Failure-Tolerant Control and Vision-Based Navigation for Hexacopters

（ヘクサコプターのための耐故障制御と視覚に基づくナビゲーション）

ラビ　クリストファー　トマス
Christopher Thomas Raabe

Failure-Tolerant Control and Vision-Based Navigation for Hexacopters

（ヘクサコプターのための耐故障制御と視覚に基づくナビゲーション）

by

Christopher Thomas Raabe

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

University of Tokyo

Examination Committee:	
Chair:	Professor Shinji Suzuki
Member:	Professor Koichi Hori
Member:	Professor Masatoshi Ishikawa
Member:	Professor Takeshi Tsuchiya
Member:	Professor Takehisa Yairi

Abstract

The research summarized in this thesis is motivated by the events at the Fukushima Daiichi nuclear power plant that resulted from the 2011 Tohoku earthquake and tsunami. Those events exposed the need for an autonomous vehicle that can be deployed remotely to closely inspect a facility that is no longer accessible by personnel. Two key elements of this mission have been identified as requiring further research. First is the need to improve safety to avoid risk of damage or injury due to a crash while in transit to the remote location. Second is the need for a sensor that can provide position measurements with much greater precision than GPS or in environments where GPS is not available.

0.1 Outline

Chapter 1 introduces the hexacopter rotorcraft configuration and compares it to other types of hovering aircraft. Previous research on the subjects of fault-tolerant control of multicopters and vision-based navigation Is reviewed.

Chapter 2 introduces the equations of motion that describe hexacopter flight. The various forces and moments that are applied to the hexacopter are cataloged and modeled. A linear model is developed for use in designing a simple Linear Quadratic Gaussian (LQG) controller.

Chapter 3 proposes an adaptive controller that can simultaneously estimate and adapt to changes in actuator effectiveness with guarantees on speed of estimation. An error projection operator is proposed for projecting 4 error measurements to the 6 motors/propellers of the hexacopter. A method for smoothly reconfiguring upon detection

of total loss of effectiveness is discussed.

Chapter 4 introduces the PTAM algorithm, which is the basis for the proposed vision-based navigation. The merits and limitations of PTAM for vision-based navigation are discussed. Several improvements are proposed to enhance the utility of PTAM for robotic navigation. Extensions for control and communication with an embedded attitude controller are presented.

Chapter 5 introduces the flying testbed that was used for control and navigation experiments. It is composed of a Mikropkopter Hexa XL hexacopter paired with a custom computer-vision package based on a Mac mini computer. The experimental embedded software that was developed to facilitate this research is discussed in detail.

Chapter 6 presents data collected from simulated and real flight tests to evaluate the performance of the methods and systems described in the preceding chapters. As a first step, a baseline controller is evaluated. Next, test results that demonstrate the precision of vision-based position estimation are presented. Then, data recorded from a successful indoor autonomous navigation task is shown. Finally, a series of simulated trials demonstrate the ability of the failure-tolerant controller to adapt to a total propeller failure and compares its performance to the baseline controller in normal operation or when subject to disturbances.

Chapter 7 concludes the thesis with a discussion of the proposed failure-tolerant control and high-speed vision-based navigation system. The merits and deficiency of each are summarized and suggestions for future work are listed.

Failure Tolerability of a Hexacopter and its Relatives

Some common configurations for hovering rotorcraft are: traditional helicopters, tricopters, quadrotors, hexacopters, and octocopters. Of these, only the hexacopter and octocopter configurations remain controllable after any single actuator failure. For hexacopters, the propeller opposite the one that has failed must be operated near zero speed

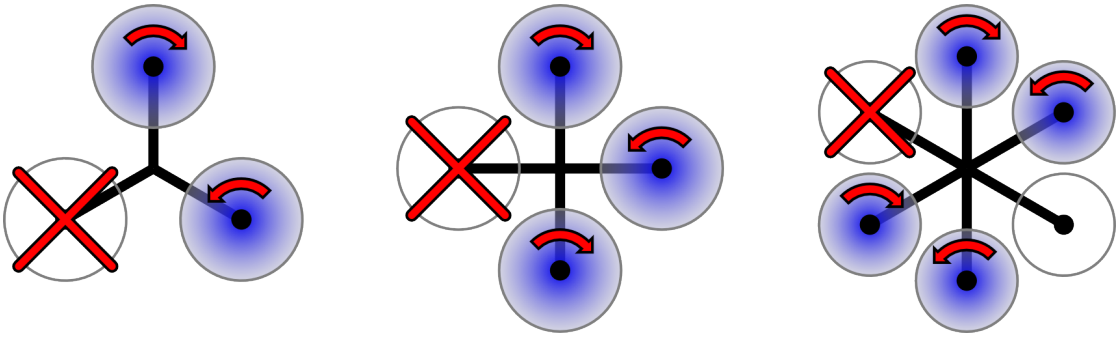


Figure 1: Single-propeller failure for some common multirotor configurations.

in order to maintain control, so that the weight of the vehicle is carried primarily on four of the six propellers. A few select configurations are shown in Figure 1.

0.2 Control of a Hexacopter

For a traditional helicopter, control of the roll, pitch, yaw, and vertical accelerations is decoupled through clever mechanical design. In the case of quadrotors, hexacopters and octocopters, decoupling is typically accomplished through the use of a signal processor that actively converts decoupled body-axis moment and thrust commands into individual propeller speed commands by inverting the system dynamics. For a quadrotor, this inversion is straightforward since the 4 propellers each orthogonally affect the 4 degrees of control freedom, so the actuation matrix is square and invertible. A hexacopter however, is an over-actuated system (it also has only 4 degrees of control freedom, but has 6 actuators), so the actuation matrix is not square and cannot be inverted. Rather, there exist an infinite number of pseudo-inversions. Taking the Moore-Penrose pseudo-inverse results in propeller speed commands that are symmetrically distributed.

With the dynamics decoupled, feedback control can be applied separately to each degree of control freedom. In the case of attitude control, typically only angular rates are measurable from the IMU, but the other states are observable, so an observer (e.g. a Kalman filter) may be used to estimate the other states in order to arbitrarily place

the poles of the closed-loop dynamics with state feedback.

0.3 Adaptive Control

Adaptation to Control Effectiveness

For a critically actuated system such as a quadrotor (where the number of actuators matches the number of degrees of control freedom), an adaptive law is proposed that simultaneously estimates and adapts to changes in actuator effectiveness. This is achieved by comparing the measured state rates to those predicted by a model of the system. The error between the two is projected back to the actuators by inverting the actuation matrix. A combination of the projected error and the actuator command is then used to drive the effectiveness estimate for each actuator. The model actuation matrix combined with the estimated effectiveness is then inverted to compute a new control signal. The control inversion and adaptive law are structured such that, under nominal conditions (i.e. the system is well modeled, the real actuator effectiveness is not changing, and the propeller speed commands are away from zero), the effectiveness estimates will approach their true values exponentially with a time constant that is the reciprocal of the adaptive gain.

Sensitivity

The adaptive law described in the above section is analyzed for sensitivity to disturbances, measurement noise, and modeling uncertainty. It is found that a fundamental trade-off exists between the speed of convergence and the sensitivity to disturbances, noise, and uncertainty in the state function, especially for small actuator commands. A sufficient condition that guarantees stability in the presence of modeling uncertainty remains an open area of investigation.

Extension to the hexacopter

In the case of a hexacopter, even with perfect measurement of the state and state rates and perfect knowledge of the propeller speeds, it is not possible to determine the effectiveness of 6 actuators from only 4 measurements. Rather, there exists an infinite possibility of combinations of actuator effectiveness that could result in the measured error. Some extra information is required to find a unique solution to this system, which has six unknowns with only four relationships.

This thesis proposes applying likely constraints in order to make the problem solvable. For example, it could be assumed that 5 of the 6 actuators are performing close to design effectiveness and only 1 actuator is deviant. This constraint effectively reduces the number of unknowns from 6 to 1. When examining each possible deviant propeller, there are 4 equations and only 1 unknown. Each set of 4 equations can be solved for each propeller, and the propeller with the least variance in its solutions is guaranteed to be the propeller with the most deviant effectiveness. However, no information is learned about the effectiveness of the other propellers.

Another possible constraint could be to assume that 3 propellers have some nominal deviation from design effectiveness and the remaining 3 have unique deviations from design effectiveness. This constraint reduces the number of unknowns from 6 to 4. While it has the potential to give much higher fidelity in the effectiveness estimates for each of propeller, it turns out that many of the possible scenarios are not testable due to symmetry in the actuation matrix which leads to singularities in many cases.

Several other possible constraints lie in between the two mentioned above. It is determined that the only practical constraint is to assume that 5 of the propellers have some nominal deviation from design effectiveness and 1 propeller is uniquely degraded. This constraint allows for adaptation to uniform differences in actual performance and allows detection and adaptation to partial or total degradation of a single propeller.

Replacing the projection of measured error through inversion of the actuation matrix

with the projection operator described above effectively extends the adaptive law to the case of the hexacopter.

0.4 Vision-Based Navigation

PTAM

Parallel Tracking and Mapping (PTAM) is an algorithm developed by Klein and Murray at the University of Oxford for the purpose of simultaneous localization and mapping (SLAM) for augmented reality. As its name implies, this algorithm divides the localization task and the map building task into separate parallel threads to improve performance. "Map building" refers to the process of storing the location of observed features for future comparison, even after those features have exited the field of view. "Localization" refers to the process of matching currently observed features to those stored in the map to attempt to determine the current position and rotation of the camera relative to the origin of the map.

Modifications

The PTAM algorithm was originally designed for use with augmented reality, where the position of a camera is tracked so that artificial objects can be superimposed onto the scene of the real world. Several modifications have been made to adapt the algorithm to the task of autonomous navigation.

PTAM arbitrarily sets the orientation and scale of the map upon initialization. Some extra information is required to align PTAMs map with world coordinates. The ability to recognize a fiducial marker of known size was added so that origin and scale of the map could be aligned to the marker.

In the case of augmented reality, it is not expected that the camera will move large distances, but that is not true for navigation. An expanding map consumes increasing

computation time and memory and will saturate at some point. To alleviate this risk, additions to the database were made more judicious.

For augmented reality, much of the computation is involved in rendering the scene. For navigation, rendering is only necessary for visual confirmation of proper tracking and therefore can be made much lower priority. To increase performance, rendering and some other functions, were split into separate threads of execution so that tracking could be performed as quickly as possible.

0.5 Experimental Results

Failure-Tolerant Control

Unfortunately, it was not possible to perform flight tests with the adaptive controller that is proposed in this thesis. However, its performance was tested extensively in simulation. Results are presented that show that this method can quickly recover in the case of a single propeller failure. It can also accurately estimate the changing effectiveness of several propellers. However, it is also shown that typical performance is degraded compared to the baseline, non-adaptive, controller. Furthermore, strong disturbances may cause instability. The hexacopter is particularly sensitive to yaw disturbances with this control scheme.

Vision-based navigation

A pair of experiments compares the precision of position estimates from PTAM to those from GPS and a high-precision motion capture system. In these experiments a camera recorded downward-looking video aboard vehicles whose positions were tracked either by GPS or a motion capture system. The recoded video was post-processed with PTAM and the resulting trajectory was compared. It was found that the system is capable of accuracy on the order of a few centimeters.

Another experiment investigated the ability to perform indoor autonomous navigation using the proposed vision system. In this experiment, the hexacopter autonomously repeated a circuit of takeoffs and landings at 3 waypoints. The circuit was repeated 5 times. It was found that the navigation was drift free and landing was accurate to within 20 cm.

Acknowledgments

The research presented in this thesis was financially supported through a monthly stipend from the 21 st Century (Global) COE Program Mechanical Systems Innovation from October 2010 through March 2013, a generous contribution from the Ishimori Memorial Scholarship Foundation from March 2012 through September 2013, and the SEUT Fellowship B of the University of Tokyo Graduate School of Engineering Doctoral Student Special Incentive Program from March 2013 through September 2013.

I am extremely grateful to Professor Shinji Suzuki for giving me the opportunity to do research at the University of Tokyo and for all of his support and guidance. His kindness, humility, and knowledge have been inspirational to me.

I would like to thank Professor Masatoshi Ishikawa and Dr. Carson Reynolds for allowing me to collaborate on their research. I'd especially like to thank Dr. Reynolds and Daniel Henell of the Ishikawa-Oku lab who contributed greatly to the computer vision portion of this research. I was constantly humbled by their knowledge and skill.

Professor Takeshi Tsuchiya, Professor Koichi Hori, Professor Takehisa Yairi, and Professor Masatoshi Ishikawa are acknowledged for taking the time to review this thesis and for their comments and discussion.

Thanks to Dr. John Vian and Dr. Emad Saad of Boeing Research and Technology for their support, inspiration, and contributions to this research.

My colleagues, especially Dr. Takuma Hino, Dr. Adriana Andreeva-Mori, Dr. Miles Coleman, Dr. Jorg Entzinger, and Atsushi Matsushita, were excellent friends and were

incredible supportive, both technically and otherwise.

Many thank Kenji Karasawa for his exceptional support as my laboratory's administrator. He helped me with many official and practical details (including helping me move apartments) and always made sure that our lab's drinking water was free of lead.

Ms. Yukimi Umeda, Ms. Yoshie Minegishi, and Ms. Kaori Sato of the MEM international graduate program have been extremely helpful to me and many other international students. They eased the process of enrolling in the university, helped us apply for housing and funding, cared for us after the 2011 earthquake, and arranged some very fun and memorable international student study tours.

I would like to thank my employer, the Boeing Company, and particularly my supervisor Todd Peach. They have been incredibly flexible in allowing me to go on leave of absence for such a long time, and to return for short summer stints to refresh my coffers.

A big thanks to the staff and all of the friends that I made at the Tokyo International Exchange Center. They believe in the mission of the center and have worked hard to ensure its continuation.

Lastly and most of all, I'd like to thank my exceptional parents, Tom and Renee. Their support has been unwavering. Without their love and reassurance, I never would have made it.

Contents

List of Tables

List of Figures

Acronyms

ADC analog-to-digital converter.

BLDC brushless DC electric motor.

BSC backstepping control.

CG center of gravity.

CLF control Lyapunov function.

CPU central processing unit.

CRC circular redundancy check.

FAST features from accelerated segment test.

GPS the global positioning system.

IMU inertial measuring unit.

LED light emitting diode.

LiPO lithium ion polymer.

LQE linear quadratic estimator.

LQG linear quadratic Gaussian.

LQR linear quadratic regulator.

MoCap motion capture.

MPC model predictive control.

MRAC model reference adaptive control.

PPM pulse-poisition modulation.

PTAM parallel tracking and mapping.

PWM pulse-width modulation.

R/C radio control.

RANSAC random sample consensus.

SLAM simultaneous localization and mapping.

SMC sliding mode control.

SSD solid state drive.

TWI two wire interface.

UART universal asynchronous serial receiver and transmitter.

USB universal serial bus.

VO visual odometry.

VSTL Vehicle Swarm Technology Lab.

