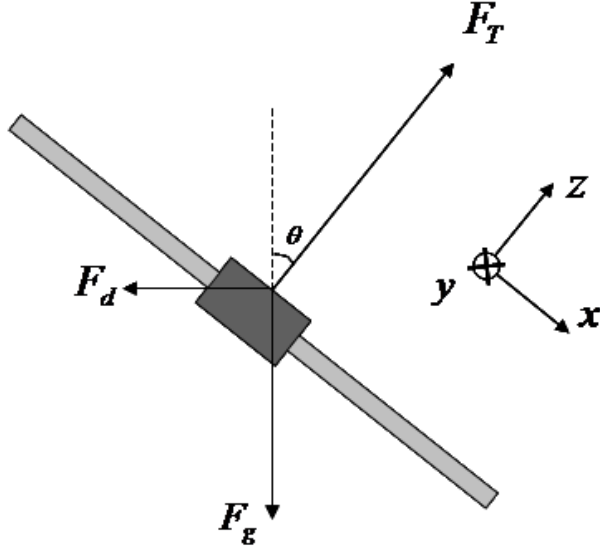


Figure 5.2: Simplified free-body diagram for a quadrotor helicopter

quadrotor helicopter is proportional to the body frame velocity of the aircraft, or

$$\mathbf{F}_d = -b \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad (5.3)$$

where b is a constant, u is the body frame velocity of the aircraft along the x -axis, and v is the body frame velocity along the y -axis, and w is the body-frame velocity along the z -axis.

Assuming that thrust is set to maintain constant altitude, the body-frame thrust vector is given by

$$F_T = \begin{pmatrix} 0 \\ 0 \\ \frac{mg}{\cos(\phi)\cos(\theta)} \end{pmatrix}. \quad (5.4)$$

Combining all known forces into Equation (5.1), the expected accelerometer measurement is given by

$$\hat{\mathbf{a}} = \frac{1}{m} \begin{pmatrix} -bu \\ -bv \\ -F_T - bw \end{pmatrix}. \quad (5.5)$$

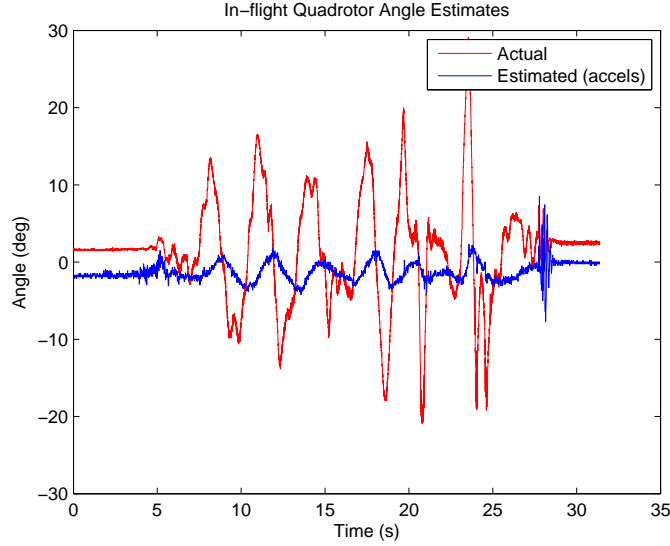


Figure 5.3: Accelerometer-based angle estimates on the Hummingbird quadrotor

In practice, the constant b is small so that for low velocity flight,

$$\hat{\mathbf{a}} \approx \frac{1}{m} \begin{pmatrix} 0 \\ 0 \\ -F_T \end{pmatrix}. \quad (5.6)$$

Referring back to Equation (5.1), the assumption that $\mathbf{F} \approx 0$ is clearly inappropriate. In fact, for low-velocity flight on a quadrotor, the x and y components of measured acceleration are expected to be close to zero. In other words, the gravity vector is not being measured, and the accelerometer-based attitude estimate will be close to zero. To demonstrate this behavior, accelerometer-based attitude estimates on a quadrotor were compared with actual attitude using the testbed described in Chapter 3. Figure 5.3 shows the results. Note that the accelerometer-based estimates do not approach the actual angles. Non-zero angle estimates result because the accelerometers start to measure the effects of aerodynamic drag at higher velocities.

It is, of course, overly simplistic to assume that measurements along the x and y accelerometer axes will always be zero. Given a constant non-zero angle, the aircraft will

accelerate until it reaches some steady-state velocity, at which point \mathbf{F} will actually approach zero, and the correct angle will be directly measureable. The interesting question to consider is how long it will take for the accelerometers to measure the correct angle, and what the angle output of the accelerometers will be in the interim.

Since the x and y accelerometers measure scaled versions of the body frame velocities u and v , x and y axis accelerometer behavior can be characterized by considering the relationship between attitude and body-frame velocity. For this analysis, it is assumed that the transfer functions from attitude to velocity for each axis are decoupled (a valid assumption for small angles). It is also assumed as before that thrust is high enough to maintain constant altitude. The actual acceleration of the aircraft along the body-frame x -axis is given by

$$a_x = g \sin(\theta) - \frac{bu}{m} \cos(\theta). \quad (5.7)$$

Using the small-angle approximation, the x -axis acceleration becomes

$$a_x = g\theta - \frac{bu}{m}. \quad (5.8)$$

Taking the Laplace transform, we get

$$sU(s) = g\theta(s) - \frac{b}{m}U(s), \quad (5.9)$$

which gives the transfer function

$$\frac{U(s)}{\theta(s)} = \frac{\frac{mg}{b}}{\frac{m}{b}s + 1}. \quad (5.10)$$

Finally, by combining Equations (5.10) and (5.5), the transfer function relating the measured accelerometer output on the x -axis to the angle θ is obtained as

$$\frac{A_{m,x}(s)}{\theta(s)} = -\frac{g}{\frac{m}{b}s + 1}. \quad (5.11)$$

Equation (5.11) describes a single-pole low-pass filter with gain g and time constant $\tau = m/b$. Higher aerodynamic drag increases b , thereby decreasing the time constant. This is as expected: higher drag reduces the amount of time it takes to reach steady-state velocity.

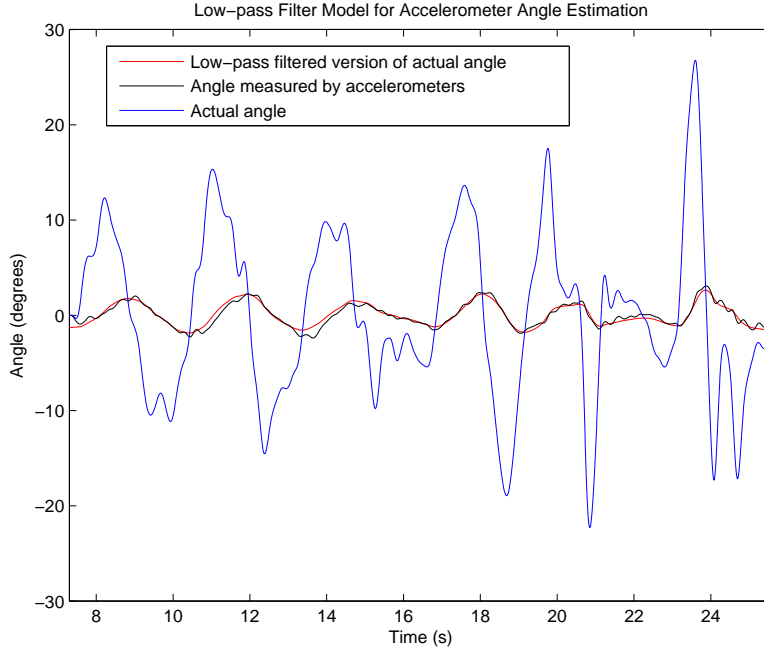


Figure 5.4: A comparison of actual quadrotor pitch angles compared to accel-based estimates and a low-pass filtered version of the actual angles. This figure demonstrates that a low-pass filter model accurately describes accel-based attitude estimates on rotorcraft.