Nomenclature

A	state matrix
A_x	reference area for computing drag along the x body-axis [m^2]
A_y	reference area for computing drag along the y body-axis [m^2]
A_z	reference area for computing drag along the z body-axis [m^2]
B	actuation matrix
C_x	coefficient of drag along the x body-axis
C_y	coefficient of drag along the y body-axis
C_z	coefficient of drag along the z body-axis
$\mathbf{d}$	disturbance vector
D_n	drag from the n^{th} propeller (opposite direction of horizontal motion) [N]
d_n	direction of rotation of the n^{th} propeller (1: right-hand about z body-axis, -1: left-hand about z body-axis)
$\mathbf{e}$	state error vector
$\mathbf{F}$	body-frame force vector acting at the center of mass [N]
$\mathbf{F}_{ab}$	body-frame force vector due to body aerodynamics [N]

$\mathbf{F}_{ap}$ body-frame force vector due to propeller aerodynamics [N]

$\mathbf{F}_{g}$ body-frame force vector due to gravity [N]

$f(\mathbf{x})$ state function

$g(\mathbf{r}, \epsilon)$ a function that inverts $\mathbf{r}$ with divide-by-zero protection in the region ϵ around zero

$h(\mathbf{e})$ state error projection operator

I moment of inertia matrix about the center of mass [kg·m^2]

I_p propeller moment of inertia matrix [kg·m^2]

$\mathbf{K}$ state feedback gain vector

k_Q constants of proportionality for thrust from a change in rotational speed of a propeller [N/(rad/s)]

k_T constants of proportionality for thrust from a change in rotational speed of a propeller [N/(rad/s)]

l_n lever-arm from the cg to the n^{th} propeller hub [m]

$\mathbf{M}$ body-frame moment (torque) vector acting at the center of mass [N·m]

m mass [kg]

$\mathbf{M}_{ab}$ body-frame moment vector due to body aerodynamics [N·m]

$\mathbf{M}_{ap}$ body-frame moment vector due to propeller aerodynamics [N·m]

$\mathbf{M}_{gp}$ body-frame moment vector due to the reaction from the gyroscopic effects of the spinning propellers [N·m]

$\mathbf{n}$ measurement noise vector

p angular velocity along the x body-axis [rad/s]

P_n rolling moment from the n^{th} propeller (about the direction of horizontal motion) [N]

q angular velocity along the y body-axis [rad/s]

Q_n torque from the n^{th} propeller (about the shaft) [N]

R earth-body rotation matrix

r angular velocity along the z body-axis [rad/s]

$\mathbf{r}$ reference actuator command vector

t time

T_n thrust from the n^{th} propeller [N]

u velocity of the cg along the x body-axis [m/s]

$\mathbf{u}$ actuator command vector

u_n speed command for the n^{th} propeller

u_n velocity of the n^{th} propeller hub along the x body-axis [m/s]

v velocity of the cg along the y body-axis [m/s]

$\mathbf{V}$ body-frame velocity vector at the center of mass [m/s]

v_n velocity of the n^{th} propeller hub along the y body-axis [m/s]

w velocity of the cg along the z body-axis [m/s]

w_n velocity of the n^{th} propeller hub along the z body-axis [m/s]

x position along the x earth-axis (typically north) [m]

$\mathbf{x}$	states vector
y	position along the y earth-axis (typically east) [m]
z	position along the z earth-axis (down) [m]
c	grouping of state rates to be directly controlled
d	a desired value
m	a modeled value
u	grouping of state rates to be uncontrolled
$\hat{}$	an estimated value
$\tilde{}$	a measured value
γ	adaptive gain (constant)
θ	pitch Euler angle [rad]
Λ	diagonal matrix of propeller effectiveness values
ρ	density of the air [kg/m^3]
τ	time constant for motor/propeller dynamics
ϕ	roll Euler angle [rad]
ψ	heading Euler angle [rad]
$\mathbf{\Omega}$	body-frame angular velocity vector [rad/s]
ω_n	rotational velocity of the n^{th} propeller [rad/s]

Chapter 1

Introduction

1.1 Motivating Mission

The research summarized in this thesis is motivated by the events of the Fukushima Daiichi nuclear disaster: a series of equipment failures, nuclear meltdowns and releases of radioactive materials at the Fukushima I Nuclear Power Plant following the Tohoku earthquake and tsunami on 11 March 2011. Those events exposed the need for a robotic vehicle that can be deployed remotely to closely inspect a facility that is no longer accessible by personnel.

Table 1.1 attempts to formalize the mission requirements that drive the selection of the aircraft configuration and focus of this research. The mission is illustrated in figure 1.1.

	Requirements
1	The vehicle should be deployed at least 1 kilometer from the inspection site
2	The vehicle should be able to inspect the entire exterior of the site from a distance as close as 2 meters
3	The vehicle should be able to enter structures at the site if accessible

Table 1.1: Assumed requirements for motivating mission

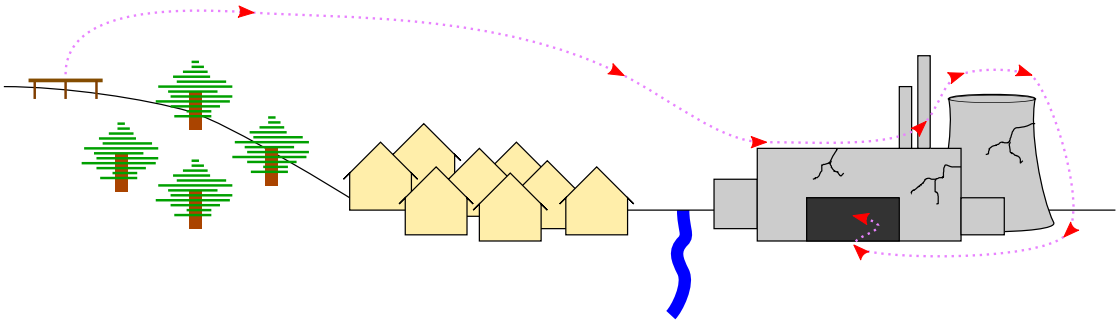


Figure 1.1: Illustration of the motivating mission

A small hovering aerial vehicle is an obvious choice to satisfy such mission requirements. An aerial vehicle can easily traverse the distance from the deployment site to the inspection site in a direct line without concern about obstructions on the ground. Given adequate sensors, a hovering aerial vehicle should be able to freely navigate the accessible 3-dimensional space in and around a structure.

Two key elements of this mission have been identified as requiring further research. First is the need to improve safety to avoid risk of damage or injury due to a crash while in transit to the remote location. Second is the need for a sensor that can provide position measurements with much greater precision than the the global positioning system (GPS) or in environments where GPS is not available.

1.2 Why a Hexacopter

Several aircraft configurations are capable of hover. Some of the more common configurations are rotorcraft, blimps, ducted fans, and flapping wings. Of these, rotorcraft seem to be the best suited for the missions requirements in section 1.1. Table 1.2 summarizes the deficiencies of other hovering configurations when compared to rotorcraft.

Focusing on rotorcraft, they can be further sub-categorized according to the number and orientation of the rotors. Some common rotorcraft configurations are: traditional helicopters (two articulated rotors), tricopters, hexacopters, and octocopters.

Configuration	Deficiency
Blimp	slow; sensitive to wind
Ducted Fan	short endurance
Flapping Wing	short endurance; unsteady

Table 1.2: Deficiencies of other hovering configurations when compared to rotorcraft

1.2.1 Traditional Helicopters

The traditional helicopter configuration (an example shown in figure 1.2) has been the dominant choice for the production of rotorcraft of all sizes until the recent decade or so. The traditional helicopter uses a pair of articulated rotors (typically a main rotor and a smaller anti-torque rotor, or two counter-rotating main rotors) to achieve control. The main benefit of this configuration is that the desired 4 degrees of control freedom are decoupled through clever mechanical design and therefore an active controller is not required. The disadvantage is that the complex mechanics introduce many potential points of failure, the majority of which will lead to an immediate loss of control.[1]

Figure 1.2: Georgia Tech's modified Yamaha RMax helicopter.

1.2.2 Quadrotors and Tricopters

Recently, quadrotors and tricopters (examples shown in figure 1.3) have become a popular configuration for miniature (on the order of 1 meter or smaller) rotorcraft. Both

[1]Helicopters can maintain control in the case of drive failure by autorotation, but this requires immediate landing. Failure of any other actuator or linkage leads to a total loss of control.

typically employ fixed-pitch propellers,[2] so their mechanically complexity is dramatically reduced compared to a traditional helicopter. As the names suggest, quadrotors employ 4 propellers, typically arranged so that the propellers lie at the vertices of an imaginary square. Tricopters employ 3 propellers, with at least one hinged on the arm that it is attached to so that a yawing torque can be produced by vectoring the thrust.

(a) Tricopter (b) Quadrotor

Figure 1.3: The Tricopter Delrin (a) from fpvmanuals.com and the MK-Quadrokopter (b) from HiSystems GmBH.

These configurations were not practical until recently for several reasons. First, control is typically achieved by modulating the speed of each propeller.[2] The torque that is required to accelerate a propeller is proportional to the 5th power of the length of the propeller, as is shown in appendix A. Therefore, larger propellers require too much torque to achieve the bandwidth necessary for control. At present it seems that the upper limit for propeller size is on the order of about 20 centimeters. Second, each of the propellers effect several of the desired degrees of control freedom and therefore require an active controller to translate decoupled motion commands to propeller speed commands. Third electric motors are much better suited to this application than combustion engines and therefore require the vehicle to carry batteries. Only recently have the performance to weight ratio of electronics and batteries become high enough to be carried aboard rotorcraft with small propellers.

[2]The use of variable pitch rotors has been demonstrated as in [30], but had not yet come into popular use.

In terms of failure tolerability, tricopters will experience an immediate loss of control in the event of loss of thrust from any of the propellers (drive failure, loss of propeller, etc.). Quadrotors are subject to an uncontrollable yawing moment upon loss of thrust from any singe propeller.[3] Details of the failure-tolerance analysis can be found in appendix B.

1.2.3 Hexacopters and Octocopters

Hexacopters and octocopters (examples shown in figure 1.4) can be thought of as an extension of quadrotors since the mechanics are essentially the same except that hexacopters and octocopters employ 6 and 8 propellers respectively. The main reason for the extra propellers is to increase payload while maintaining the use of small propellers with small inertia. However, the presence of the extra propellers also has significant consequences for the control of the vehicle.

(a) Hexacopter (b) Octocopter

Figure 1.4: The University of Tokyo's modified MK-Hexa XL (a) and the MK-Okto XL (b), both from HiSystems GmBH.

In the case of traditional helicopters, tricopters and quadrotors, the number of independent means to affect the vehicle's motion (hereto referred to as *actuators*) is equal to the number of degrees of freedom. This type of system is called a *critically-actuated system*. In such a system, total failure of an actuator implies the loss of at least one

[3] Control of a quadrotor with two failed propellers has been demonstrated in [45].

degree of freedom.

In the case of hexacopters and octocopters, the number of actuators is greater than the number of degrees of freedom. This type of system is called an *over-actuated system.* There are two important consequences of this type of system. First, it might be possible to maintain control of all degrees of freedom even upon total failure of an actuator. Second, there are infinite possible combinations of actuator states that will give the same effect to the degrees of freedom, so control design is not as straightforward as in the case of a critically-actuated system.

For a hexacopter that experiences loss of thrust from a propeller, the propeller opposite the one that has failed must be operated near zero speed in order to maintain control so that the weight of the vehicle is carried primarily on four of the six propellers. That propeller may also have to operate in reverse in order to counteract the coupled roll/yaw from the remaining propellers.

The octocopter configuration is the best suited to the loss of thrust from a propeller. In such a case, an octocopter could simply revert to a quadrotor configuration. It is also possible to distribute the thrust to all 7 of the remaining functional propellers so that, unlike the hexacopter configuration, all functional propellers continue to support the weight of the vehicle in hover and it is not necessary for any propeller to operate in reverse.

Details of the failure-tolerability analysis can be found in appendix B.

1.2.4 Rotorcraft Configurations Summary

Table 1.3 summarizes a comparison between several miniature rotorcraft configurations. It is the author's opinion that for miniature rotorcraft (on the order of 1 meter or less), the hexacopter offers the best blend of performance and safety due to its simple mechanics and failure tolerability.

A depiction of the failure tolerance of a few select configurations is given in Figure 1.5.

Configuration	Pros	Cons	Single Rotor Failure
Helicopter	mechanically decoupled control	many points of failure	immediate landing via autorotation in case of drive failure, total loss of control otherwise
Tricopter		requires propeller swivel	total loss of control
Quadrotor	simple mechanics		uncontrollable yawing moment
Hexacopter	simple mechanics	structure for 6 propellers	control maintained; one propeller must operate near zero or in reverse
Octocopter	simple mechanics	structure for 8 propellers	control maintained

Table 1.3: Comparison of rotorcraft configurations

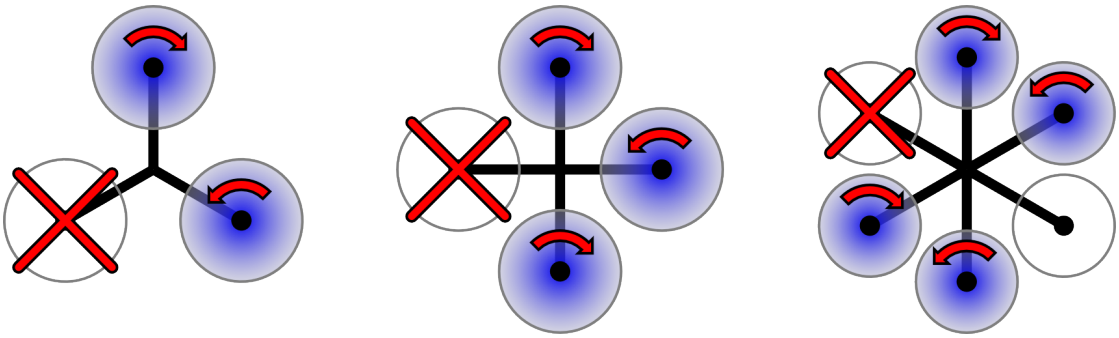


Figure 1.5: Single-propeller failure for some common configurations.

A more detailed analysis of the failure tolerability of various rotorcraft configurations is given in appendix B.

1.3 Failure-Tolerant Control

Avoiding the risk of property damage or bodily harm due to loss of control of an aerial vehicle should be a primary concern for aeronautical engineers. Redundancy is a powerful tool for decreasing the probability of such catastrophic failures. By definition,

a redundant component can only cause a catastrophic failure when at least one other component also fails. Even if the failure of only two redundant components can cause a catastrophic failure, the probability of both of those components failing simultaneously is the product of their individual probabilities of failure (assuming they are uncorrelated). This multiplicative relationship makes catastrophic failure due to redundant components extremely rare. However, the topic of safety and redundancy seems to be mostly ignored for small autonomous vehicles. It is this author's opinion that this subject requires more consideration as small autonomous vehicles become more prolific and government bodies attempt to apply regulations.

The ability to maintain controllability in the event of a fault or failure can be categorized as either active or passive fault-tolerant controls. Passive fault tolerant controls are not adaptive in nature, but rather have accounted for possible failures during the design phase and robustness has been designed in to cover such cases. On the other hand, active fault-tolerant controls are adaptive by nature and therefore change in structure in the presence of a fault or failure.

1.3.1 Literature Review

Research into adaptive controls for aerospace applications has been active for at least 50 years, from MIT's approach to model reference adaptive control (MRAC)[49]. Recently much research has been published about the application of fault-tolerant adaptive control for quadrotors.

A gain-scheduled PID approach was presented in [36]. A fault detection and diagnosis module was used to estimate the health of each propeller. The health variable was then used to smoothly transition between PID gain sets that had been pre-optimized for various fault scenarios. The ability to recover control was proven in flight test by cutting off the blade tips of one of the propellers in flight.

Three different versions of MRAC were compared for fault-tolerant control of a

quadrotor in [7]. The authors referred to these variations on MRAC as the MIT rule MRAC, conventional MRAC, and modified MRAC. In these types of adaptive controllers, the measured performance of the real system is compared to the performance predicted by a model of the system. In direct MRAC, the control parameters are directly adjusted based on the error between the real and modeled performance. In indirect MRAC, the error between the real and modeled performance is used to update the parameters of the model to more closely represent the real system. The model parameters are then used in constructing the control signal. In this case again, all three variations of MRAC were proved to be effective at recovering controllability in flight tests where the blade tips of a single propeller were cut off in flight.

Sliding mode control (SMC) for fault tolerant control of a quadrotor was presented in [28]. SMC is a good candidate for passive fault tolerant control since it is robust to uncertainties and disturbances. This method is a passive fault-tolerant method since the controller is fixed (i.e. it does not adapt), but is robust. To achieve this robustness, performance in fault scenarios are considered during design and an integral sliding mode technique is employed to enhance the robustness and speed of convergence to stable control compared to the case where faults are not considered. This method was demonstrated successful in simulation where the effectiveness of a single propeller was instantaneously reduced.

Backstepping control (BSC) was attempted in [48]. Like SMC, it is a passive approach to fault tolerance. A backstepping controller uses the control Lyapunov function (CLF) to guarantee the global stability. As the name suggests, the controller is designed from "stepping back" from the CLF to the control input. Simulation results showed that this method is robust to small deviations in propeller effectiveness, but perhaps not as suitable as an active fault-tolerant controller.

Model predictive control (MPC) was applied to quadrotors in [21]. MPC employs a reference model and finds the optimal control input for a desired response over a finite

horizon. This optimization is repeated at every time step. However, this method relies on a nearly explicit model of the system, which must also include faults. Therefore a separate health monitor is required to drive the fault parameters. This method was proven to be effective in simulations where effectiveness of one or several propellers were instantaneously reduced.

In all of the above-mentioned works, the faults that are being tolerated are partial loss of effectiveness of one or more propellers. As mentioned in section 1.2.2, quadrotors experience uncontrollable yaw acceleration upon the total loss of thrust from any single propeller and therefore cannot maintain controllability. However, with the hexacopter configuration, control can be maintained after complete failure of any single propeller. To clarify this distinction, the author refers to the controller proposed in this work as failure-tolerant controllers.