This analysis suggests that accelerometer-based attitude estimates should look like a low-pass filtered version of the actual angles, with the corner frequency set by the aerodynamic drag coefficient and mass of the aircraft.

To confirm this idea, accelerometer telemetry was logged along with truth data from the flight testbed. Accelerometer-based attitude estimates were then computed, and a low-pass filter model was fit to the data. The resulting transfer function from the actual angle to the measured angle is given by

$$\frac{\theta_m}{\theta} = \frac{1}{2.775s + 1}. \quad (5.12)$$

Figure 5.4 shows the results. As shown, the low-pass filter model describes actual angle outputs very well.

The main implication of this analysis is that accelerometers cannot be used to capture high frequency attitude changes on a rotorcraft. Lower frequency attitude information, on

the other hand, can be measured directly. For example, if the aircraft maintained a constant non-zero pitch angle to hover in wind, that pitch angle would be measured correctly by the accelerometers. Other deviations in angle caused by disturbances would go unnoticed. This is why rotorcraft attitude estimators that use accelerometers directly tend to work well despite the poor assumption about accelerometer behavior. Accelerometer-based correction keeps the attitude estimates in the right ball-park because the average attitude is actually measured correctly.

There are, however, interesting performance implications associated with using accelerometers directly. These implications will be described in the next section.

5.3 Estimator Design

As described in Section 5.1, an intuitive approach to attitude estimation is to combine rate gyros and accelerometers to take advantage of the relative strengths of each type of sensor. Rate gyro measurements are integrated to provide short-term estimates that are insensitive to acceleration, while accelerometer measurements are used to gradually compensate for drift in the gyro-based estimates. In practice, a variety of methods can be used to combine accelerometer and rate gyro data for attitude estimation. In this section, a Kalman Filter is used. Two filters are designed. In the first, it is assumed that non-gravity accelerations measured by the accelerometers are minimal ($\mathbf{F} = 0$), and that the accelerometers can therefore be used to measure attitude directly. Simulation results demonstrate that this method can cause appreciable angle estimate errors when operating onboard a rotorcraft. The second filter specifically accounts for the expected accelerometer behavior on rotorcraft. Simulation results demonstrate that better performance can be obtained with the improved model.

5.3.1 Background

Let the quadrotor body-frame axes be a set of orthogonal axes attached to the body of the rotorcraft, with the x -axis aligned with the front rotor, the y -axis aligned with the right rotor, and the z -axis pointing down. The attitude of the aircraft is represented with a set of angles describing rotation from an inertial frame to the body-frame of the aircraft.

Yaw represents rotation about the z -axis by an angle ψ , pitch represents rotation about the y -axis by an angle θ , and roll represents rotation about the x -axis by an angle ϕ . All rotations are right-handed. The sequence of rotations used to move from the inertial to the body frame is first yaw, then pitch, then roll.

Let p , q , and r represent the angular rates about the body-frame x , y and z axes, respectively. Since the rate gyros are (ideally) aligned with the rotorcraft body-frame axes, p , q , and r can be measured directly with the rate gyros. The transformation from body-frame angular rates to $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ is given by

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \quad (5.13)$$

Given measurements of p , q , and r from the rate gyros, the attitude estimate can be computed over time by integrating $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ from Equation (5.13).

Equation (5.13) is nonlinear, so that an Extended Kalman Filter would be required for estimation. For simplicity, it is assumed that motion only occurs about the y -axis (pitch). Given this restriction, the equation relating body-frame angular rates to $\dot{\theta}$ becomes simply

$$\dot{\theta} = q. \quad (5.14)$$

Equation (5.14) will be used for state prediction in both estimators presented in this chapter.

In the next section, an attitude estimator is developed that uses accelerometers directly. The estimator is tested in simulation using a quadrotor system model determined experimentally using the system described in Chapter 3.

5.3.2 Naive Estimator Design

Assuming that external non-gravity forces are negligible and that roll and yaw angles are fixed at zero, the three-axis accelerometer measurement is given by

$$\mathbf{a}_m = \begin{pmatrix} g \sin \theta \\ -g \cos \theta \\ -g \cos \theta \end{pmatrix}. \quad (5.15)$$

The accelerometer-based pitch angle estimate, θ_{acc} can thus be computed using

$$\theta_{acc} = \arcsin \frac{a_x}{g}. \quad (5.16)$$

Let $\hat{\theta}[k]$ be the estimate of the pitch angle θ at the discrete time interval k . The value of $\hat{\theta}[k]$ is estimated when new data from rate gyros and accelerometers is received. Using Equations (5.14) and (5.16), the pitch angle estimator is given by

$$\hat{\theta}[k]^- = \hat{\theta}[k-1] + Tq_{gyro}[k-1], \quad (5.17)$$

$$\hat{\theta}[k] = L(\theta_{acc}[k] - \hat{\theta}[k]^-), \quad (5.18)$$

where $\hat{\theta}[k]^-$ is the angle estimate before accelerometer-based correction is applied, $q_{gyro}[k]$ is the angular rate measured by the y -axis rate gyro at time index k , T is the sampling period of the sensors, $\theta_{acc}[k]$ is the angle measured by the accelerometers at time interval k , and L is an observer gain. In this case, the observer gain is computed using a Kalman Filter. The term $Tq_{gyro}[k]$ integrates gyro measurements to produce angle estimates, while the term $L(\theta_{acc}[k] - \hat{\theta}[k]^-)$ corrects the angle estimate based on accelerometer measurements. From an intuitive standpoint, a high value of L weights the accelerometers heavily, while a low value of L weights the rate gyros more heavily.

As mentioned at the end of Section 5.2, this estimation method should work tolerably well despite the fact that the accelerometers should only measure low-frequency components of the aircraft attitude. However, as will be shown, there are performance limitations. Consider, for example, the form of the accelerometer correction term in Equation (5.18).

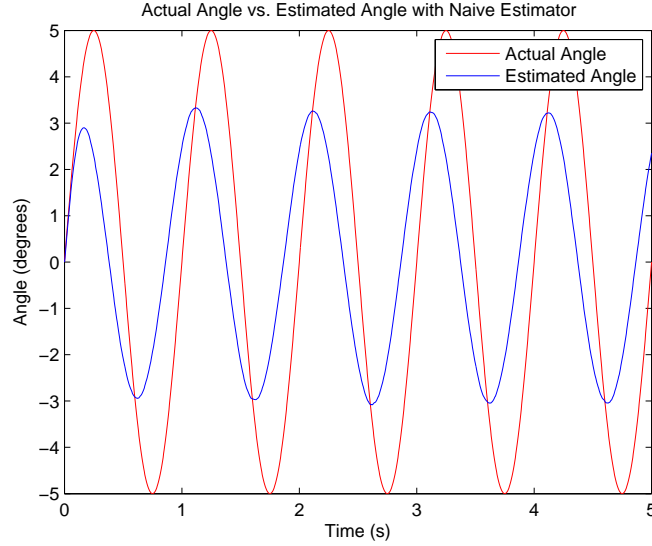


Figure 5.5: Actual pitch angle vs. estimated pitch angle using accelerometers for state correction directly.