1.3.2 Application to Hexacopters

Surprisingly little has been published about control of a hexacopter (although it could be considered a simple extension of quadrotor control), and no publications could be found relating to fault-tolerant control of hexacopters. This is surprising because a hexacopter is capable of continued controlled flight following total failure of a single propeller, while a quadrotor is not.

This thesis explores the mechanics of a hexacopter that has experienced a complete loss of thrust from any single propeller and the changes in control structure that are required to maintain control in such a case. It is assumed that the hexacopter does not have any special sensors for detecting a propeller failure, so a methodology is proposed for inferring propeller effectiveness from the motion of the vehicle as measured by the inertial measuring unit (IMU). Finally, an adaptive controller is proposed to compensate for partial or total loss of effectiveness of any single propeller.

1.4 Computer-Vision Based Navigation

The ability to sense position is critical for autonomous or semi-autonomous missions. The use of the GPS is probably the most common method for determining position in the earth reference frame. However, the spatial resolution of the GPS varies according to Misra et al., from "tens of meters to millimeters" [31] depending upon a variety of factors:

- quality of the GPS receiver
- number of satellites within the receiver's line-of-sight
- fusion with other sensors such as an IMU
- atmospheric distortion
- signal reflections off of structures

One study [44] recorded GPS performance from many locations and estimated error to be 6.9 meters vertically (elevation) and 3.0 meters horizontally (latitude and longitude). Therefore, to achieve higher precision, GPS needs to be either replaced or fused with another sensor.

1.4.1 Literature Review

Recently, much research has focused on the use of digital video cameras as a means to enhance the sensor suite of aerial vehicles. Clever processing of the incoming video stream can provide information about the camera's speed, relative position, or global position depending on the processing method and information available. This processing of incoming digital images is referred to as computer vision. Several approaches have been presented for various types of missions and environments. Approaches include: pattern matching, monocular (single-camera) vision, stereo vision, and some unconventional configurations.

Pattern Matching

Computer vision was applied to the problem of precise autonomous helicopter landing in [37]. The vision system searches for a pattern that the helicopter is to land on. If found, it computes various parameters such as rotation and skew of the pattern compared to the image that corresponds to the landing position. A behavior-based controller is used to compute a trajectory that will move the camera (and vehicle) to the landing position. This method is referred to as visual servoing, where the term servoing implies a direct connection between the inferred vision parameters and the actuator commands. In this case, the rotation and skew of the detected pattern is compared to that of the desired image and the difference is used directly to manipulate the motion of the vehicle.

Another visual servoing technique is presented in [15]. In this case a quadrotor carried a down-ward facing camera and was flown above a target pattern comprised of four black circles. The difference from the current position and the desired position could be determined by the particular arrangement of the four circles that appear in the image. The quadrotor was shown to be able to achieve stable hover above the target using this approach.

A visual-servoing technique similar to the above-mentioned was demonstrated in [33], where color detection of four points, rather than the particular arrangement of circles was used. Again, the vehicle was shown to achieve stable hover.

A unique application of computer vision was presented in [8] for navigating close to and along power lines for inspection. In this case, the computer vision algorithm compared the incoming images of the power lines to an a priori model to determine the precise location of the camera relative to the power lines. This allowed the vehicle to fly along the power lines or maintain a fixed position.

In [38] the regular features of man-made structures were exploited for navigation and tracking. In particular, the windows of a building were detected using pattern matching and tracked over time. The information was used to compute the velocity

of the camera with respect to the windows. The vision system was carried aboard a gas-powered helicopter. When combined with differential GPS, this vehicle was able to perform precise navigation.

A vision-based navigation and control system was developed in [32] for use in a glider. In this work, control was achieved using computer vision only. Inertial sensors such as accelerometers and gyros were not used. Fast pattern matching was employed to detect and compute the relative position of a known pattern on the ground. Combined with an extended Kalman filter for state estimation, control commands for navigation could be computed.

A vision system for autonomous navigation and object location and tracking for Georgia Tech's RMax helicopter was presented in [29]. Motion data from both the inertial measuring unit and the computer-vision system were fused using an extended Kalman filter to produce a full estimate of the vehicle's state.

Monocular Vision

The works mentioned in the previous subsection relied on the use of land marks or artificial features to perform localization. However, it is possible to determine some information about position and/or velocity without any prior knowledge.

Images form a single camera combined with a structure-from-motion algorithm was presented in [22] for landing in an location with unknown terrain. The structure-from-motion algorithm allowed the creation of a topographical map of the area below the helicopter that carried the computer-vision system. Using this map, the helicopter could avoid landing on hazardous areas that might contain rocks, steep slopes, etc., and could choose the most suitable landing location. The helicopter was able to land autonomously using a hierarchical control structure.

A vision-based strategy that does not rely on prior knowledge of landmarks or the use of other artificial visual cues was presented in [6]. This approach used visual odometry

(VO), simultaneous localization and mapping (SLAM), and GPS to precisely determine the location of the vehicle. GPS was used as the base sensor, but VO was used to fill in the gaps when GPS did not provide sufficient precision. SLAM was used to remove the drift of VO.

Visual and inertial measurements are combined in [2] in order to navigate an unknown indoor environment without the use of GPS. The vision system, carried aboard a quadrotor, demonstrated the ability to perform fast mapping and to avoid obstacles. The motion of the quadrotor was estimated from the frame-by-frame motion of tracked features. These motion estimates were combined with inertial measurements using an extended Kalman filter.

Stereo Vision

As seen in the references in the previous subsection, it is possible to determine motion from a single camera. Doing so however, requires the ability to recognize the size and orientation of something in the scene, whether it be a predetermined marker or pattern, or some previously unknown features that have been discovered and mapped. With two cameras in a stereo configuration, depth can be computed without any previous knowledge other than the position of the two cameras relative to each other (which is typically predetermined and fixed). This allows direct estimation of the three-dimensional motion of the vehicle without the need of assumptions for simplifying the problem.

Using a pair of cameras in a stereo arrangement to perform VO is descried in [1]. The pose (position and orientation) of the stereo cameras is estimated with each incoming image and the derivative is used to calculate the motion of the vehicle. A dense-stereo procedure is employed to verify the pose estimation and the results drive a robust estimate based on the random sample consensus (RANSAC) algorithm. SLAM is used to remove any remaining drift.

Unconventional Configurations

The stereo configuration used in the references mentioned in the previous subsection may seem natural as it imitates the stereo configuration of our own eyes. However, we can expand the possibilities by using more cameras, cameras of different type, or removing the spatial constraint on the location of the cameras.

An interesting approach for estimating position and attitude using a pair of cameras was presented in [3]. Rather than putting the cameras in a fixed stereo configuration, one was attached to a quadrotor and the other was placed on the ground so that both cameras could see each other. This approach was proven stable in flight experiments.

A technique in which cameras of different type are used in a stereo configuration was presented in [11]. In this case an extremely wide-angle (fish-eye) lens was paired with an ordinary rectilinear lens. The authors referred to this configuration as a mixed stereoscopic vision system. It was shown that the disparity between the two cameras could be used to for altitude estimation.

A vision systems employing omnidirectional cameras was explored in [19] for control of a helicopter. Experiments showed that both navigation and obstacle avoidance could be achieved with such a system.

Omnidirectional vision using a catadioptric (curved mirror) lens was demonstrated in [9]. A method for determining the horizon by distinguishing the sky from the ground was discussed in order to efficiently compute pitch and roll angles. Experiments were carried out by post-processing video captured aboard a fixed-wing aircraft.

A combination of optical flow and stereo vision is presented in [18] for the purpose of navigating through an urban environment. The combination of these two techniques was evaluated in real-time experiments. Stereo vision allowed the vehicle to detect obstacles ahead of itself, and optic flow allowed it to turn away from obstacles to the side.

1.4.2 High-Speed Vision-Based Navigation for a Hexacopter

This thesis proposes the application of high-speed monocular SLAM to give precise estimates of vehicle position and attitude. "High-speed" refers to the use of a video stream that presents image frames much faster than the normal video rate of 30 frames per second. While such rates are faster than is required for position control, they affords several advantages such as: a more reliable track, higher trackable velocity (for the given lens and camera geometry), and surplus phase margin for filtering to increase precision and reduce noise.

The research into high-speed vision-based navigation was performed in collaboration with the Ishikawa-Oku laboratory of the University of Tokyo School of Information Science and Technology. Dr. Carson Reynolds and Daniel Henell both contributed extensively to this research.

1.5 Thesis Outline

The following is a brief description of the contents of the coming chapters.

Chapter 2 introduces the mathematics that describes the motion of a hexacopter. The various forces and moments that are applied to the hexacopter are cataloged and modeled. Finally, a linear model is developed for use in designing a simple linear controller.

Chapter 3 proposes and analyzes a controller that can adapt to changes in actuator effectiveness. A method for estimating the effectiveness of the propellers from inertial measurements is proposed. Finally, reconfiguration upon detection of total loss of effectiveness is discussed.

Chapter 4 introduces the parallel tracking and mapping (PTAM) algorithm that is the basis for the proposed high-speed vision-based navigation. Advantages and drawbacks of the PTAM algorithm are discussed and modifications are presented to improve

its capability for navigation. Finally, extensions for controlling the position of a hexacopter are introduced.

Chapter 5 introduces the hardware that was used as the flying testbed for control and navigation experiments. This includes the hexacopter platform and the computer-vision package. Modifications to the off-the-shelf hexacopter platform are described in detail.

Chapter 6 presents the data from simulated and real experiments. These experiments include real tests of the accuracy of vision-based navigation versus GPS and a motion capture system, autonomous vision-based navigation including takeoff and landing, and simulated adaptation to a single-propeller failure.

Chapter 7 concludes the thesis the a discussion of the proposed failure-tolerant control and high-speed vision-based navigation approaches. Suggestions for future work are listed.

Chapter 2

Hexacopter: Modeling and Control

This chapter presents a methodology for designing a simple linear controller for a hexacopter. The process involves first formalizing a physical model of a hexacopter. Insights from that model are then used to form a strategy for decoupling the desired degrees of freedom. Finally the decoupled system is linearized and LQG is used to develop a robust observer and state-feedback controller pair for each degree of freedom.

In order to develop the physical model, it will be assumed that the hexacopter can be approximated as a rigid body so that it can be modeled using Newton-Euler rigid-body dynamics. The following forces and moments will be considered as drivers of the dynamics:

1. The force of gravity, F_g
2. Aerodynamic forces and moments from the propellers, F_{ap} and M_{ap}
3. Gyroscopic moments of from the spinning propellers, M_{gp}
4. Aerodynamic forces and moments from the body, F_{ab} and M_{ab}

2.1 Newton-Euler Rigid-Body Dynamics

In this section a model is developed for how the motion of the hexacopter is affected by the application of forces and moments.

Assumption 1. *The hexacopter can be approximated as a rigid body.*

Assuming that a hexacopter can be closely approximated as a rigid body, it follows that its dynamics can be described by the Newton-Euler formalism of rigid-body dynamics as given in [41] as:

$$\begin{bmatrix} mI_{3\times 3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}} \\ \dot{\mathbf{\Omega}} \end{bmatrix} + \begin{bmatrix} \mathbf{\Omega} \times m\mathbf{V} \\ \mathbf{\Omega} \times I\mathbf{\Omega} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \tag{2.1}$$

where $\mathbf{F}$ is the body-frame force vector acting at the center of mass, $\mathbf{M}$ is the body-frame torque vector acting at the center of mass, m is the hexacopter mass, I is the hexacopter moment of inertia matrix about the center of mass, $\mathbf{V}$ is the body-frame velocity vector at the center of mass, $\mathbf{\Omega}$ is the body-frame angular velocity vector, and $I_{3\times 3}$ is the 3-by-3 identity matrix.

Using the body-fixed frame is beneficial for the following reasons:

1. The moment of inertia matrix is constant for a rigid body
2. Choosing body axes that coincide with principle axes will simply the mathematics, as will be shown later
3. The aerodynamic and gyroscopic forces act along constant directions in the body frame
4. Most sensors on the vehicle sense along constant directions in the body frame

Let the body axis be defined according to figure 2.1.

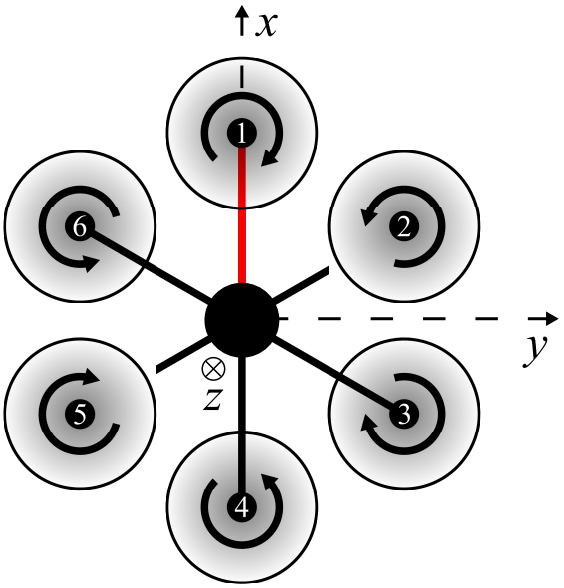


Figure 2.1: A top-down view of a hexacopter showing the directions of the principle body-axis vectors. the x axis points "forward," the y axis points "right," and the x axis points "down."

Assumption 2. *The body axes defined according to figure 2.1 can be considered principle axes.*

The ideal hexacopter configuration shown in figure 2.1 has three planes of symmetry. One corresponds to the x-z body-axis plane, and the other two occur at consecutive 60-degree rotations from the first plane along the z axis. From this information it can be concluded that the z-axis must be one principle axis. Another principle axis can be chosen perpendicular to the z-axis along one of the planes of symmetry. The final principle axis is perpendicular to both of the previous axes. Indeed the body axes satisfy these requirements. It is assumed that the real hexacopter is close to this ideal geometry.

If the body axes are coincident with principle axes, then the moment of inertia matrix

is diagonal and can be written as:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.2}$$

Replacing I in equation 2.1 with the above moment of inertia matrix and expanding the cross products gives:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} m(wq - vr) \\ m(ur - wp) \\ m(vp - uq) \\ (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \tag{2.3}$$

where $\begin{bmatrix} u & v & w \end{bmatrix}^T$ is $\mathbf{V}$ and $\begin{bmatrix} p & q & r \end{bmatrix}^T$ is $\mathbf{\Omega}$. Solving equation 2.3 for the accelerations gives:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}F_x + vr - wq \\ \frac{1}{m}F_y - ur + wp \\ \frac{1}{m}F_z + uq - vp \\ [M_x + (I_{yy} - I_{zz})qr]\frac{1}{I_{xx}} \\ [M_y + (I_{zz} - I_{xx})pr]\frac{1}{I_{yy}} \\ [M_z + (I_{xx} - I_{yy})pq]\frac{1}{I_{zz}} \end{bmatrix} \tag{2.4}$$

Equation 2.4 shows that each acceleration in the body axis comes from a linear response to a force or moment plus a nonlinear term accounting for the motion of the body-axis reference frame relative to an inertial frame.

2.2 Motion in the Earth Frame

In the previous section, a model of the acceleration of a rigid body given the application of forces and moments was developed in the body-fixed frame. However, the goal is to be able to control the attitude or position of the vehicle in the earth frame (assumed to be approximately inertial). To that end, this section will introduce the conversions for motion from the body frame to the earth frame.

The angular relation between the body frame and earth frame (referred to as the vehicle's attitude), can be described through a sequence of three consecutive rotations of an intermediate rotating frame. This intermediate frame begins aligned to the earth frame, but ends aligned to the body frame as a result of the rotations. The convention in aeronautics is to first rotate about the earth z-axis, then about the new y-axis, and finally about the new x-axis. The angle of each rotation is referred to as the heading (ψ), pitch (θ), and roll (ϕ) Euler angles respectively.

Using this convention, a rotation matrix that transforms vector quantities from the earth frame to the body frame and the reverse can be formulated as shown below. Details of the derivation of these matrices are given in appendix C.