Because $\theta_{acc}[k]$ does not include high-frequency data that will be included in $\hat{\theta}[k]$, the angle estimate will be erroneously pulled toward the average attitude. Note that in the case of a small quadrotor, high-frequency is really not that high at all. In [13], Martin and Salaun show that the drag coefficient for their small rotorcraft was only 0.25 s^{-1} . This results in a low-pass filter with a time constant of six seconds for a quadrotor of mass 1.5 kg. Thus a sinusoid in the pitch angle with a period of several seconds will still not be captured properly by the accelerometers, and the accel-based correction will therefore cause the filter to consistently underestimate the magnitude of the sinusoid.

This behavior was confirmed via simulation as shown in Figure 5.5. In the simulation, the aircraft pitch angle was excited with a 1 Hz sinusoid with an amplitude of 5 degrees. As shown, the estimated state lags the actual state significantly, and the magnitude of the estimated pitch angle is always lower than that of the actual pitch angle. The amount of error resulting from this use of accelerometers can be reduced by weighting the filter differently: By trusting the rate gyros more and the accelerometers less, error is minimized. However, weighting the gyros more tends to accentuate the effect of non-zero biases on rate gyro output, which are a significant contributor to long-term angle estimate error.

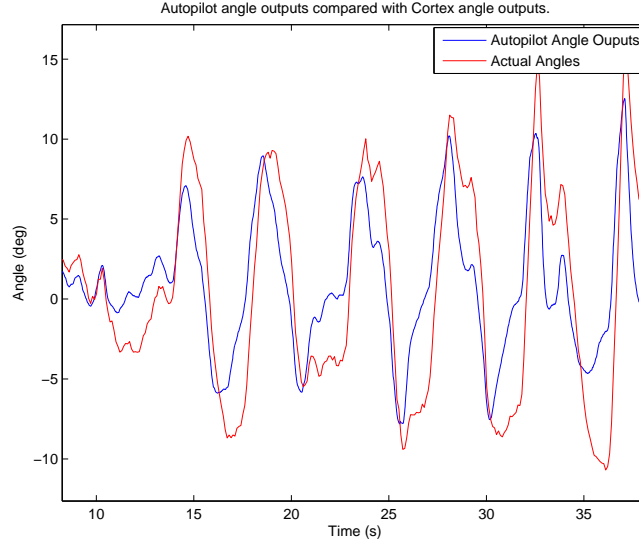


Figure 5.6: Angle outputs from the Hummingbird autopilot compared to actual angles obtained using Cortex.

This same behavior is evident on hardware as well. The Hummingbird quadrotor from Ascending Technologies incorporates an onboard autopilot that handles attitude estimation. Autopilot telemetry and truth data from Cortex was logged and compared. While the details about the state estimator running on the autopilot are not published, the angle output errors are consistent with the type of estimator described in this section. As shown in Figure 5.6, the amplitude of the estimated angle was consistently underestimated by the autopilot. The errors are less substantial than the errors shown in Figure 5.5, and the phase lag in the estimate is almost non-existent, suggesting that rate gyros are trusted more heavily. The errors are nevertheless significant.

For slow flight near hover, these kinds of angle estimate errors will be minimal and may be acceptable. However, in applications where precision telemetry is needed (i.e., for sensor registration), an alternative state estimation method must be used.

5.3.3 Estimator Incorporating Low-pass Filter Model

In this section, an estimator is presented that improves performance by explicitly accounting for accelerometer behavior on rotorcraft. Using the low-pass filter model described

in Section 5.2, the transfer function describing the relationship between the measured angle and the actual angle is given by

$$\frac{\theta_{acc}}{\theta} = \frac{1}{\tau s + 1}, \quad (5.19)$$

where τ is the time constant determined by the mass and the drag coefficient of the aircraft. Let $\hat{\theta}_{acc}$ be the expected accelerometer-based angle estimate, computed by filtering angle estimates $\hat{\theta}$. Using the bilinear transform, a digital implementation of Equation (5.19) can be used to estimate $\hat{\theta}_{acc}$ as

$$\hat{\theta}_{acc}[k] = C_1 \hat{\theta}[k] + C_1 \hat{\theta}[k-1] + C_2 \hat{\theta}_{acc}[k-1], \quad (5.20)$$

where T is the sample period,

$$C_1 = \frac{T}{2\tau + T}, \quad (5.21)$$

and

$$C_2 = \frac{2\tau - T}{2\tau + T}. \quad (5.22)$$

Computation of the expected accelerometer angle measurement therefore depends on the most recent angle estimate, $\hat{\theta}[k]$, one previous angle estimate, $\hat{\theta}[k-1]$, and the previous expected accelerometer output, $\hat{\theta}_{acc}[k-1]$.

The pitch angle observer accounting for the low-pass filtered nature of accelerometer-based angle estimates is given by

$$\hat{\theta}[k]^- = \hat{\theta}[k-1] + T q_{gyro}[k-1], \quad (5.23)$$

$$\hat{\theta}_{acc}[k] = C_1 \hat{\theta}[k]^- + C_1 \hat{\theta}[k-1] + C_2 \hat{\theta}_{acc}[k-1], \quad (5.24)$$

$$\hat{\theta}[k] = \hat{\theta}[k]^- + L(\theta_{acc}[k] - \hat{\theta}_{acc}[k]), \quad (5.25)$$

where $\hat{\theta}[k]^-$ is the angle estimate before accelerometer-based correction is applied. As in the previous observer, the observer gain L is computed using a Kalman Filter.

This estimation method removes transient errors from state estimates by explicitly accounting for the influence of quadrotor dynamics on acceleration measurements. Simula-

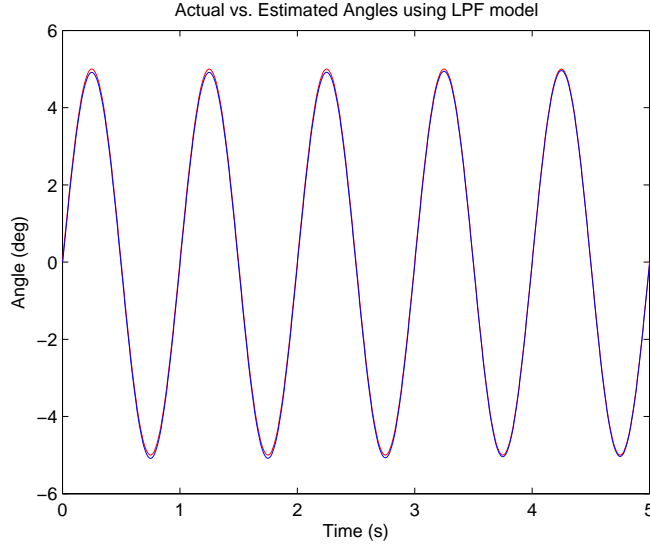


Figure 5.7: Actual pitch angle vs. estimated pitch angle using LPF accelerometer model

tion results of this estimator are shown in Figure 5.7. Note that using the low-pass filter model reduced the magnitude and phase of the error significantly.

5.3.4 Hardware Demonstration of Attitude Estimators

The estimators described by Equations (5.17)-(5.18) and Equations (5.23)-(5.25) were implemented in hardware to evaluate their performance on a real-world system. To facilitate comparison, the two estimators operated simultaneously on one quadrotor platform with access to the same in-flight gyro and accelerometer data. The output of both estimators was logged along with truth data from Cortex.

The autopilots onboard the Hummingbird and the Pelican quadrotors are capable of streaming raw sensor data that could conceivably be used for off-board state estimation. However, the data-rates achievable over the autopilots' UART interfaces were too low to be practical for state estimation. The estimators needed to run onboard the aircraft using higher-bandwidth sensor data. The Hexacopter aircraft from Mikrokopter provided a viable alternative because the autopilot firmware is open-source and could be modified to incorporate custom filters. In the interest of simplicity, however, it was decided to implement

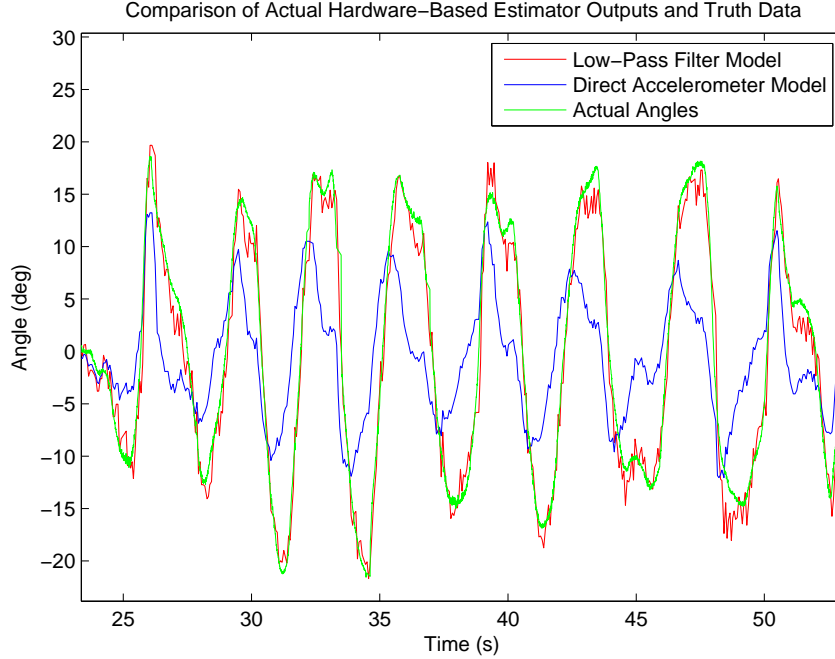


Figure 5.8: A comparison of estimation methods on a flying rotorcraft. The direct accelerometer method results in poor performance due to the neglected dynamics of the rotorcraft. The estimation method that accounts for aircraft dynamics produces results that are consistent with truth data.

the custom estimators on an independent device to avoid interfering with the control code already operating on the aircraft.

The estimators were therefore implemented onboard the UM6 Orientation Sensor from CH Robotics. Since the UM6 firmware is open-source, it was straightforward to write new estimation algorithms using data from its onboard sensors. Attitude estimates were computed using both estimators simultaneously. The attitude estimates were then transmitted by the UM6 and logged along with truth data from Cortex. The results are shown in Figure 5.8

The direct accelerometer estimation method produces significant errors in the attitude estimates. As discussed, this behavior occurs because the accelerometers are incapable of measuring higher-frequency attitude information and therefore erroneously pull the estimates toward the average attitude. On the other hand, the estimator considering the effect of aerodynamic drag produces estimates that are much more accurate.

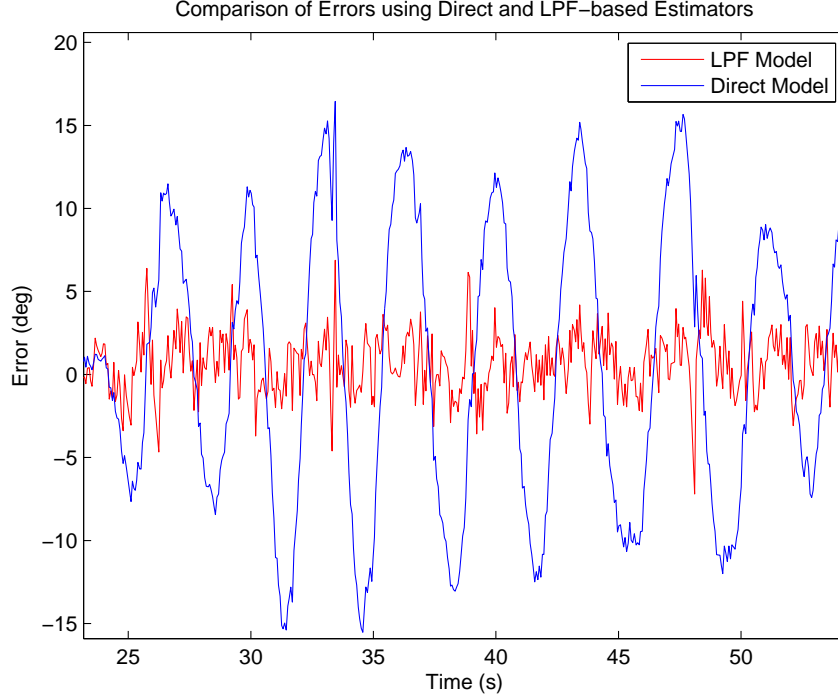


Figure 5.9: Angle estimation errors using accelerometers directly compared to errors resulting when considering the effect of aerodynamic drag

Figure 5.9 compares the errors obtained using the direct estimation method with errors obtained using the estimator taking into account aerodynamic drag. The average magnitude of the error for the direct method was 4.25 degrees, while the average error magnitude for the low-pass filter method was 1.41 degrees, representing a 66.8% improvement. Note that the UM6 is a low-grade orientation sensor. Better results could be obtained using higher-end, well-calibrated sensors.

5.4 Conclusions

In this section, accelerometer behavior on rotorcraft was described. A low-pass filter model of accelerometer behavior was developed, and two state estimation methods were presented. In the first method, the estimator erroneously assumed that accelerometers measured the gravity vector. It was shown that because of this poor assumption, angle estimates were inaccurate. In the second method, the low-pass filter model was incorporated into the

estimator, resulting in much higher accuracy. Both estimators were implemented in hardware and their outputs were compared with actual angles. It was shown that the low-pass filter model produces much more reliable estimates.

While much of the work in this chapter was addressed by Martin and Salaun in [13], this chapter provided a more detailed description of the characteristics of the errors introduced through improper use of accelerometers. This chapter also presented an alternative method for using accelerometers properly. The observer presented in this chapter highlights the filtered nature of accelerometer-based attitude estimates, and provides an intuitive description of what is happening on the hardware.

Chapter 6

Conclusions and Future Work

6.1 UAV Path Planning for Ground Target Observability

6.1.1 Overview and Contributions

In Chapter 2, a model was developed to describe the effect of occlusions on ground target observability. Using this occlusion model, two different path planners were developed. In the first, paths represented using continuous-time parametric functions were optimized to maximize the probability that the aircraft will maintain line of sight with the ground target. Any parametric path can be optimized using the planner. Two different path shapes were described: a simple orbital path, and a canyon-following path.

The second path planner operates using a simulated chain inside a force field. The force-field is defined using the occlusion model to force the chain into regions of high detection probability. By adding constraint forces, the path formed by the chain remains flyable by a fixed-wing UAV.