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.5}$$

$$R^{-1} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.6}$$

It follows then that the velocity vector in the earth frame can be derived from the velocity vector in the body frame as follow:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\phi, \theta, \psi)^{-1} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.7}$$

Finally, the derivative of the Euler angles can be derived from the angular velocity with the following relation (see appendix C for the derivation):

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.8}$$

2.3 Applied Forces and Moments

2.3.1 Gravity

Gravity acts directly down in the earth frame (along the positive z earth-axis). Therefore, it can readily be converted to the body axis using the rotation matrix in equation 2.5.

$$\mathbf{F}_g = R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\sin\theta \\ g\sin\phi\cos\theta \\ g\cos\phi\cos\theta \end{bmatrix} \tag{2.9}$$

2.3.2 Propeller Aerodynamics

The forces and moments generated by the spinning propellers are the primary means for controlling the hexacopter. However, the aerodynamics of propellers is a very complex phenomenon.

Assumption 3. *The propeller aerodynamics can be approximated using a combination*

of momentum theory and blade-element theory with uniform flow through the propeller disk, without interference from its own wake or the wake of other propellers.

To make modeling of the aerodynamics more tractable, several simplifying assumptions are imposed so that momentum theory can be used to determine the velocity of the air flowing through the propeller disk, and blade element theory can be used to determine the net force and moment at the hub.

The aerodynamic force and moment at the hub can be decomposed into the following components:

$$T_n(\omega_n, w_n, \sqrt{u_n^2 + v_n^2}, \rho) \tag{2.10a}$$

$$D_n(\omega_n, w_n, \sqrt{u_n^2 + v_n^2}, \rho) \tag{2.10b}$$

$$Q_n(\omega_n, w_n, \sqrt{u_n^2 + v_n^2}, \rho, d_n) \tag{2.10c}$$

$$P_n(\omega_n, w_n, \sqrt{u_n^2 + v_n^2}, \rho, d_n) \tag{2.10d}$$

where T_n, D_n, Q_n, and P_n are the thrust, drag (opposite the direction of horizontal motion), propeller torque (about the z body-axis), and rolling (about the direction of horizontal motion) for the n^{th} propeller. Each is a nonlinear function of the n^{th} propeller's rotational speed (ω_n), the velocity of the n^{th} propeller hub along the z body-axis (w_n), the horizontal velocity of the n^{th} propeller hub in the body frame ($\sqrt{u_n^2 + v_n^2}$), and the density of the air (ρ). The sign of Q_n and P_n also depends on the direction of rotation (d_n) of the n^{th} propeller. d_n is defined such that 1 corresponds to right-hand rotation about z body-axis and -1 corresponds to left-hand rotation about z body-axis. The horizontal velocity is the norm of the vector sum of u_n and v_n, which are the velocity of the n^{th} propeller hub along the x and y body-axes respectively.

Figure 2.2 shows how the blade aerodynamics vary with propeller speed, vertical velocity, and horizontal velocity according to the relations in equations 2.10 From the

figure, the following generalizations can be made:

- For a given vertical speed, T and Q vary almost linearly with the square of propeller speed.
- The slopes of the above relationships vary with vertical speed.
- For a given horizontal speed and vertical speed, D and P vary almost linearly with propeller speed.
- The slopes of the above relationships vary with horizontal speed, and the offsets vary with both vertical and horizontal speed.

It should be noted that the above generalizations are only valid under "normal" operating ranges where the propeller is neither "windmilling" (occurs when the vertical velocity is greater than the inflow velocity) nor creating a ring vortex (occurs when the vertical velocity is quite opposite the inflow velocity).

Since the hubs of each of the propellers are distant from the hexacopter's cg, these aerodynamic forces generated at the propeller hubs generate a moment through the corresponding lever-arm. Collecting the aerodynamic moments and also the aerodynamic forces with their corresponding lever-arms in equations 2.10 gives the following forces and moments:

$$\mathbf{F}_{ap} = \begin{bmatrix} -\sum_{i=1}^{6} D_{xi} \\ \sum_{i=1}^{6} D_{yi} \\ \sum_{i=1}^{6} T_i \end{bmatrix} \tag{2.11}$$

$$\mathbf{M}_{ap} = \begin{bmatrix} \sum_{i=1}^{6} \left[-T_i l_{yi} + P_{xi}\right] \\ \sum_{i=1}^{6} \left[T_i l_{xi} + P_{yi}\right] \\ \sum_{i=1}^{6} \left[-D_{xi} l_{yi} + D_{yi} l_{xi} + Q_i\right] \end{bmatrix} \tag{2.12}$$

where l_{xn} and l_{yn} are the lever-arms from the cg to the n^{th} propeller hub along the x

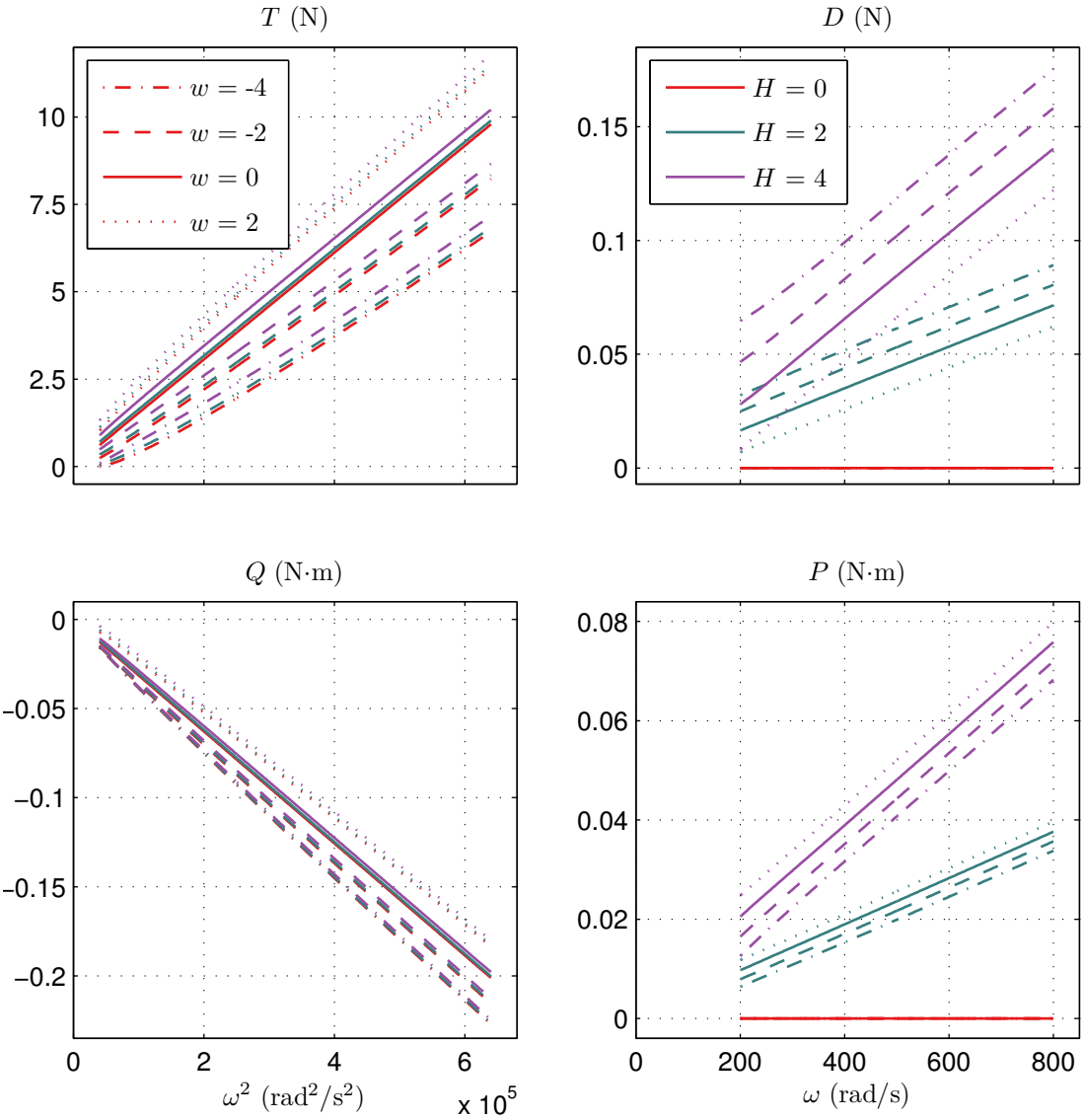


Figure 2.2: Example forces and moments calculated using momentum theory and blade element theory. T_n, D_n, Q_n, and P_n are the thrust, drag (opposite the direction of horizontal motion), propeller torque (about the shaft), and rolling (about the direction of horizontal motion) for the n^{th} propeller. Each is a nonlinear function of the n^{th} propeller's rotational speed (ω_n), the velocity of the n^{th} propeller hub along the z body-axis (w_n), the horizontal velocity of the n^{th} propeller hub in the body frame (H_n), and the density of the air (ρ).

and y body-axes respectively. T_n, D_n, P_n, and Q_n are all functions of ω_n, w_n, H_n, ρ, and d_n (omitted from the above notation for brevity).

2.3.3 Gyroscopic Moments from the Spinning Propellers

Let $\mathbf{M}_{gp}$ be the body-frame reaction moment vector due to the acceleration and rotation of the spinning propellers. Since it is the reaction moments, it can be formulated as the opposite of the moments that are responsible for the acceleration and rotation the propellers. Note that the angular motion of the propellers is restricted to the z body-axis:

$$\mathbf{M}_{gp_n} = -I_p \begin{bmatrix} 0 \\ 0 \\ d_n\dot{\omega}_n \end{bmatrix} - I_p\Omega \times \begin{bmatrix} 0 \\ 0 \\ d_n\omega_n \end{bmatrix} = \begin{bmatrix} -d_n I_{p_{zz}} q\omega_n \\ d_n I_{p_{zz}} p\omega_n \\ -d_n I_{p_{zz}}\dot{\omega}_n \end{bmatrix} \tag{2.13}$$

where I_p is the propeller moment of inertia matrix (which is diagonal due the symmetry of the propellers).

Summing over the six propellers gives:

$$\mathbf{M}_{gp} = I_{p_{zz}} \begin{bmatrix} -q \cdot \sum_{i=1}^{6}(d_i\omega_i) \\ p \cdot \sum_{i=1}^{6}(d_i\omega_i) \\ -\sum_{i=1}^{6}(d_i\dot{\omega}_i) \end{bmatrix} \tag{2.14}$$

2.3.4 Body Aerodynamics

Assumption 4. *Body aerodynamics contribute only viscous drag.*

Body aerodynamics is likely the least significant contributor of force and moments of the sources consider thus far. In order to simplify the analysis, let us assume that the

body's sole aerodynamic contribution is drag, which can be represented as:

$$\mathbf{F}_{ab} = -\frac{\rho}{2}\begin{bmatrix} C_x A_x u|u| \\ C_y A_y v|v| \\ C_z A_z w|w| \end{bmatrix} \tag{2.15}$$

where C_x, C_y, and C_z are non-dimensional drag coefficients, and A_x, A_y, and A_z are reference areas.

2.3.5 Forces and Moments Summary

The forces and moments modeled in this section can be collected and substituted into equation 2.4 to give the following relations for the acceleration of the hexacopter in the body axis:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \left[-g\sin\theta - \frac{\rho}{2}C_x A_x u|u| + \sum_{i=1}^{6} D_{ix}\right]\frac{1}{m} + vr - wq \\ \left[g\sin\phi\cos\theta - \frac{\rho}{2}C_y A_y v|v| + \sum_{i=1}^{6} D_{iy}\right]\frac{1}{m} + wp - ur \\ \left[g\cos\phi\cos\theta - \frac{\rho}{2}C_z A_z w|w| - \sum_{i=1}^{6} T_i\right]\frac{1}{m} + uq - vp \\ \left\{\sum_{i=1}^{6}\left[P_{ix} - T_i l_{yi} - I_{p_{zz}} q d_i \omega_i\right] + (I_{yy} - I_{zz})qr\right\}\frac{1}{I_{xx}} \\ \left\{\sum_{i=1}^{6}\left[P_{iy} + T_i l_{xi} + I_{p_{zz}} p d_i \omega_i\right] + (I_{zz} - I_{xx})pr\right\}\frac{1}{I_{yy}} \\ \left\{\sum_{i=1}^{6}\left[Q_i - D_{xi} l_{yi} + D_{yi} l_{xi} - I_{p_{zz}} d_i \dot{\omega}_i\right] + (I_{xx} - I_{yy})pq\right\}\frac{1}{I_{zz}} \end{bmatrix} \tag{2.16}$$

2.4 Actuator Dynamics

Assumption 5. *The response to propeller speed commands can be approximated with a lag filter.*

An actuator refers to some mechanism for affecting the motion of a physical system. In the case of the hexacopter, the actuators are the six propellers, each driven by an electric motor. Either a brushed or brushless motor may be used for this purpose. Brushless motors require an active controller for speed control. Brushed motors can be

operated open-loop, but speed will vary with both load and applied voltage, so some feedback controller must be used to guarantee faithful tracking of speed commands.

In either case, it is assumed that the bandwidth of the controlled motor/propeller pair can be modeled by as simple lag filter as follows:

$$\dot{\omega}_n = \tau^{-1}(u_n - \omega_n) \tag{2.17}$$

where u_n is the speed command for the n^{th} propeller.

2.5 Linear Model

In preparation for designing a simple linear controller, it is necessary to construct a linear model of the system. Let us first note that the system dynamics can be fully described by the differential equations 2.7, 2.8, 2.16, and 2.17.

These equations can be linearized by first choosing an operating condition (nominal values for the dependent variables) and then finding the constant of proportionality for the change in each of the derivatives due to an infinitesimally small perturbation of each of the dependent variables (considered separately).

An obvious choice for the nominal operating condition is the hover condition. During hover the hexacopter is stationary with respect to the earth, so $\mathbf{V}$, and $\mathbf{\Omega}$ are zero, which further implies from equation 2.16 that θ and ϕ are zero. The propellers are spinning at some constant velocity to generate thrust to counteract the force of gravity. ψ can be assigned arbitrarily since it has no effect on the motion of the hexacopter other than the orientation of subsequent motions relative to the z earth-axis. x, y, and z do not appear as dependent variables and therefore have no effect on the linearization.

Using the above conditions, and arbitrarily setting ψ to zero, results in the following

sets of independent linear equations (separated for readability):

$$\dot{\Delta\boldsymbol{\omega}} = -\frac{1}{\tau} I_{6\times 6}\Delta\boldsymbol{\omega} + \frac{1}{\tau} I_{6\times 6}\mathbf{u} \tag{2.18}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} \\ 0 & 0 & 0 & -\frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & -\frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & 0 & \frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & \frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} \\ 0 & 0 & |l|\frac{k_T}{I_{yy}} & \frac{|l|}{2}\frac{k_T}{I_{yy}} & -\frac{|l|}{2}\frac{k_T}{I_{yy}} & -|l|\frac{k_T}{I_{yy}} & -\frac{|l|}{2}\frac{k_T}{I_{yy}} & \frac{|l|}{2}\frac{k_T}{I_{yy}} \\ 0 & 0 & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \Delta\boldsymbol{\omega} \end{bmatrix} \tag{2.19}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = I_{3\times 3} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.20}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = I_{3\times 3} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.21}$$

where $\Delta\boldsymbol{\omega}$ is the vector of deviations from the hover rotational speed of the propellers. k_T and k_Q are the constants of proportionality for thrust and torque respectively due to a change in rotational speed of a propeller.

For ease of notation, let us define the matrix E as the following from equation 2.19:

$$E = \begin{bmatrix} -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} & -\frac{k_T}{m} \\ 0 & -\frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & -\frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & 0 & \frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} & \frac{\sqrt{3}|l|}{2}\frac{k_T}{I_{xx}} \\ |l|\frac{k_T}{I_{yy}} & \frac{|l|}{2}\frac{k_T}{I_{yy}} & -\frac{|l|}{2}\frac{k_T}{I_{yy}} & -|l|\frac{k_T}{I_{yy}} & -\frac{|l|}{2}\frac{k_T}{I_{yy}} & \frac{|l|}{2}\frac{k_T}{I_{yy}} \\ -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \tag{2.22}$$

Let us call this matrix the *actuator mixing matrix*, since it describes how the actuators

mix to affect the controllable degrees of freedom. This matrix was derived with a little bit of foresight, making assumption that all propellers are rotating with the same speed in hover. If that is true, then linearization of thrust and torque will result in the same constants of proportionality for each propeller. Also, the geometry of figure 2.1 was assumed when determining the lever arms.