The occlusion model and path planners developed in Chapter 2 are unique in that they specifically account for the presence of occlusions, and that they allow for a much broader set of possible paths than existing methods for path planning with occlusions.

6.1.2 Future Work

The parametric planner described in Chapter 2 could easily be extended to describe other path shapes like figure eights and ellipses. By expanding the observability model into a third dimension, full 3D paths could also be optimized using a methodology similar to the one described in the chapter.

The chain-based planner could be similarly modified to work in three dimensions. It could also be modified to allow multiple UAVs to plan complementary paths for observing ground targets - by adding another chain to the planner and forcing the chains to repel each other, multiple UAVs could be pushed into paths that provide a high degree of coverage of the area occupied by the target.

6.2 Motion Capture Testbed for Indoor Aircraft Navigation and Control

6.2.1 Overview and Contributions

In Chapter 3, a camera-based 3D positioning system from Motion Analysis is implemented to provide ground truth for experiments in state estimation, control, system identification, and path planning. An architecture is described that reads information provided by the positioning system and makes it available for use in control systems. Specifically, an autonomous controller for the Hummingbird and Pelican quadrotors from Ascending Technologies is developed to allow the aircraft to hold position and velocity, and follow predetermined paths provided by Matlab.

While Chapter 3 does not present any new research, it describes the architecture that was developed to allow in-flight experiments to be performed. In the future, it will serve as a starting-point for experiments in indoor navigation, landing-site identification, and cooperative multi-agent state estimation, among other things.

6.2.2 Future Work

The architecture described in Chapter 3 is in a somewhat raw form: there is no graphical interface for interacting with the controller and tuning gains, and the software has to be modified and rebuilt to allow even minor changes to be made. A graphical interface would simplify future development and reduce the learning-curve for new researchers.

6.3 System Identification for Miniature Quadrotor Aircraft

6.3.1 Overview and Contributions

In Chapter 4, system identification techniques are used to develop transfer functions relating commanded angles to actual angles on the Hummingbird and Pelican quadrotors from Ascending Technologies. System identification is a well-developed field, and Chapter 4 does not provide any new research in that area. Rather, the positioning system described in Chapter 3 is used for system id, and the process is documented to facilitate future system identification projects in the MAGICC Lab.

6.3.2 Future Work

The models developed in Chapter 4 assume that motion occurs about only one axis at a time. This simplifies the problem considerably, but if the aircraft is expected to perform aggressive maneuvers involving more than one axis or movement, the developed models may be insufficient. An interesting area for future work may be to use system identification techniques to identify system parameters in a complete 3-axis nonlinear quadrotor model.

6.4 Using Accelerometers for Rotorcraft State Estimation

6.4.1 Overview and Contributions

In Chapter 5, the effect that quadrotor flight dynamics has on accelerometer measurements is described. It was demonstrated that an estimator that ignores these dynamic effects can produce significant angle estimate errors, though these errors may not be observable without ground truth for comparison. An intuitive model describing accelerometer measurements as low-pass filtered, scaled versions of body-frame velocities is presented. The corner frequency of the filter model is set by the mass and drag coefficient of the aircraft.

An attitude estimator is presented that accounts for the aircraft flight dynamics properly and achieves much higher accuracy as a result. Estimators are implemented in simulation and in hardware to demonstrate both the errors that arise through improper accelerometer use, and the accuracy improvement that can be obtained by properly accounting for aircraft flight dynamics.

6.4.2 Future Work

The estimator developed in Chapter 5 assumes that motion is limited to one axis, and that motion about each axis is therefore independent. It would be interesting to develop a non-linear estimator that take into account the full 3D kinematics and dynamics of a flying quadrotor aircraft.

It would also be interesting to develop methods for determining the drag coefficient of the aircraft using low-cost sensors. The drag coefficient was determined by using the motion analysis system developed in Chapter 3, but such systems can be cost-prohibitive. It may be possible to add low cost optic-flow sensors to a quadrotor to measure body-frame velocities. These velocities could then be used in conjunction with measured acceleration to experimentally determine the drag coefficient.

Bibliography

- [1] Z. Tang and U. Ozguner, “Sensor fusion for target track maintenance with multiple UAVs based on Bayesian filtering method and hospitability map,” in *Proc. 42nd IEEE Conference on Decision and Control*, vol. 1, 9–12 Dec. 2003, pp. 19–24. 1, 7
- [2] U. Zengin and A. Dogan, “Real-time target tracking for autonomous UAVs in adversarial environments: A gradient search algorithm,” *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 294–307, April 2007. 1, 7
- [3] S. Martinez and F. Bullo, “On optimal sensor placement and motion coordination for target tracking,” in *Proc. IEEE International Conference on Robotics and Automation ICRA 2005*, 18–22 April 2005, pp. 4544–4549. 1, 7
- [4] S. Kanchanavally, R. Ordonez, and J. Layne, “Mobile target tracking by networked uninhabited autonomous vehicles via hospitability maps,” in *Proc. American Control Conference the 2004*, vol. 6, 30 June–2 July 2004, pp. 5570–5575. 1, 7
- [5] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, Sept. 2006. 1, 7
- [6] J. K. Y. Kim, “Moving target tracking in dense obstacle areas using UAVs,” in *Proc. 17th IFAC World Congress, Seoul, Korea*, 2008. 1, 7
- [7] T. Gao, Z. Gong, J. Luo, W. Ding, and W. Feng, “An attitude determination system for a small unmanned helicopter using low-cost sensors,” in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2006. 4, 44
- [8] R. He, S. Prentice, and N. Roy, “Planning in information space for a quadrotor helicopter in a GPS-denied environment,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008. 4, 44
- [9] G. M. Hoffmann, H. Huang, S. L. Wasl, and E. C. J. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2007. 4, 44
- [10] G. B. Jonathan M. Roberts, Peter I. Corke, “A low-cost and low-weight attitude estimation system for an autonomous helicopter,” in *Proceedings of the Australasian Conference on Robotics and Automation*, 2002. 4, 44
- [11] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a quad-rotor robot,” in *Proceedings of the Australasian Conference on Robotics and Automation*, 2006. 4, 44

- [12] J. Stowers, A. Bainbridge-Smith, M. Hayes, and S. Mills, “Optical flow for heading estimation of a quadrotor helicopter,” *International Journal of Micro Air Vehicles*, vol. 1, 2009. 4, 44
- [13] P. Martin and E. Salaun, “The true role of accelerometer feedback in quadrotor control,” in *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, 2010, pp. 1623–1629. 4, 5, 44, 45, 53, 59
- [14] T. W. McLain and R. W. Beard, “Trajectory planning for coordinated rendezvous of unmanned air vehicles,” in *Proc. GNC 2000*, 2000, pp. 1247–1254. 15
- [15] L. Lennart Ljung, *System Identification: Theory for the User (Second Edition)*. Prentice Hall PTR, 1999. 33
- [16] J.-N. Juang, *Applied System Identification*. Prentice Hall, 1993. 33
- [17] R. Isermann and M. Munchhof, *Identification of Dynamic Systems: An Introduction with Applications (Advanced Textbooks in Control and Signal Processing)*. Springer, 2010. 33
- [18] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, “A complementary filter for attitude estimation of a fixed-wing uav,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2008*, 2008, pp. 340–345. 43
- [19] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, “An extended kalman filter for quaternion-based orientation estimation using MARG sensors,” in *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 4, 2001, pp. 2003–2011. 43
- [20] D. F. Sebastian Thrun, Wolfram Burgard, *Probabilistic Robotics*. The MIT Press, 2005. 43

Appendix A

Flying Rotorcraft Using Cortex for State Estimation

A.1 Introduction

This appendix provides a detailed, step-by-step overview of everything needed to do computer-controlled quadrotor flight using the system described in Chapter 3. In order to run a successful and safe flight, each step should be followed carefully to avoid destructive and potentially dangerous crashes.