Next, let us attempt to restate equations 2.18 and 2.19 as a function of the second derivative of $\mathbf{V}$ and $\mathbf{\Omega}$ through a change of variables. First, let us take the derivative of the last 4 rows of equation 2.19 and notice that:

$$\begin{bmatrix} \ddot{w} \\ \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} = E(\dot{\Delta\boldsymbol{\omega}}) \tag{2.23}$$

Next, let us multiply 2.18 by E then substitute in 2.23 and the bottom 4 rows of equation 2.19 to obtain:

$$\begin{aligned} E(\dot{\Delta\boldsymbol{\omega}}) &= -\frac{1}{\tau}E(\Delta\boldsymbol{\omega}) + \frac{1}{\tau}E\mathbf{u} \\ \Rightarrow \begin{bmatrix} \ddot{w} \\ \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} &= -\frac{1}{\tau}\begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \frac{1}{\tau}E\mathbf{u} \end{aligned} \tag{2.24}$$

Assembling equations 2.24, 2.20, and 2.21 gives the entire linearized dynamics as

follows:

$$\begin{bmatrix} \ddot{w} \\ \ddot{p} \\ \ddot{q} \\ \ddot{r} \\ \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ u \\ v \\ w \\ p \\ q \\ r \\ x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} + \frac{1}{\tau} E u \tag{2.25}$$

2.6 Simple Control

For a traditional helicopter, control of $\dot{w}$, $\dot{p}$, $\dot{q}$, and $\dot{r}$ is decoupled through clever mechanical design so no active control is needed. In the case of multi-rotors like quadrotors and hexacopter, each element of the control, $\mathbf{u}$, affects several state rates through the mixing in matrix E.

2.6.1 Quadrotor Control

In the case a quadrotor, linearizing the dynamics about the hover condition results in the same dynamics given in equation 2.25, but with the following actuator mixing matrix:

$$E = \begin{bmatrix} \frac{-k_T}{m} & \frac{-k_T}{m} & \frac{-k_T}{m} & \frac{-k_T}{m} \\ 0 & -|l|\frac{k_T}{I_{xx}} & 0 & |l|\frac{k_T}{I_{xx}} \\ |l|\frac{k_T}{I_{yy}} & 0 & -|l|\frac{k_T}{I_{yy}} & 0 \\ -\frac{k_Q}{I_{yy}} & \frac{k_Q}{I_{yy}} & -\frac{k_Q}{I_{yy}} & \frac{k_Q}{I_{yy}} \end{bmatrix} \tag{2.26}$$

This quadrotor version of the E matrix is square and non-singular, so it is readily invertible. This means that an input $\mathbf{u}$ can be constructed from decoupled acceleration commands as follows:

$$u = E^{-1} \begin{bmatrix} \dot{w}_{cmd} \\ \dot{p}_{cmd} \\ \dot{q}_{cmd} \\ \dot{r}_{cmd} \end{bmatrix} \tag{2.27}$$

in which case, E cancels with E^{-1} and the dynamics decouple into the following four equations:

$$\begin{bmatrix} \ddot{w} \\ \dot{w} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{w} \\ w \\ z \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \end{bmatrix} \dot{w}_{cmd} \tag{2.28}$$

$$\begin{bmatrix} \ddot{p} \\ \dot{p} \\ \dot{\phi} \\ \dot{v} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{p} \\ p \\ \phi \\ v \\ y \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \dot{p}_{cmd} \tag{2.29}$$

$$\begin{bmatrix} \ddot{q} \\ \dot{q} \\ \dot{\theta} \\ \dot{u} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q} \\ q \\ \theta \\ u \\ x \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \dot{q}_{cmd} \tag{2.30}$$

$$\begin{bmatrix} \ddot{r} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{r} \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \end{bmatrix} \dot{r}_{cmd} \tag{2.31}$$

2.6.2 Hexacopter Control

In the case of a hexacopter, the dynamics are under-constrained, meaning that there are more variables than degrees of freedom. As such, there are infinite possible combinations of pseudo-inversions of the E matrix. A good solution is the Moore-Penrose pseudoinverse as proposed by Baranek [5], which results in the following relationship:

$$u = \begin{bmatrix} -\frac{m}{6k_T} & 0 & \frac{1}{3|l|}\frac{I_{yy}}{k_T} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6k_T} & -\frac{\sqrt{3}}{6|l|}\frac{I_{xx}}{k_T} & \frac{1}{6|l|}\frac{I_{yy}}{k_T} & \frac{I_{zz}}{6k_Q} \\ -\frac{m}{6k_T} & -\frac{\sqrt{3}}{6|l|}\frac{I_{xx}}{k_T} & -\frac{1}{6|l|}\frac{I_{yy}}{k_T} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6k_T} & 0 & -\frac{1}{3|l|}\frac{I_{yy}}{k_T} & \frac{I_{zz}}{6k_Q} \\ -\frac{m}{6k_T} & \frac{\sqrt{3}}{6|l|}\frac{I_{xx}}{k_T} & -\frac{1}{6|l|}\frac{I_{yy}}{k_T} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6k_T} & \frac{\sqrt{3}}{6|l|}\frac{I_{xx}}{k_T} & \frac{1}{6|l|}\frac{I_{yy}}{k_T} & \frac{I_{zz}}{6k_Q} \end{bmatrix} \begin{bmatrix} \dot{w}_{cmd} \\ \dot{p}_{cmd} \\ \dot{q}_{cmd} \\ \dot{r}_{cmd} \end{bmatrix} \tag{2.32}$$

Using the Moore-Penrose pseudoinverse enforces the hexagonal symmetry of the hexacopter. In this way, a pure $\dot{w}_{cmd}$ is distributed evenly across each of the propellers (confirming our previous assumption that all propellers should have the same speed during hover) and angular acceleration about each of the arms looks identical but rotated 60 degrees.

With the dynamics decoupled, feedback control can be applied separately to each axis. Only some of the rates are measurable using typical sensor packages, but the rest are observable so an observer may be used to estimate the other states in order to arbitrarily place the poles of the closed-loop dynamics with state feedback.

2.6.3 Linear Quadratic Gaussian Controller

A well-known approach to control design for linear systems with incomplete and noisy state measurements is the linear quadratic Gaussian (LQG) controller. This controller combines a linear quadratic regulator (LQR) for optimal state feedback, with state estimates from alinear quadratic estimator (LQE) (i.e. Kalman filter).

LQR is an approach that finds an optimal solution to the goal of driving the states of a linear system to zero from some non-zero initial condition. State feedback gains are solved for that will result in a response that minimizes a cost function. The cost function is parameterized so that the control designer may decide how heavily to penalize deviations of the state vector and control signal from zero.

LQE, or the Kalman filter, is a method to optimally blend noisy and/or incomplete state measurements with predictions based on some knowledge of the system dynamics to estimate the full state vector. This method assumes that the measurements and states are subject to some Gaussian noise, referred to as the measurement noise and process noise respectively.

2.6.4 Controller Modifications

The LQG controller described in the previous section is meant for linear systems, but the hexacopter is not a linear system. However, with good choices of LQR cost-function parameters, the resulting LQG gives good performance in a large region around the hover operating condition. Still, some modifications to the LQG controller should made to ensure that control remains stable, or to improve performance.

Square Propeller Speed

As mentioned in section 2.3.2, for a given vertical speed, T and Q vary almost linearly with the square of propeller speed. Therefore, equation 2.32 should be reformulated so that the constants of proportionality, k_T and k_Q, are with respect to the square of propeller speed and $\mathbf{u}$ is the square-root of the result.

As an aside, it was found that it is not unreasonable to consider the square of propeller speed as a lag of the square of the propeller speed command. This means that the linear model in equation 2.25 can be maintained with E reformulated as described above, and u as the command of the square of propeller speeds.

Transit Speed Through Error Saturation

With a linear controller, a large error gives a large command. However, the actuators have operational limits (i.e. there is a maximum speed that the motor controllers are able to achieve). The commands issued to the actuators should never exceed their operational limits. One way to achieve this is to apply a limit to the error.

Positional error is particularly critical since it could become extremely large in the case of waypoint navigation and therefore must be limited. Conveniently, the value of this limit determines the speed of transit. As a proof, let us first assume that the hexacopter is at steady state; traveling with zero heading, at constant velocity and attitude, and that the target destination is distant so the positional error is being limited (i.e. constant). Then equation 2.30 becomes:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \theta_{\text{steady}} \\ u_{\text{steady}} \\ x_{\text{limit}} \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \dot{q}_{cmd} \tag{2.33}$$

where $\dot{q}_{cmd}$ is the result of state feedback, so:

$$\dot{q}_{cmd} = \mathbf{K} \cdot x = K_{\dot{q}} \cdot 0 + K_q \cdot 0 + K_\theta \cdot \theta_{\text{steady}} + K_u \cdot u_{\text{steady}} + K_x \cdot x_{\text{limit}} \tag{2.34}$$

where $\mathbf{K}$ is the state feedback gain vector. Equation 2.33 implies that $\dot{q}_{cmd}$ must be zero in this case. Making the further (reasonable)assumption that $K_\theta \cdot \theta_{\text{steady}}$ is small and solving equation 2.34 for u_{steady} gives:

$$u_{\text{steady}} \approx \frac{K_x}{K_u} x_{\text{limit}} \tag{2.35}$$

Appending Integrators

The hexacopter is most likely subject to some non-zero-mean disturbances. For example, misalignment of a rotor, or deviation of the CG from the ideal location will mean that the propellers will have to produce slightly different thrust in order to maintain equilibrium. Integral control is typically used to deal with such steady-state errors.

In the context of the LQR, this is achieved by appending an extra state to the dynamics and setting its derivative to the state that is desired to be regulated. For the example, in the case of controlling altitude, integral control is achieved by solving the LQG problem for equation 2.28 with an extra state appended as follows:

$$\begin{bmatrix} \ddot{w} \\ \dot{w} \\ \dot{z} \\ \dot{z}_{\text{int}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{w} \\ w \\ z \\ z_{\text{int}} \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} \\ 0 \\ 0 \\ 0 \end{bmatrix} \dot{w}_{cmd} \tag{2.36}$$

Integrator Regions (Anti-Windup)

The function of the integrator described above is to push the command in the direction opposite sustained error to force the system toward zero error. However, in some cases,

sustained error is expected and allowing the integrator to accumulate the error will lead to an unnecessarily large command in the direction opposite the error, which will ultimately lead to a large overshoot (or instability or actuator saturation). For example, in the case of waypoint navigation, positional error is expected to be sustained as the hexacopter is in transit.

One strategy to prevent unnecessary winding of the integrator is to prevent its action outside of some neighborhood of zero error. In the case of waypoint navigation, we may be confident that the hexacopter will always be capable of arriving at within some radius of the waypoint without the use of integral control, so the integrator is forbidden from accumulating outside of that radius. Increasing the radius will increase confidence that the hexacopter will arrive at its destination, but will also increase overshoot.

Attitude Washout

So far, discussion has been focused on position control. However, in cases where position information is not available or not reliable, a human pilot may assume the duty of outer-loop position controller. In that case the pilot manipulates the hexacopter's attitude and thrust through visual feedback. The hexacopter's controller should then be responsible for tracking the attitude and thrust commands through sensor feedback. However, typically attitude is estimated by integrating on-board gyros that measure the hexacopter's angular rates. Any noise in those measurements will also be integrated, causing the estimates to drift. Left unchecked, the pilot may have to issue increasingly larger commands to counteract the drifting attitude estimates, eventually reaching a command limit, catastrophically losing the ability to counteract further drift.

A solution to this problem is to apply a high-pass filter to the attitude estimates to gently push them back towards zero. However, the cutoff frequency must be chosen very carefully since this filter will cause an instability. For a sustained attitude command, the controller will drive the attitude to the command. The low-pass filter will then reduce

the estimate of the attitude and the controller will drive the attitude further in order to satisfy the command. This is a divergent situation. Therefore, the cutoff frequency must be made as low as possible so that the divergence is very slow and well within the pilot's control bandwidth. However, the cutoff frequency should not be made so low that the drift can overwhelm the filter.

Chapter 3

Failure-Tolerant Control

In section 1.2.3 it was stated that Hexacopters and Octocopters can recover control of all 4 desired degrees of control freedom upon total loss of effectiveness of a single propeller. This chapter proposes a failure-tolerant controller for this purpose.

3.1 Control Inversion

3.2 Adaptive Compensation for Actuator Effectiveness

There are several ways to estimate propeller effectiveness. Thrust and torque could be measured directly with sensors, or could be estimated from the current draw or speed of the driving motor. However, in many cases (particularly in low-cost designs) measurements of those quantities are not available. Typically, propeller speed commands are issued to motor controllers and it is simply assumed that the motor controllers dutifully track the commands. However, even without any information fed back from the motor controllers, it is still possible to estimate propeller effectiveness by comparing the measured dynamics to the modeled system dynamics.

As a first step, let us explore the possibility of adapting to deviation from design

actuator effectiveness for a certain class of systems. Let us suppose that the ideal (model) system can be represented by the following differential equation:

$$\dot{\mathbf{x}}_m(t) = f_m(\mathbf{x}_m(t)) + B_m\mathbf{r}(t) \tag{3.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state of the system; $f : \mathbb{R}^n \to \mathbb{R}^n$ is a possibly non-linear state function that describes how the system state effects the state derivatives; $B \in \mathbb{R}^{n\times m}$ is a constant matrix that describes how ideally effective actuators affect the state derivatives; and $\mathbf{r}(t) \in \mathbb{R}^m$ is a reference actuator state. The subscript m indicates that this is the model of the ideal system.

Without loss of generality, the system description in equation 3.1 can be divided as follows:

$$\begin{bmatrix} \dot{\mathbf{x}}_{mc}(t) \\ \dot{\mathbf{x}}_{mu}(t) \end{bmatrix} = \begin{bmatrix} f_{mc}(\mathbf{x}_m(t)) \\ f_{mu}(\mathbf{x}_m(t)) \end{bmatrix} + \begin{bmatrix} B_{mc} \\ B_{mu} \end{bmatrix} \mathbf{r}(t) \tag{3.2}$$

where $\mathbf{x}_c(t) \in \mathbb{R}^m$, $\mathbf{x}_u(t) \in \mathbb{R}^{n-m}$, $f_c : \mathbb{R}^n \to \mathbb{R}^m$, $f_u : \mathbb{R}^n \to \mathbb{R}^{m-n}$, $B_c \in \mathbb{R}^{m\times m}$, and $B_u \in \mathbb{R}^{(n-m)\times m}$. The subscript c indicates the grouping of state derivatives that are desired to be directly controlled and the subscript u indicates the grouping of state derivatives that are to be uncontrolled.

3.2.1 Unknown Actuator Effectiveness

Now suppose that in reality, the effectiveness of the actuators is either not precisely known, or has changed due to some damage, failure, etc. In that case, the description of the real system can be written as:

$$\begin{bmatrix} \dot{\mathbf{x}}_c(t) \\ \dot{\mathbf{x}}_u(t) \end{bmatrix} = \begin{bmatrix} f_c(\mathbf{x}(t)) \\ f_u(\mathbf{x}(t)) \end{bmatrix} + \begin{bmatrix} B_c \\ B_u \end{bmatrix} \Lambda(t)\mathbf{u}(t) \tag{3.3}$$

where $\Lambda(t) \in \mathbb{R}^{+m\times m}$ is a positive diagonal matrix representing actuator effectiveness, and $\mathbf{u}(t) \in \mathbb{R}^m$ is the real actuator state.

It is desired to construct an actuator state, $\mathbf{u}$, such that the response of the real system to this actuator state will match the response of the ideal (model) system to the reference actuator state, $\mathbf{r}$. By inspection of equations 3.2 and 3.3, it follows that the response of the real system to $\mathbf{u}$ and the response of the ideally effective system to $\mathbf{r}$, will match if the modeled dynamics perfectly describe the real system and $\mathbf{u} = \Lambda^{-1}\mathbf{r}$.

However, $\mathbf{u}$ cannot be constructed from $\Lambda^{-1}\mathbf{r}$ since the diagonal matrix of actuator effectiveness, Λ, is not known a priori. Rather, let us define $\hat{\Lambda}$ as the estimate of Λ (the hat indicates an estimated value), and construct $\mathbf{u}$ such that $\mathbf{u} = \hat{\Lambda}^{-1}\mathbf{r}$. Then the real system dynamics become:

$$\begin{bmatrix} \dot{\mathbf{x}}_c(t) \\ \dot{\mathbf{x}}_u(t) \end{bmatrix} = \begin{bmatrix} f_c(\mathbf{x}(t)) \\ f_u(\mathbf{x}(t)) \end{bmatrix} + \begin{bmatrix} B_c \\ B_u \end{bmatrix} \Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \tag{3.4}$$

Note that if the state, $\mathbf{x}$, and the state derivatives that are desired to be controlled, $\dot{\mathbf{x}}_c$, are perfectly measurable, then Λ is the only unknown in the top row of equation 3.4. However, Λ often cannot be solved for directly since one or more elements of $\mathbf{r}$ may be (close to) zero. Rather, let us attempt to define an error and a derivative of $\hat{\Lambda}$ such that $\hat{\Lambda}$ approaches Λ (i.e. an improved estimate) whenever $\mathbf{r}$ is not (close to) zero.

3.2.2 Adaptive Law

First, let us define error as the difference between the real and modeled state derivatives:

$$\mathbf{e}(t) = \dot{\mathbf{x}}_c(t) - \dot{\mathbf{x}}_{mc}(t) \tag{3.5}$$

Similarly, measured error can be written as:

$$\tilde{\mathbf{e}}(t) = \tilde{\dot{\mathbf{x}}}_c(t) - \dot{\mathbf{x}}_{mc}(t) \tag{3.6}$$

where a tilde indicates a measured value.

Substituting equations 3.2 and 3.4 into the equation 3.6 gives:

$$\tilde{\mathbf{e}}(t) = [f_c(\mathbf{x}(t)) + B_c\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)] - [f_{mc}(\tilde{\mathbf{x}}(t)) + B_{mc}\mathbf{r}(t)] \tag{3.7}$$

Making the assumptions that the real system perfectly matches the model (i.e. $f_c = f_{mc}$ and $B_c = B_{mc}$) and that the system state is perfectly measurable (i.e. $\tilde{\mathbf{x}} = \mathbf{x}$ and $\tilde{\mathbf{e}} = \mathbf{e}$), equation 3.7 can be rewritten as:

$$\begin{aligned}\mathbf{e}(t) =& [\cancel{f_{mc}(\mathbf{x}(t))} + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)] - [\cancel{f_{mc}(\mathbf{x}(t))} + B_{mc}\mathbf{r}(t)] \\ =& B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t)\end{aligned} \tag{3.8}$$

Notice that if the estimate $\hat{\Lambda}$ approaches Λ, then error will approach zero.

Next, let us make the further assumption that the system is critically actuated, so the number of actuators is equal to the number of state rates that are effected by the actuators, and the corresponding B_{mc} is non-singular. With this assumption, let us propose the definition for the derivative of $\hat{\Lambda}$ as:

$$\dot{\hat{\Lambda}}(t) = \gamma \, \mathrm{diag}(B_{mc}^{-1}\tilde{\mathbf{e}}(t)) \, \hat{\Lambda}(t) \, \mathrm{diag}[g(\mathbf{r}(t), \epsilon)] \tag{3.9}$$

where $\gamma, \epsilon \in \mathbb{R}^+$ are constants to be defined by the control designer, and $g(\mathbf{r}(t), \epsilon)$ is a

modified inverting function given by:

$$(g(\alpha,\epsilon))_i = \begin{cases} {}^{1}\!/_{\alpha_i} & |\alpha_i| > \epsilon \\ {}^{\alpha_i}\!/_{\epsilon^2} & |\alpha_i| \leq \epsilon \end{cases} \tag{3.10}$$

Assuming perfect measurement, equation 3.8 can be substituted into equation 3.9, and properties of diagonal matrices can be employed to simplify the estimation dynamics to:

$$\begin{aligned} \dot{\hat{\Lambda}}(t) &= \gamma \operatorname{diag}[\cancel{B_{mc}^{-1}B_{mc}}(\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\mathbf{r}(t)]\hat{\Lambda}(t) \operatorname{diag}[g(\mathbf{r}(t),\epsilon)] \\ &= \gamma\ (\Lambda(t)\hat{\Lambda}^{-1}(t) - I) \operatorname{diag}(\mathbf{r}(t))\hat{\Lambda}(t) \operatorname{diag}[g(\mathbf{r}(t),\epsilon)]^{\S} \\ &= \gamma\ (\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\hat{\Lambda}(t) \operatorname{diag}(\mathbf{r}(t))\operatorname{diag}[g(\mathbf{r}(t),\epsilon)]^{\dagger} \\ &= \gamma\ (\Lambda(t)\cancel{\hat{\Lambda}^{-1}(t)\hat{\Lambda}(t)} - \hat{\Lambda}(t)) \operatorname{diag}(\mathbf{r}(t))\operatorname{diag}[g(\mathbf{r}(t),\epsilon)] \\ &= \gamma\ (\Lambda(t) - \hat{\Lambda}(t)) \operatorname{diag}(\mathbf{r}(t)) \operatorname{diag}[g(\mathbf{r}(t),\epsilon)] \end{aligned} \tag{3.11}$$

In order to simplify future notation, let us define a few new matrices as follows:

$$\begin{aligned} R(t) &= \operatorname{diag}(\mathbf{r}(t)) \\ G(t,\epsilon) &= \operatorname{diag}[g(\mathbf{r}(t),\epsilon)] \\ \mathrm{K}(t,\epsilon) &= R(t)G(t,\epsilon) \end{aligned} \tag{3.12}$$

$^{\dagger}\operatorname{diag}(\vec{a})\operatorname{diag}(\vec{b}) = \operatorname{diag}(\vec{b})\operatorname{diag}(\vec{a})$

$^{\ddagger}\operatorname{diag}(\vec{a})\vec{b} = \operatorname{diag}(\vec{b})\vec{a}$

$^{\S}\operatorname{diag}(\operatorname{diag}(\vec{a})\vec{b}) = \operatorname{diag}(\vec{a})\operatorname{diag}(\vec{b})$

$^{\P}\operatorname{diag}(\vec{a}+\vec{b}) = \operatorname{diag}(\vec{a}) + \operatorname{diag}(\vec{b})$

Upon inspection of K, we find that the elements are:

$$\kappa_{ii}(t,\epsilon) = \begin{cases} 1 & |r_i(t)| > \epsilon \\ r_i^2(t)/\epsilon^2 & |r_i(t)| \leq \epsilon \end{cases} \tag{3.13}$$

which indicates that K is a positive-diagonal matrix.

Substituting matrix K into equation 3.11 gives:

$$\dot{\hat{\Lambda}}(t) = \gamma \left(\Lambda(t) - \hat{\Lambda}(t)\right) \mathrm{K}(t,\epsilon) \tag{3.14}$$

Note that the above equation is strictly diagonal, so each diagonal element can be considered individually as follows:

$$\dot{\hat{\lambda}}_{ii}(t) = \gamma \left(\lambda_{ii} - \hat{\lambda}_{ii}(t)\right) \kappa_{ii}(t,\epsilon) \tag{3.15}$$

Now let us consider the above equation in 3 cases at some time t:

1. $r_i(t) = 0$
2. $|r_i(t)| > \epsilon$
3. $0 < |r_i(t)| \leq \epsilon$

3.2.3 Analysis Case 1

In the first case, $r_i = 0$ makes $\kappa_{ii} = 0$, thereby simplifying equation 3.15 to:

$$\dot{\hat{\lambda}}_{ii}(t) = 0 \tag{3.16}$$

So at time t, $\hat{\lambda}_{ii}$ is neither converging toward nor diverging from the true value, λ_{ii} (i.e. critically stable).

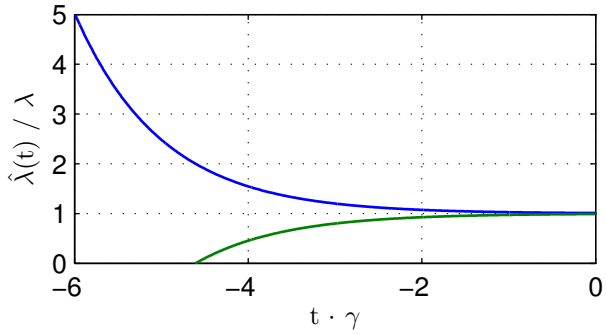


Figure 3.1: Approach trajectories of actuator effectiveness estimates.

3.2.4 Analysis Case 2

In the second case, $|r_i| > \epsilon$ makes $\kappa_{ii} = 1$, thereby simplifying equation 3.15 to:

$$\dot{\hat{\lambda}}_{ii}(t) = \gamma \left(\lambda_{ii} - \hat{\lambda}_{ii}(t)\right) \tag{3.17}$$

Taking the Laplace transform of this differential equation gives the following transfer function:

$$\frac{\hat{\lambda}_{ii}(s)}{\lambda_{ii}(s)} = \frac{1}{\gamma^{-1}s + 1} \tag{3.18}$$

which is a simple first-order lag of λ_{ii} with time-constant γ^{-1}.

Figure 3.1 shows the positive and negative approach trajectories for $\hat{\lambda}_{ii}$ to a constant λ_{ii}, moving backwards in time from when $\hat{\lambda}_{ii}$ is within 1% of the value of λ_{ii}.

Notice that increasing γ increases the speed of the approach.

3.2.5 Analysis Case 3

In the third case, $0 < |r_i(t)| \leq \epsilon$ makes $\kappa_{ii}(t, \epsilon) = {}^{r_i^2(t)}/_{\epsilon^2}$, and equation 3.15 becomes:

$$\dot{\hat{\lambda}}_{ii}(t) = \gamma \left(\lambda_{ii} - \hat{\lambda}_{ii}(t)\right) \frac{r_i^2(t)}{\epsilon^2} \tag{3.19}$$

The case statement $0 < |r_i(t)| \leq \epsilon$ can be rephrased as $0 < {}^{r_i^2(t)}/_{\epsilon^2} \leq 1$. Therefore,

the effect of $r_i^2(t)/\epsilon^2$ in the above equations is equivalent to decreasing the value of γ, i.e. a slower asymptotic approach to λ_{ii}.

3.2.6 Error Dynamics

Sections 3.2.3 through 3.2.5 prove that, assuming perfect measurement and a perfect model of the system, the estimates of actuator effectiveness, $\hat{\Lambda}$, are guaranteed to asymptotically approach the true values, Λ, whenever $\mathbf{r} \neq 0$. Furthermore, the speed of approach will be maximum (i.e. time constant of γ^{-1}) whenever $|\mathbf{r}| \geq \epsilon$.

Inspection of equation 3.8 has already shown that error, $\mathbf{e}$, goes to zero when $\hat{\Lambda}$ approaches Λ. Now let us examine the error dynamics during that approach.

Taking the derivative of equation 3.8 with respect to time gives:

$$\begin{aligned}\dot{\mathbf{e}}(t) = & B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\dot{\mathbf{r}}(t) \\ & + B_{mc}\dot{\Lambda}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\ & - B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\dot{\hat{\Lambda}}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)\end{aligned} \tag{3.20}$$

Let us make a few assumptions to simplify this analysis. First, let us assume that real actuator effectiveness does not change often (i.e. damage or failures are rare) so $\dot{\Lambda}$ is typically zero and therefore the second term in equation 3.20 is typically zero. Second, we have just shown that the estimate $\hat{\Lambda}$ is guaranteed to approach a constant Λ, so the first term in this equation is also guaranteed to go to zero. Furthermore, since $\mathbf{r}$ is usually bounded and smooth for actuators, $\dot{\mathbf{r}}$ is also bounded and cannot be sustained away from zero. Therefore, we can ignore the first two terms while being aware that they may occasionally perturb the remaining dynamics.

Ignoring the first two terms of equation 3.20 and substituting in equation 3.8 and

the simplified notation of equations 3.12 gives:

$$\begin{aligned}\dot{\mathbf{e}}(t) = & -B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)[\gamma\ \mathrm{diag}(B_{mc}^{-1}\mathbf{e}(t))\ \cancel{\hat{\Lambda}(t)}\ G(t,\epsilon)]\cancel{\hat{\Lambda}^{-1}(t)}\mathbf{r}(t)^{\dagger} \\ = & -\gamma\ B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)G(t,\epsilon)\ \mathrm{diag}(\mathbf{r}(t))B_{mc}^{-1}\mathbf{e}(t)^{\dagger\ddagger} \\ = & \underbrace{-\gamma\ B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon)B_{mc}^{-1}}_{A}\mathbf{e}(t) \end{aligned} \tag{3.21}$$

According to [23] The stability of the above equation can be determined by analyzing the eigenvalues of matrix A, where eigenvalues are all of the solutions, ν, to the equation:

$$\det\{A - \nu I\} = 0 \tag{3.22}$$

Expanding matrix A and using properties of determinants, equation 3.22 becomes:

$$\begin{gathered}\det\{-\gamma\ B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon)B_{mc}^{-1} - \nu I\} = 0 \\ \Downarrow \\ \det\{B_{mc}[-\gamma\ \Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon) - \nu I]B_{mc}^{-1}\} = 0 \\ \Downarrow \\ \cancel{\det(B_{mc})}\det\{-\gamma\ \Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon) - \nu I\}\cancel{\det(B_{mc}^{-1})} = 0^{*\|} \\ \Downarrow \\ \det\{-\gamma\ \Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon) - \nu I\} = 0 \end{gathered} \tag{3.23}$$

What remains is purely diagonal, so the eigenvalues are simply each of the diagonal

*$\det(AB) = \det(A)\det(B)$

‖$\det(A^{-1}) = \det(A)^{-1}$

elements of the inner left part:

$$\nu_i(t) = -\gamma \frac{\lambda_{ii}(t)}{\hat{\lambda}_{ii}(t)} \kappa_{ii}(t) \tag{3.24}$$

Since γ, λ_{ii}, $\hat{\lambda}_{ii}$, and κ_{ii} are positive, it is clear that the i^{th} eigenvalue is real and negative when r_i is not zero (and zero when it is). Negative eigenvalues indicate that the error will asymptotically approach zero with speed proportional to the size of the eigenvalue. Therefore, increasing γ and decreasing ϵ will generally result in an increase the speed of the approach.

3.3 Sensitivity Analysis

Section 3.2 introduced the theory and stability proofs for the adaptive control inversion method presented thus far. However, the stability proofs often relied on strict assumptions of perfect measurement and incorporation of a perfectly accurate model of the system. This section will consider various deviations from those assumptions, similar to the analysis performed in [24].

3.3.1 Measurement Noise and Disturbances

For this analysis, let us assume that the real system perfectly matches the model of the system (i.e. $f_c = f_{mc}$ and $B_c = B_{mc}$), except that it is subject to some disturbances to both the state and state derivatives. In addition, let us assume that the measurements of both the state and state derivatives contain some noise. Then the system can be

described by:

$$\begin{aligned}
\dot{\mathbf{x}}_c(t) &= f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t)) + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)\mathbf{d}_{\dot{x}_c}(t) \\
\tilde{\mathbf{x}}(t) &= \mathbf{x}(t) + \mathbf{d}_x(t) + \mathbf{n}_x(t) \\
\tilde{\dot{\mathbf{x}}}_c(t) &= \dot{\mathbf{x}}_c(t) + \mathbf{n}_{\dot{x}_c}(t)
\end{aligned} \tag{3.25}$$

In this case, the measured error, $\tilde{\mathbf{e}}$, will differ from true error since it will contain measurement noise. Substituting the above system descriptions into the definition of measured error, equation 3.6, gives:

$$\begin{aligned}
\tilde{\mathbf{e}}(t) &= [f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t)) + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) + \mathbf{d}_{\dot{x}_c}(t) + \mathbf{n}_{\dot{x}_c}(t)] \\
&\quad - [f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t) + \mathbf{n}_x(t)) + B_{mc}\mathbf{r}(t)] \\
&= B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t) + \\
&\quad \left.\begin{array}{l} [\mathbf{d}_{\dot{x}_c}(t) + \mathbf{n}_{\dot{x}_c}(t) + f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t)) \\ \quad - f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t) + \mathbf{n}_x(t))] \end{array}\right\} \tilde{\mathbf{n}}(t) \\
&= B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t) + \tilde{\mathbf{n}}(t)
\end{aligned} \tag{3.26}$$

where all of the noise and disturbance terms have been grouped into a single term called $\tilde{\mathbf{n}}$.