At the time of writing this appendix, other researchers in the MAGICC lab have developed different software for interfacing with the Hexacopter aircraft from Mikrocopter. The steps described in this appendix are applicable only to the Hummingbird and Pelican quadrotors from Ascending Technologies. In addition, some of the steps described here will become obsolete in the event that a graphical interface is added to simplify the procedures required for running flight tests. In particular, the steps detailing PC code modifications to interface with the aircraft will not be necessary.

All PC-based code described in this appendix is available on the MAGICC Lab SVN server, and as of February, 2011, it was accessible at the SVN address

```
svn+ssh://magiccvcs.et.byu.edu/svn/indoor_nav/CortexQuadLink.
```

A.2 Architecture Overview

A block diagram showing the overall architecture used in the flight control system is given in Figure A.1. The main components of the architecture are the aircraft itself, hardware

for communicating with the aircraft wirelessly, control software, the motion analysis system, and finally the Matlab-based path planner.

The control software operates onboard a PC ground-station. It interfaces with the motion analysis system, which provides states (position and orientation) at up to 200 Hz over a TCP/IP connection. The control software can also optionally interface with external software to accept high-level position and velocity commands over a TCP/IP connection. This allows the software to receive commands from a path planner. Technically, the control software can receive commands from any program that can establish a TCP/IP connection, but in this system, Matlab was used for path planning. The control software link to Matlab operates both ways, so that state information from the motion analysis system can be transmitted to the path planner.

A communication link with the quadrotor is made using 2.4 Ghz XBee radios from Digi. The radios are designed to operate as a transparent serial “pipe”, with serial appearing on the TX pin on one radio also appearing on the RX pin on the other. The wireless transmission itself is transparent to the user. While the radios work well, their bandwidth is limited, so that in practice it is difficult to both transmit and receive data at high rates simultaneously. When using the XBee radios for autopilot telemetry logging, control commands from the ground station cannot be transmitted to the aircraft at high enough rates to maintain consistent control - the channel becomes clogged and long delays between control commands cause the aircraft to revert to manual control. The implication is that if autopilot telemetry needs to be logged while the aircraft is under computer control, the data must be logged onboard the aircraft itself. There is no native support for doing this, and no hardware was developed to that end. If it is needed, it will need to be developed.

A.3 Pre-Flight Checklist

A variety of things need to happen before the aircraft can be flown under computer control, including prepping the aircraft for flight, enabling the Motion Analysis system, start-

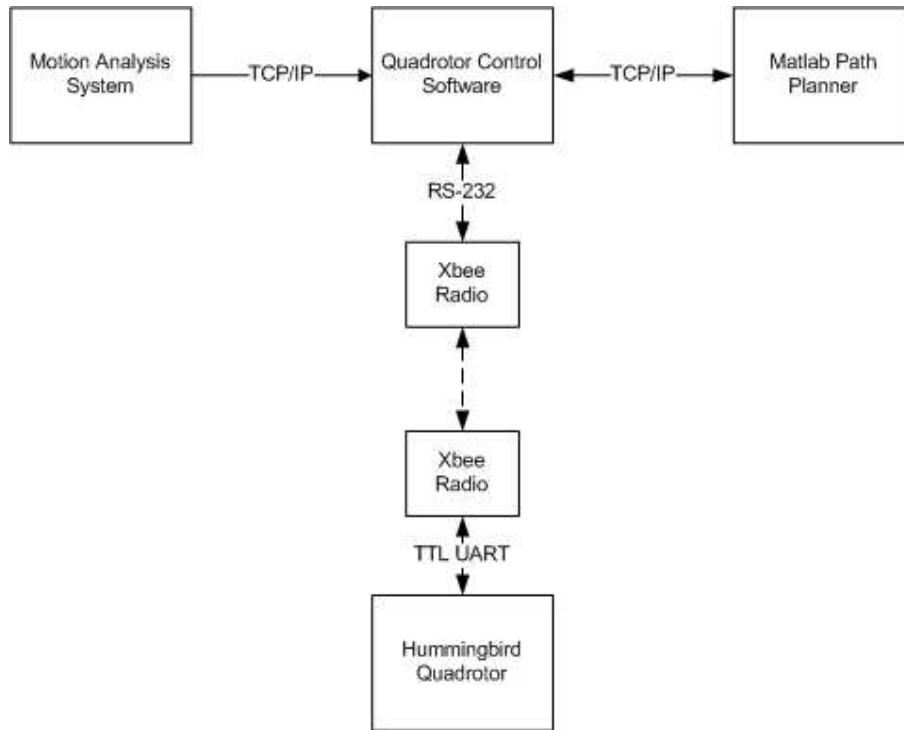


Figure A.1: Flowchart for experimental setup.

ing the control software, and (if applicable) starting the path planning software. Prepping the aircraft for flight involves following the steps below:

1. Inspect the quadrotor airframe, propellers, and autopilot to ensure that there is no obvious damage.
2. Obtain a battery and ensure that it is fully charged.
3. Mount the battery securely in the quadrotor and airframe and plug it in.
4. Turn on the RC transmitter.
5. Place the quadrotor level on the ground and push the power button. After the button is pressed, the motors should beep once, after which the rate gyros will be zeroed. Do not move the aircraft until the motors beep again to indicate that gyro calibration is complete.

If needed, it can be verified that the aircraft is ready for flight by turning the motors on temporarily. The motors are turned on by lowering the throttle all way and pushing the yaw input entirely to the left (moving the left transmitter control stick all the way down and to the left). If the motors turn on, the aircraft is ready to fly.

With the quadrotor ready to fly, the Motion Analysis system should be turned on and configured to track the aircraft and transmit states over the TCP/IP line. The individual steps required to do so are detailed below.

1. Turn on the camera power supply and wait for the camera numbers to appear on each camera's LCD display.
2. Open Cortex and click the "Connect to Cameras" radio button. After a delay, a dialog should appear stating that 8 Hawk cameras were found.
3. Add the correct prop to the system so that Cortex will track its position and orientation. The Hummingbird prop used in this thesis is called Prop_2.3.10.prop. The Pelican prop used in this theis is named Pelican_9.10.prop. You may have to run a search on the Cortex machine to find these files, and then copy them into the Cortex software working directly. Alternatively, new props could be made.
4. Make sure that the number of lines required for the cameras to detect a marker is set to two (See Figure A.2). This ensures that the system will be able to track the aircraft everywhere inside the volume.
5. Make sure that the software SDK is enabled (see Figure A.3). This ensures that position and orientation data is transmitted over the TCP/IP block.
6. Click "Run" and verify that the aircraft is being tracked by the system (see Figure A.4). As shown, if the aircraft is being tracked, lines will be drawn between the markers used to define the prop.