The above description of measured error can then be substituted into the definition

of the estimate derivative, equation 3.9, and simplified as follows:

$$\begin{aligned}
\dot{\hat{\Lambda}}(t) &= \gamma \, \mathrm{diag}\{B_{mc}^{-1}[B_{mc}(\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\mathbf{r}(t) + \tilde{\mathbf{n}}(t)]\}\hat{\Lambda}(t)G(t,\epsilon) \\
&= \gamma \, \hat{\Lambda}(t)\{\mathrm{diag}[(\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\mathbf{r}(t)] + \mathrm{diag}(B_{mc}^{-1}\tilde{\mathbf{n}}(t))\} \, G(t,\epsilon)^{\dagger\P} \\
&= \gamma \, \hat{\Lambda}(t)\{(\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\mathrm{diag}(\mathbf{r}(t)) + \mathrm{diag}(B_{mc}^{-1}\tilde{\mathbf{n}}(t))\} \, G(t,\epsilon)^{\S} \\
&= \gamma \, \{(\Lambda(t) - \hat{\Lambda}(t)) + \hat{\Lambda}(t)\mathrm{diag}(B_{mc}^{-1}\tilde{\mathbf{n}}(t))R^{-1}(t)\}R(t) \, G(t,\epsilon) \\
&= \gamma \, \{\Lambda(t)\mathrm{K}(t,\epsilon) - \hat{\Lambda}(t)[\mathrm{K}(t,\epsilon) - \mathrm{diag}(B_{mc}^{-1}\tilde{\mathbf{n}}(t))G(t,\epsilon)]\}
\end{aligned} \tag{3.27}$$

Upon inspection, we find that the resulting estimate dynamics are identical to the ideal case, equation 3.14, except that they include an extra term containing $\tilde{\mathbf{n}}$. Here are few observations that can be made from the presence of this extra term:

- $(B_{mc}^{-1}\tilde{\mathbf{n}})_i \cdot g(r_i, \epsilon)$ perturbs $\hat{\lambda}_{ii}$ either toward zero if negative or toward positive infinity if positive

- This perturbation will occur with a "gain" of $\gamma \cdot g(r_i, \epsilon)$

- A sustained $(B_{mc}^{-1}\tilde{\mathbf{n}})_i \cdot g(r_i, \epsilon) > \lambda_{ii} \cdot \kappa_{ii}$ will cause $\hat{\lambda}_{ii}$ to diverge toward infinity at an exponentially increasing rate

Now let us examine the effects of these estimation dynamics on the real error dynamics. Substituting the system description from equations 3.25 into the definition of

real error, equation 3.5, gives:

$$\begin{aligned}
\mathbf{e}(t) &= [f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t)) + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) + \mathbf{d}_{\dot{x}_c}(t)] \\
&\quad - [f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t) + \mathbf{n}_x(t)) + B_{mc}\mathbf{r}(t)] \\
&= B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t) + \\
&\quad \left.\begin{array}{l} [\mathbf{d}_{\dot{x}_c}(t) + f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t)) \\ \quad - f_{mc}(\mathbf{x}(t) + \mathbf{d}_x(t) + \mathbf{n}_x(t))] \end{array}\right\} \mathbf{n}(t) \\
&= B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t) + \mathbf{n}(t)
\end{aligned} \tag{3.28}$$

which is identical to the measured error, but without the influence of noise from the measurement of the state rates.

Taking the derivative of the above equation with respect to time gives:

$$\begin{aligned}
\dot{\mathbf{e}}(t) = B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\dot{\mathbf{r}}(t) \\
+ B_{mc}\dot{\Lambda}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\
- B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\dot{\hat{\Lambda}}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\
+ \dot{vn}(t)
\end{aligned} \tag{3.29}$$

Let us analyze this equation one term at a time as had been done for the ideal case, equation 3.8, in section 3.2.6. Let us again assume that real actuator effectiveness does not change often (i.e. damage or failures are rare) so $\dot{\Lambda}$ is typically zero and therefore the second term in equation 3.29 is typically zero. It was also shown in section 3.2.6 that the third term is stabilizing and will drive the error asymptotically toward zero. The fourth term is the direct effect of the disturbances and noise on the error dynamics and is not affected by the designer's choice of γ and ϵ. While this term will perturb error away from zero, the third term's stabilizing effect will bring the error back toward zero.

In section 3.2.6, it had been argued that the first term in equation 3.8 would quickly go to zero since $\hat{\Lambda}$ was guaranteed to approach Λ. In this case however, equation 3.27 shows that noise and disturbances can perturb $\hat{\Lambda}$ away from Λ, and that those perturbations have a "gain" of $\gamma \cdot g(r_i, \epsilon)$. A large deviation from Λ, especially toward zero, can cause error to become very sensitive to actuator transients.

In order to mitigate the negative effects of noise and disturbances that have been exposed in this section, the "gain", $\gamma \cdot g(r_i, \epsilon)$, should be made as small as possible. However, this exposes a design trade-off. It has been shown in section 3.2 that the speed of approach of the effectiveness estimates to the true values is proportional to $\gamma \cdot \kappa_{ii}$, which is $\gamma \cdot g(r_i, \epsilon) \cdot r_i$. Clearly the designer must carefully choose γ and ϵ to get the fastest estimation possible while protecting against sensitivity to noise and disturbances.

3.3.2 Uncertainty in the State Function

For this analysis, let us assume perfect measurements and that the real system input matrix perfectly matches the model (i.e. $B_c = B_{mc}$), but that the real state function, f_c, differs from the model, f_{mc}. Then the system can be described by:

$$\begin{aligned} \tilde{\dot{\mathbf{x}}}_c(t) &= \dot{\mathbf{x}}_c(t) = f_c(\mathbf{x}(t)) + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\ \tilde{\mathbf{x}}(t) &= \mathbf{x}(t) \end{aligned} \tag{3.30}$$

Substituting the above system descriptions into the definition of measured error,

equation 3.6, gives:

$$\begin{aligned}\tilde{\mathbf{e}}(t) &= [f_c(\mathbf{x}(t)) + B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)] \\ &\quad - [f_{mc}(\mathbf{x}(t)) + B_{mc}\mathbf{r}(t)] \\ &= B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\mathbf{r}(t) + \\ &\quad \underbrace{[f_c(\mathbf{x}(t)) - f_{mc}(\mathbf{x}(t))]}_{\delta_f(\mathbf{x}(t))}\end{aligned} \tag{3.31}$$

where the difference in contribution from the modeled and real state functions has been grouped into a single term called $\delta_f(\mathbf{x}(t))$.

The above description of measured error can then be substituted into the definition of the estimate derivative, equation 3.9, giving:

$$\begin{aligned}\dot{\hat{\Lambda}}(t) = \gamma \{&\Lambda(t)\mathrm{K}(t,\epsilon) - \hat{\Lambda}(t)[\mathrm{K}(t,\epsilon) \\ &- \mathrm{diag}(B_{mc}^{-1}\delta_f(\mathbf{x}(t)))G(t,\epsilon)]\}\end{aligned} \tag{3.32}$$

This result is strikingly similar to that for the previously investigated case of the system in the presence of disturbances and noise. In this case again, modeling error in the state function perturbs $\hat{\Lambda}$ away from Λ with a "gain" of $\gamma \cdot g(r_i, \epsilon)$. The same observations that were made about the influence of noise and disturbances can be made about modeling error in the state function.

As for the error dynamics, since we have assumed perfect measurement, then real error, $\mathbf{e}$, is identical to measured error, $\tilde{\mathbf{e}}$, so the time derivative of $\mathbf{e}$ can be written by

taking the derivative of equation 3.31, giving:

$$\begin{aligned}\dot{\mathbf{e}}(t) = & B_{mc}[\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\dot{\mathbf{r}}(t) \\ & + B_{mc}\dot{\Lambda}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\ & - \gamma\, B_{mc}\Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon)B_{mc}^{-1}\mathbf{e}(t) \\ & + J_{f_c - f_{mc}}(\mathbf{x}(t))\,\dot{\mathbf{x}}_c(t)\end{aligned} \tag{3.33}$$

This result is also similar to that for the previously investigated case of the system in the presence of disturbances and measurement noise. The only difference in this case is that the control designer may have some influence on the final term of the error dynamics by the choice of the model state function. It behooves the control designer to include a model that is as accurate as possible to mitigate the negative effects of state function modeling error.

3.3.3 Uncertainty in the Input Matrix

For this analysis, let us assume perfect measurements and that the real system state function perfectly matches the model (i.e. $f_c = f_{mc}$), but that the real input matrix, B_c, differs from the model, B_{mc}. Then the system can be described by:

$$\begin{aligned}\tilde{\dot{\mathbf{x}}}_c(t) &= \dot{\mathbf{x}}_c(t) = f_{mc}(\mathbf{x}(t)) + B_c\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\ \tilde{\mathbf{x}}(t) &= \mathbf{x}(t)\end{aligned} \tag{3.34}$$

Substituting the above system descriptions into the definition of measured error, equation 3.6, gives:

$$\begin{aligned}\tilde{\mathbf{e}}(t) &= [\cancel{f_{mc}(\mathbf{x}(t))} + B_c\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)] - [\cancel{f_{mc}(\mathbf{x}(t))} + B_{mc}\mathbf{r}(t)] \\ &= [B_c\Lambda(t)\hat{\Lambda}^{-1}(t) - B_{mc}]\mathbf{r}(t)\end{aligned} \tag{3.35}$$

The above description of measured error can then be substituted into the definition of the estimate derivative, equation 3.9, giving:

$$\begin{aligned}\dot{\hat{\Lambda}}(t) &= \gamma\ \mathrm{diag}\{B_{mc}^{-1}[B_c(\Lambda(t)\hat{\Lambda}^{-1}(t) - I)\mathbf{r}(t)]\}\hat{\Lambda}(t)G(t,\epsilon) \\ &= \gamma\ \hat{\Lambda}(t)[\mathrm{diag}(B_{mc}^{-1}B_c\Lambda(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t)) - \mathrm{diag}(\mathbf{r}(t))]\ G(t,\epsilon) \\ &= \gamma\ [\mathrm{diag}(\hat{\Lambda}(t)B_{mc}^{-1}B_c\hat{\Lambda}^{-1}(t)\Lambda(t)r(t))G(t,\epsilon) - \hat{\Lambda}(t)\mathrm{K}(t,\epsilon)] \end{aligned} \tag{3.36}$$

Here it is found that the failure of B_{mc}^{-1} to cancel with B_c prevents much of the simplification that would have been possible otherwise. This equation shows that the estimate dynamics will act as a first-order lag filter of $\mathrm{diag}(\hat{\Lambda}B_{mc}^{-1}B_c\hat{\Lambda}^{-1}\Lambda\mathbf{r})G(t,\epsilon)$, which will likely be different from the true values, Λ, as a function of $\mathbf{r}$.

Ideally, $B_{mc}^{-1}B_c$ should be close to I, but a reasonably sufficient condition for that case has not yet been identified.

As for the error dynamics, since we have assumed perfect measurement, then real error, $\mathbf{e}$, is identical to measured error, $\tilde{\mathbf{e}}$, so the time derivative of $\mathbf{e}$ can be written by taking the derivative of equation 3.35, giving:

$$\begin{aligned}\dot{\mathbf{e}}(t) = B_{mc}[&B_{mc}^{-1}B_c\Lambda(t)\hat{\Lambda}^{-1}(t) - I]\dot{\mathbf{r}}(t) \\ &+ B_c\dot{\Lambda}(t)\hat{\Lambda}^{-1}(t)\mathbf{r}(t) \\ &- \gamma\ B_c\Lambda(t)\hat{\Lambda}^{-1}(t)\mathrm{K}(t,\epsilon)B_{mc}^{-1}\mathbf{e}(t)\end{aligned} \tag{3.37}$$

To ensure stability of these dynamics the eigenvalues of the matrix $-\gamma B_c\Lambda\hat{\Lambda}^{-1}\mathrm{K}(r,\epsilon)B_{mc}^{-1}$ should be negative. However, a reasonably sufficient condition for that case has not yet been identified.

3.4 Extension to a Hexacopter

In the case of a hexacopter, the state rates that are desired to be controlled are: $\dot{\mathbf{x}}_c = \begin{bmatrix} \dot{w} & \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^T$. For the system model, let us tentatively use the linearized dynamics of equation 2.19 from section 2.5. Then $f_{mc} = 0_{4x1}$, $B_{mc} = E$, and $u = \Delta\boldsymbol{\omega}$.

The adaptive controller presented in section 3.2 is premised on the assumption that the system to be controlled is critically actuated, i.e. the number of actuators is equal to the number of state rates that are effected. However, a hexacopter is an over-actuated system, in that its 6 actuators effect only 4 state rates. As a consequence, the matrix B_{mc} is not square and the effectiveness estimator of equation 3.9 cannot be employed.

The crux of the problem is the term $B_{mc}^{-1}\tilde{\mathbf{e}}$ in equation 3.9. Essentially, this term uses the left inverse of the actuation matrix to project state-rate error back into the actuators. It is the inverse of the relationship $\mathbf{e} = B_{mc}\mathbf{e_u}$, where $\mathbf{e}_u$ represents the contribution to error from each of the actuators. However, for the hexacopter, the actuation matrix is a 4×6 matrix and therefore does not have a left inverse. The implication is that there exist a set of infinite possible propeller error combinations that result in the same combination of state-rate error.

3.4.1 Applying Likely Constraints

Some extra information is required to determine which actuators are responsible for the state-rate error. This system of equations has six unknowns with only four relationships.

One approach is to apply likely constraints. For example, one might be tempted to assume that 3 of the 6 propellers have almost identical effectiveness and therefore contribute the same nominal magnitude of error, while the other 3 propellers contribute some unique error. When assuming that propellers #4, #5, and #6 share the same nominal effectiveness, the state-rate error, e, can be re-expressed as a projection from individual propeller contributions to error by equation 3.38.

$$\mathbf{e} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} + b_{15} + b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} + b_{25} + b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} + b_{35} + b_{36} \\ b_{41} & b_{42} & b_{43} & b_{44} + b_{45} + b_{46} \end{bmatrix} \begin{bmatrix} e_{u_1} \\ e_{u_2} \\ e_{u_3} \\ 3e_{u_n} \end{bmatrix} \tag{3.38}$$

If this assumption, that propellers #4, #5, and #6 share the same nominal effectiveness, is correct, then the exact contribution to error from each propeller could be extracted by inverting equation 3.38.

However, while it might be reasonable to assume that 3 of the 6 propellers share almost the same effectiveness, there are 20 possible combinations of 3 nominally effective propellers, but there is no way to determine which candidate combination is the best fit. Furthermore, for 12 of those 20 possible combinations, the matrix in the relation is rank deficient and therefore not invertible.

Another possibility would be to assume that 4 of the 6 propellers have almost identical effectiveness, while the other 2 propellers are uniquely deviant. When assuming that propellers #3, #4, #5, and #6 share the same nominal effectiveness, the state-rate error, e, can be re-expressed as a projection from individual propeller contributions to error by equation 3.39.

$$\mathbf{e} = \begin{bmatrix} b_{11} & b_{12} & b_{13} + b_{14} + b_{15} + b_{16} \\ b_{21} & b_{22} & b_{23} + b_{24} + b_{25} + b_{26} \\ b_{31} & b_{32} & b_{33} + b_{34} + b_{35} + b_{36} \\ b_{41} & b_{42} & b_{43} + b_{44} + b_{45} + b_{46} \end{bmatrix} \begin{bmatrix} e_{u_1} \\ e_{u_2} \\ 4e_{u_n} \end{bmatrix} \tag{3.39}$$

This represents a system of 4 equations with only 3 unknowns. Therefore, the system might be solvable using only 3 of the 4 equations.

There are 15 possible combinations of 4 nominally effective propellers, and some criterion is desired for selecting the combination that is the best fit. Since there are 3

possible combinations of 3 of 4 equations, the variance of the solution of each of those possible combinations of equations may give some clue to the fitness of the solution.

However, in the case of the hexacopter, at least one of the possible combinations of equations for each of the 15 possible combinations of 4 nominally effective propellers results in a singular matrix. For 3 of the propeller combinations, there is only a single solvable combination of equations, and for another 3 propeller combinations, there is no solution.

Another possibility would be to assume that 5 of the 6 propellers have almost identical effectiveness, while only 1 propeller is uniquely deviant. When assuming that propellers #2, #3, #4, #5, and #6 share the same nominal effectiveness, the state rate error $\mathbf{e}$ can be re-expressed as a projection from individual propeller contributions to error by equation 3.40.

$$\mathbf{e} = \begin{bmatrix} b_{11} & b_{12} + b_{13} + b_{14} + b_{15} + b_{16} \\ b_{21} & b_{22} + b_{23} + b_{24} + b_{25} + b_{26} \\ b_{31} & b_{32} + b_{33} + b_{34} + b_{35} + b_{36} \\ b_{41} & b_{42} + b_{43} + b_{44} + b_{45} + b_{46} \end{bmatrix} \begin{bmatrix} e_{u_1} \\ e_{u_2} \\ 5e_{u_n} \end{bmatrix} \tag{3.40}$$

This represents a system of 4 equations with only 2 unknowns. Therefore, the system might be solvable using only 2 of the equations.