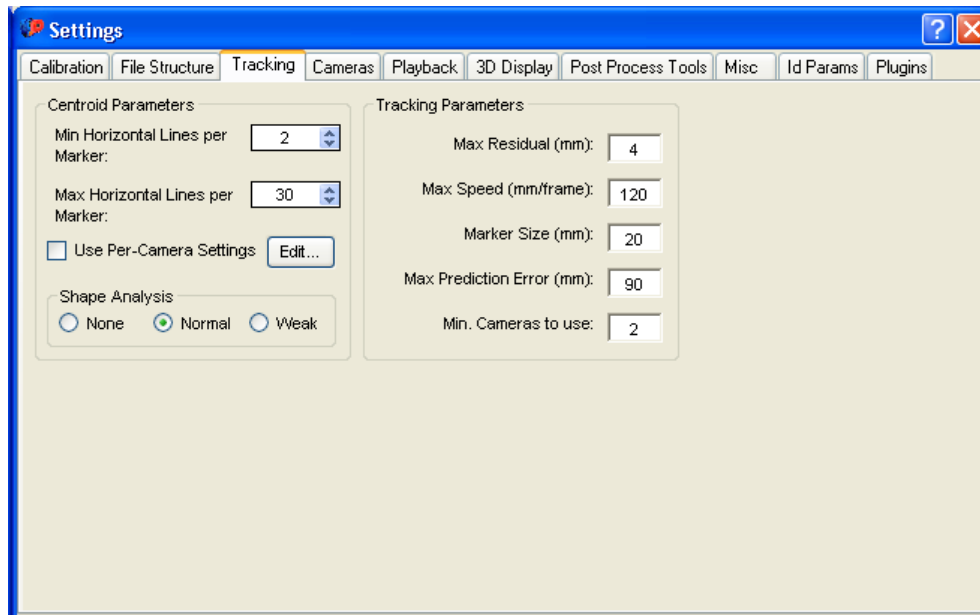


Figure A.2: Dialog for setting the minimum number of lines (pixels) required for the Motion Analysis system to detect and track a marker. For tracking the Hummingbird and Pelican props, this should be set to two.

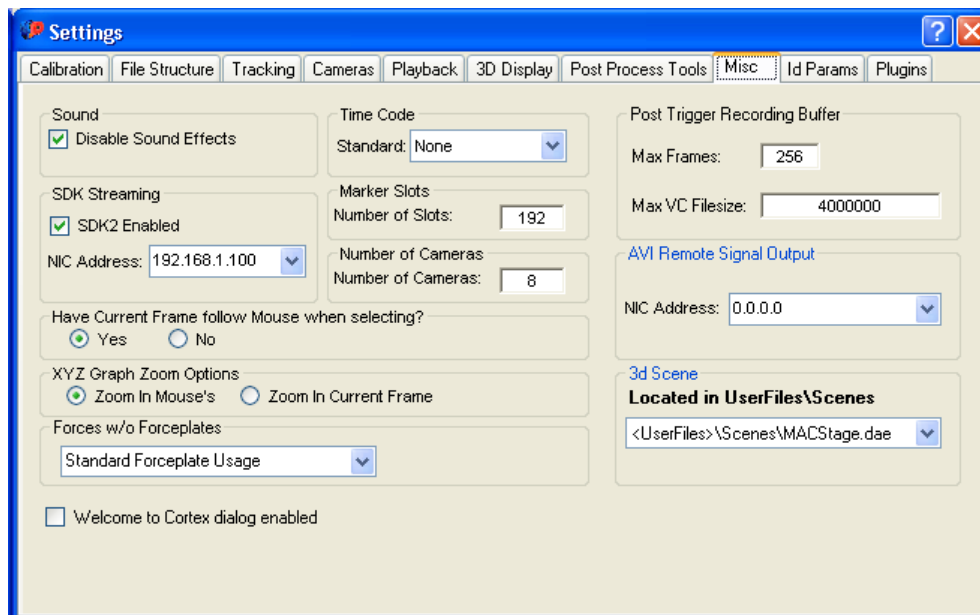


Figure A.3: Dialog for enabling the SDK for broadcasting states computed by Cortex over the TCP/IP connection.

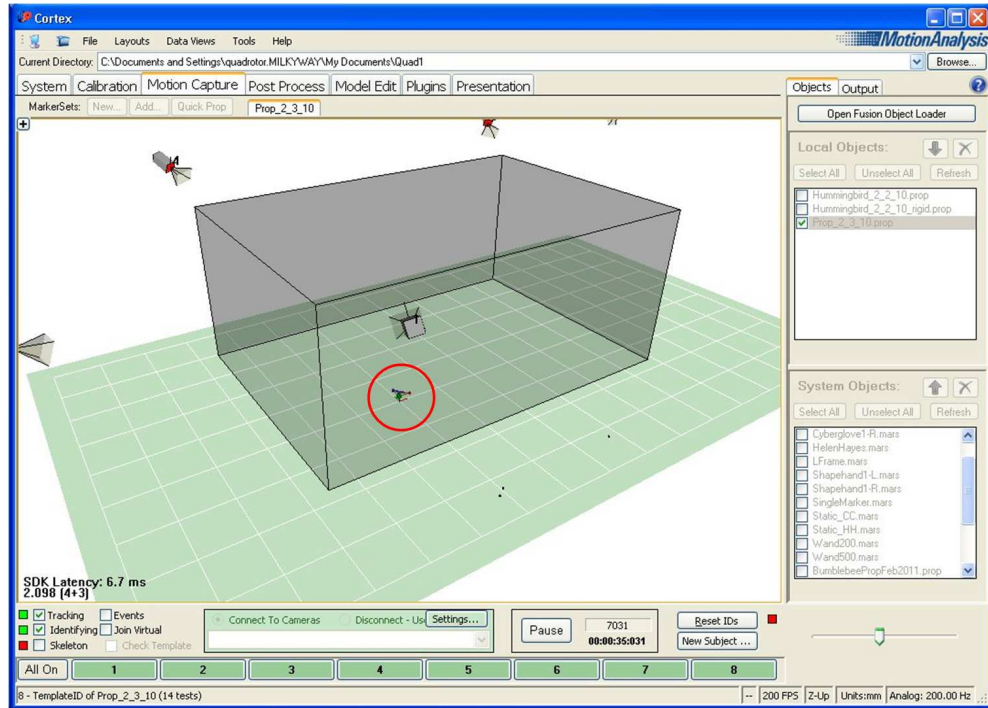
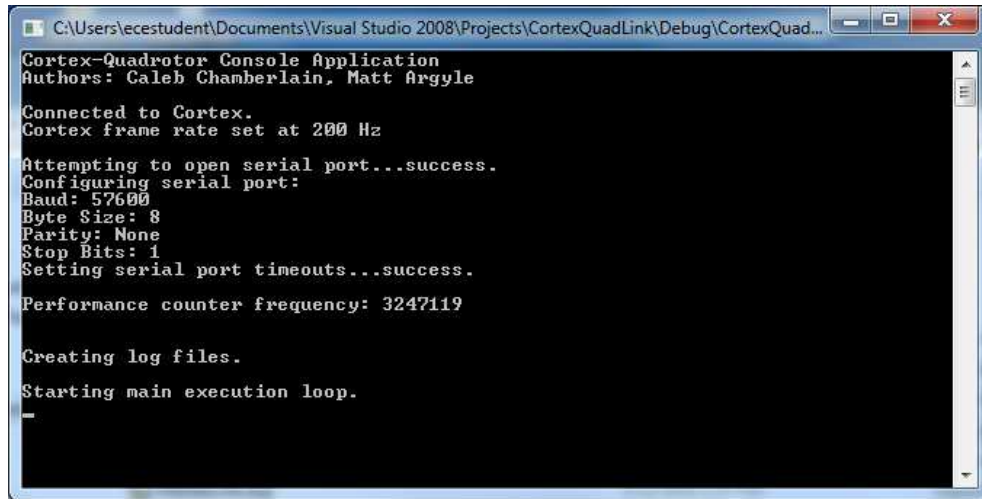


Figure A.4: Main Cortex display window when a prop is being tracked. Lines should be drawn between the markers in the prop as shown in the area highlighted in red.

Once Cortex is running and tracking the aircraft consistently, and once the aircraft is ready for flight, the control software should be run. The steps for doing so are enumerated below:

1. Make sure that an XBee radio is connected to the ground station computer, and that a matching XBee radio is connected to the aircraft. The radios will need to be configured using the XCTU configuration software from Digi. The serial baud rate of the aircraft's radio should be set to 57600. The serial baud rate of the ground station's radio can be set to anything, but the software will need to be rebuilt to connect at the correct baud rate.
2. Make sure that the CortexQuadLink control software is configured to control the correct aircraft (more details on this later).



```
C:\Users\ecestudent\Documents\Visual Studio 2008\Projects\CortexQuadLink\Debug\CortexQuad...
Cortex-Quadrotor Console Application
Authors: Caleb Chamberlain, Matt Argyle

Connected to Cortex.
Cortex frame rate set at 200 Hz

Attempting to open serial port...success.
Configuring serial port:
Baud: 57600
Byte Size: 8
Parity: None
Stop Bits: 1
Setting serial port timeouts...success.

Performance counter frequency: 3247119

Creating log files.
Starting main execution loop.
-
```

Figure A.5: Output of CortexQuadLink software when it has successfully connected to Cortex and is running control loops. If the XBee modems are properly connected and the aircraft is turned on, then the aircraft should be receiving control commands.

3. Run the CortexQuadLink interface software. It should connect to Cortex and start computing control commands and communicating with the aircraft (see Figure A.5).
4. Check to make sure that the aircraft radio is receiving consistent control commands - the RX light on the radio board should be consistently lit. If it isn't, COM is bad. Do not fly the aircraft under these conditions.
5. If COM is good, and if an external path planner is not being used, then the aircraft is now ready to fly. Turn on the motors and flip the manual/computer control switch on the transmitter to switch to computer control.

If an external path planner is being used (i.e. in Matlab), then the path planning software should be run before trying to switch to computer control. When the CortexQuadLink software is run, it will wait for a client connection before continuing to run control loops (See Figure A.6).

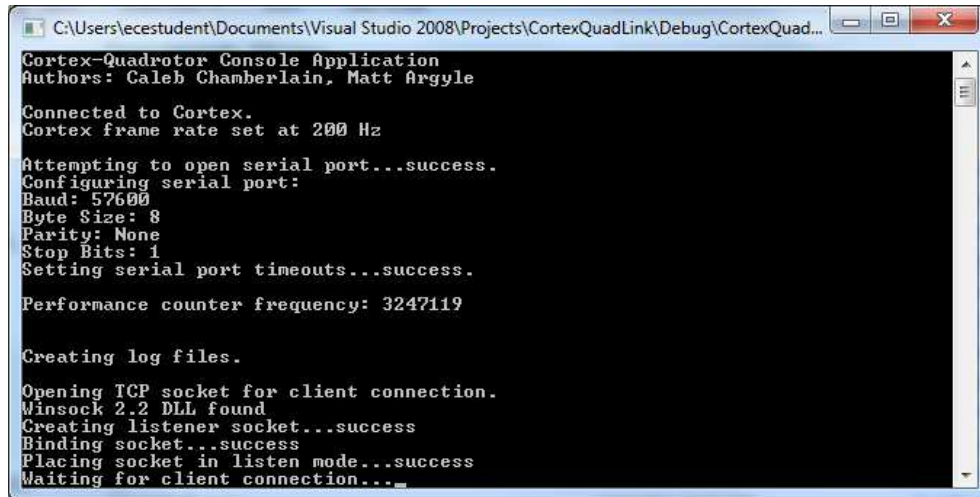


Figure A.6: Main Cortex display window when a prop is being tracked. Lines should be drawn between the markers in the prop as shown in the area highlighted in red.

A.4 Configuring CortexQuadLink Control Software

The control software developed in this thesis, called CortexQuadLink, does not include a user-interface of any sort to change settings, which means that to change anything, the code has to be opened, modified, and recompiled. In this section, an overview of all the necessary software changes is given for different flight modes. Listing the exact code is beyond the scope of this Appendix, but general directions and the names of the files to be modified are given.

The control software supports a variety of different options that influence how the software behaves. These options should be set before attempting to fly the aircraft. The list given below summarizes the different options that must be set.

- Specify whether the Pelican or the Hummingbird is being used (the Pelican used in this thesis uses the same autopilot as the Hummingbird, so changing aircraft only involves setting the correct control gains).
- Specify whether an external path planner is being used (if so, the software will try to connect before running the control algorithms).

- Specify the control mode. The control software can run position hold, velocity hold, and course/heading control loops.
- Set up specific control modes for the aircraft itself. Any combination of control channels (pitch, roll, thrust, yaw rate, etc.) can be controlled by the computer while the remaining channels remain under manual RC control.

To specify which aircraft is being used, only the control gains need to be modified. This can be done in the main file, `CortexQuadLink.cpp`. A section in that file sets control gains for either the Hummingbird or the Pelican. The control gains that aren't being used are simply commented out. To change the aircraft being used, just uncomment the relevant section and comment out the code for the other aircraft.

If an external path planner is being used, then the `CortexQuadLink` software needs to be set up to connect to a client over the TCP/IP connection, and to send and receive packets from that client. The code that does this is also in the `CortexQuadLink.cpp` file. One section of code appears just before the main program loop (this section handles the initial connection), and another section is within the main loop itself (this section sends and receives TCP/IP packets).

The controller itself can operate in position-hold, velocity-hold, or course/heading hold modes. Depending on the types of commands that will be sent to the aircraft, the proper control mode must be set. This can be changed in the file `QuadControl.h` by setting a preprocessor definition. There are three definitions already written in the file, with only one uncommented based on the desired mode. The code

```
#define COURSE_HEADING_CONTROL
```

enables course and heading control, while the code

```
#define VELOCITY_CONTROL
```

enables velocity control, and

```
#define POSITION_CONTROL
```

enables position control.

Finally, any subset of channels can be controlled by the computer while the remaining channels remain under manual control. When sending control commands to the aircraft, a portion of the control packet identifies which channels should remain under human control, and which channels will be controlled using data in the control packet. The CortexQuadLink software handles the low-level details of the control packet construction. To alter which channels are controlled by the computer, a structure called “hummingbirdConfig”, located in CortexQuadLink.cpp, should be modified. The structure is already filled in the aforementioned file, and the naming conventions are intuitive, so setting it up is a simple matter.

Once all the settings have been configured as desired, the code should be compiled and built. The system should then be ready to control the aircraft autonomously.

A.5 Conclusions and Future Work

Learning how to use the flight system is a non-trivial endeavour, but it becomes easy after the first few flight experiments. This Appendix provides information to help new users get started. In the future, a better system would include a graphical interface to allow users to run experiments without re-writing code and rebuilding every time gains need to be changed or configuration options adjusted. Even with a new interface (and, for that matter, even for different aircraft), the pre-flight checklists for flying the aircraft and for setting up the Motion Analysis system should remain fairly consistent.