There are 6 possible combinations of 5 nominally effective propellers and there are 6 possible combinations of 2 equations. However, at least 3 of the possible combinations of equations for each of the 6 possible combinations of 5 nominally effective propellers results in a singular matrix. For 2 of the propeller combinations, there is only a 2 solvable combinations of equations. Furthermore, due to the symmetry of the hexacopter, opposite combinations of propellers give symmetric results.

A final constraint may be to assume that several of the propellers are exactly as effective as designed, while 1, 2, 3, or 4 of the remaining propellers have some deviance.

Constraining Assumption	Possible Combinations of Propellers	Possible Combinations of Equations (per propeller combination)	Solvable Combinations of Equations (per propeller combination)
3 nominally effective propellers, 3 uniquely deviant	20	1	1 or 0
4 nominally effective propellers, 2 uniquely deviant	15	3	2, 1, or 0
5 nominally effective propellers, 1 uniquely deviant	6	6	3 or 2

Table 3.1: Summary of constraining assumptions.

This is not a likely possibility, so it is ignored.

Table 3.1 summarizes the above analysis:

The consequence of the above analysis is that there is not enough information project sate-rate error back to the propellers using the assumption that 3 or 4 propellers share some nominal effectiveness. The only remaining option then is to assume that 5 propellers are nominally effective while only a single propeller is uniquely deviant. However, as mentioned for this assumption earlier, due to the symmetry of the hexacopter, opposite combinations of propellers give symmetric results. Therefor, an extra assumption has to be made to determine which of the two symmetric results is more likely. We have adopted the assumption that the result with the smallest variance in solutions and the nominal error closest to zero is the most likely.

3.4.2 Estimating Effectiveness in an Over-Actuated System

To extend estimation of actuator effectiveness given in equation 3.9 to the case of an over-actuated system such as a hexacopter, the control matrix inverse is replaced by the above-mentioned likely constrained inverse as in the following equation:

$$\dot{\hat{\Lambda}}(t) = \gamma \, h(\tilde{\mathbf{e}}(t)) \, \hat{\Lambda}(t) \, \mathrm{diag}[g(\mathbf{r}(t), \epsilon)] \tag{3.41}$$

where $h(\mathbf{e})$ is the likely projection of error from the four controllable states to the 6 propellers, which is described in the next section. If $h(\mathbf{e})$ can be properly formulated and Λ is stationary, then the error given by equation 3.6 approaches zero asymptotically with the time constant γ^{-1}.

$h(\mathbf{e})$ is constructed by assuming that error is contributed in a nominal way by 5 of the 6 propellers and uniquely by the remaining propeller. The combination of propellers with the smallest variance of solution, and the smallest nominal contribution, is assumed to be the correct projection.

3.4.3 Reconfiguration upon Failure Detection

If a propeller has become so ineffective that it can no longer be relied upon to contribute to the control task, it must be abandoned in favor of the remaining effective actuators. This implies that the controller must be reconfigured.

To that end, the control mixing inversion performed in 2.32 in section 2.6 is reformulated as follows:

$$u = (E\hat{\Lambda})^{-1}_{\text{right}} \begin{bmatrix} \dot{w}_{cmd} \\ \dot{p}_{cmd} \\ \dot{q}_{cmd} \\ \dot{r}_{cmd} \end{bmatrix} \tag{3.42}$$

where the subscript "right" indicates the Moore-Penrose pseudoinverse. The resulting propeller commands are redistributed to rely less upon less effective propellers.

3.4.4 Controller Limitations

Section 3.2 has already shown that this adaptive scheme is sensitive to measurement noise with the inverse of the adaptive gain. In addition, it was shown that the dynamics may become unstable if the input matrix is not well formed. The extension of this adaptive approach to the hexacopter comes with several additional potential pitfalls.

First, since it is assumed that 5 of the 6 actuators share some nominal effectiveness, it is not possible to account for deviations of multiple actuators at the same time. Second, there may be situations where the projection operator assigns error to the wrong actuator.

Several of the limitations are explored experimental using simulation. The results are presented in chapter 6.

Chapter 4

Position and Attitude from Computer Vision

Of all sensors available for robotic applications, the sensor with the highest information density to cost ratio is probably a digital camera. Furthermore, the size and weight of a digital camera is mostly determined by the size and weight of the lens, so the weight and performance of the camera can be balanced to best fit the application.

The process of extracting information from a stream of digital video is referred to as *computer vision*. The downside to computer vision is that it can be quite computationally intensive, leading to extra cost, weight, and/or electrical draw.

This chapter discusses a particular approach to position sensing using computer vision. This approach is made while considering the constraint that the system must operate on board a small hexacopter.

4.1 Simultaneous Localization and Mapping

The use of a computer vision for positioning can be generalized into a problem of matching features in the camera's view to features of known location that have been stored in

a database, then using that information to compute the pose (location and orientation) of the camera. This process is referred to as *localization*, and the database of features and their corresponding locations is known as the *map*.

However, in many cases, there is no prior knowledge about the environment that the vehicle is operating in, and so there is no map. In such cases, a map can be built. The position of a newly discovered feature can be computed if the position of the camera is known. That feature and its corresponding location can then added to the database. This process is called *mapping*.

The process of determining the camera's location in a map while building that map is called *simultaneous localization and mapping (SLAM)*. However, SLAM has a starting problem since localization depends on the map, and building the map depends on localization. Therefore, some special starting routine must be used in order to perform SLAM.

4.2 Parallel Tracking and Mapping

Parallel tracking and mapping (PTAM) is a SLAM algorithm published 2007 by George Klein and David Murray of the University of Oxford [25]. It was intended to track a hand-held camera for augmented reality, it has found a following in the application to SLAM for robotic vehicles due to its ability to run real-time on relatively light computing packages, including mobile devices.[26]

The PTAM source code was made open source, another reason for its popularity. Several modifications have been made to adapt PTAM to the task of controlling the position of a hexacopter.

The ability of PTAM to run real-time is owed mainly to its innovation of splitting localization and mapping into separate tasks that run in separate CPU threads. This means that the real-time constraint can be enforced on the localization task only while

the mapping task is free to run at its own pace.

4.2.1 Prerequisites

Camera Model

For an ideal camera, light would be collected from the scene in a way such that it travels in a perfectly straight line from the source to the image sensor thorough an infinitesimally small focal point. In reality however, lenses distort the incoming light to a greater or lesser degree depending on many factors. PTAM uses a fairly simple model of a camera that accounts for the focal length and radial distortion of the lens. This model is used to convert real images to an image on an ideal image plane, and vice versa.

An automatic camera calibration program is included with PTAM. This program computes the optimal camera model parameters based on a series of images of a checkerboard.

Feature Detector

As mentioned in section 4.1, computer vision for positioning can be generalized into a problem of matching features in the camera's view to features of known location. The term "feature" means something salient (easily noticeable or distinctive) in an image. Many algorithms exist to identify different kinds of features in an image. PTAM employs the features from accelerated segment test (FAST) feature detector of [34].

FAST is an algorithm that examines an image patch for some heuristics that are representative of a corner. It examines pixel values in discrete positions along a path or a *mask* around the candidate corner. Figure 4.1 shows an example mask, the Bresenham circle. A corner is registered if there is some length of consecutive pixels along the circle that are all lighter or darker than the candidate (center).

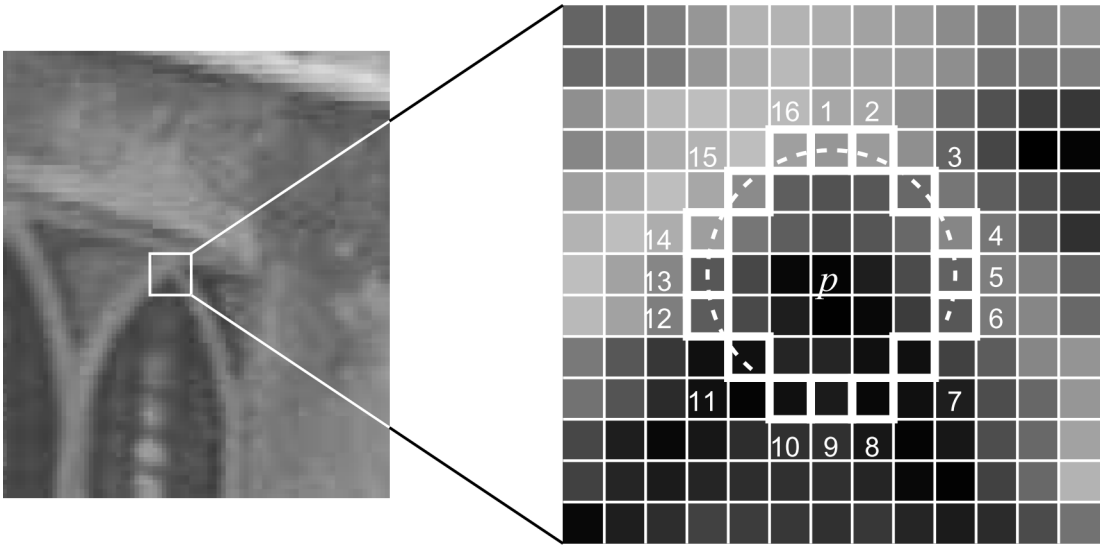


Figure 4.1: The 16 pixel Bresenham circle mask used in FAST superimposed over the examined image (Image from E. Rosten and T. Drummond [34])

Feature Quality Evaluator

It was found that, taken alone, the FAST algorithm does not produce features that are robust and detectable between frames. A feature quality evaluator was added to remedy this issue. PTAM employs the Shi-Tomasi interest operator from [40].

The Shi-Tomasi score is calculated from a window around a feature point in the image. Some characteristics of the feature point can be inferred from the eigenvalues of the structure tensor (a second-moment matrix of the window region). Two small eigenvalues means that the image patch has almost constant intensity. One large and one small value indicate that the image patch depicts something which has a shape only unique in one direction such as a line. Two large eigenvalues means a corner, salt-and-pepper texture or some other texture that can, in general, be tracked reliably. Since the last case is desirable, the Shi-Tomasi score is taken as the minimum eigenvalue.

4.2.2 The Point Cloud, Keyframes, and Pyramids

PTAM's map can be thought of as a database of locations of previously detected salient features (referred to as the *point cloud*), plus a set of *keyframes*. Each point in the point cloud is also stored with a reference to its pixel location in single a keyframe (typically the first keyframe it was observed in).

Keyframes

A keyframe is a single frame of video that is stored when the camera view begins to move to a part of the world frame that has not yet been well explored. The keyframe is stored along with the camera pose (position and orientation) at the time of capture. The keyframe is then compared to the nearest previously captured keyframe, a set of corresponding features is found and the locations of those features in the world frame are triangulated. For features that had not been viewed in the previous keyframe, their locations added to the point cloud with a reference to the pixel of the new keyframe where it was observed.

Greyscale Pyramid

In order to accommodate various scales (i.e. the camera is close to a feature or far from a feature), both the incoming images and the keyframes are processed into a pyramid of greyscale images at 4 levels of decreasing resolution. For example, for an incoming image with resolution 800x600, the bottom level (level 0) would be the full resolution image converted to 8 bits-per-pixel greyscale. The next level (level 1) would be the level 0 image down-sampled by 2, yielding a 400x300 greyscale image. Subsequent levels follow the same pattern so that level 2 would have a resolution of 200x150 and level 3 would have resolution 100x75.

For each point in the point cloud, the reference to a keyframe includes the pyramid level in which it can be found.

Using this pyramid scheme allows features from images of different scale to be matched. In addition, features at the higher levels are more resilient to motion blur, so they are useful even for images of the same scale.

4.2.3 Tracking (localization)

The term *tracking* in the acronym PTAM corresponds to the term *localization* in the acronym SLAM. As such, the tracking part of the PTAM algorithm is responsible for determining the pose of the camera in the world frame. P The tracking part can be described in the following steps:

1. The current pose of the camera is estimated from the previous pose and velocity.
2. The point cloud is reduced to a potentially visible set.
3. A wide search is performed for 50 of the points from the potentially visible set.
4. The camera pose is refined from the results of the above search.
5. A narrow search is performed for 1000 of the points in the potentially visible set given the refined pose estimate.
6. The camera pose estimate is further refined from the results of the above search.

Motion Model

PTAM uses a simple motion model for estimating the pose for a newly acquired image. In this model, velocity is calculated by differentiating the previous two poses. The pose of the camera is then projected forward by integrating the velocity plus a constant deceleration factor. Therefore, it is assumed that the camera is always decelerating relative to the world frame.

Potentially Visible Set

To reduce the point cloud to the potentially visible set of points, the points in the point cloud are first projected into the image plane using the calibrated camera projection with the current estimated camera pose. Points that lie outside the bounds of the image after projection are eliminated.

Next, the unit normal vectors of the keyframes that correspond to each of the remaining points are projected into the image plane. Points where the associated keyframe is oriented away from the camera, or nearly perpendicular to the camera, are discarded.

Finally, for the remaining points, the corresponding keyframe features is checked to see of its scale is similar to any of the pyramid levels of the current image. If not it is discarded.

Pose Estimate Refinement

The goal of this step is to refine the initial estimate of the camera's pose by aligning salient featured of the current image to some already-cataloged features (i.e. the map points in the point cloud). To this end, salient features are first identified in each level of the image's greyscale pyramid using the FAST corner detector for comparison to the potentially visible set of map points.

To perform the search potentially visible map points in the current image, a patch of the keyframe corresponding to the map point being searched for is extracted and warped to the current image plane using an affine transformation. The transformed patch is then compared to each feature of the current image that has been identified by the FAST corner detector, within a radius of where the feature is expected to be found given the current estimate of the camera's pose. The comparison is scored using a zero-mean sum of square differences, and the feature of the current image with the lowest score is selected as a match.

After a map point has been matched to a feature in the current image, the projection

error (the difference between where it appears in the image and where it is expected to appear given the current pose estimate) can be computed. The pose estimate is then optimized to minimize the mean Tukey biweight objective score.[20]

As a first attempt at aligning to the point cloud, 50 map points that should appear at the top level of the current image's greyscale pyramid are searched for. If there are not enough points are the highest level, or not enough are found in the image, then the next level is also searched.

After completion of the above first attempt, the pose is further refined by repeating the search for a much higher number of map points (around 1000) with a much smaller search radius.

Tracking Quality

The tracking quality is evaluated according to the percentage of points from the potentially visible set that were successfully matched in the latest image. Two thresholds are given for this percentage. Below the higher threshold, tracking is considered poor. Below the lower threshold, tracking is considered lost.

In the case of poor tacking, tracking will continue but new keyframes will not be added since the probability of corrupting the map with a misaligned keyframe is high. When tracking is lost, there is no confidence in the current pose estimate, and a potentially visible set cannot be constructed. In such a case, the recovery method of [47] is used until a pose estimate is is found, after which tracking will continue as normal.

4.2.4 Mapping

The previous section discussed camera localization, i.e. how to determine the location of the camera relative to an already constructed map. This section discusses how the map is built.

Initialization

As mentioned in section 4.1, SLAM has a starting problem where a map is required to localize the camera, but the position of the camera must be known in order to build the map. PTAM addresses this problem with the use of a user-driven initialization routine.

The user initiates the routine with a key press, at which point the current image is stored as the first keyframe. From there, 1000 of the most salient features at the lowest pyramid level are tracked through smooth motion between subsequent frames. The user ends the initialization with another key press, at which point the current frame is recorded as the second keyframe in the database, and the correspondence of the features that have been tracked since the first keyframe are used to with the 5-point stereo algorithm in [42] and random sample consensus (RANSAC) to compute the relative spatial locations of the features, which then forms the initial point cloud.

For this initialization to be successful, many of the initially identifies salient features must remain tracked through every frame of the initialization routine. If a feature is every lost from motion blur or because it has left the file of view, it cannot be recovered. This means that the initialization must be done with a relatively short, smooth motion.

Also, the triangulation of the relative spatial locations of the points that were successfully tracked relies on the assumption that there is a stereo baseline. Therefore, initialization with pure rotation will not produce any disparity and will not give 3-dimensional information, so some translation must be included.

Adding Keyframes

In section 4.2.2 it was mentioned that a new keyframe is stored when the camera view begins to move to a part of the world frame that has not yet been well explored. To be more precise, the conditions for adding a new keyframe are:

1. Tracking quality is good.

2. At least 20 frames have arrived since the last added keyframe.

3. The nearest keyframe is farther than a certain threshold.

These criteria are monitored by the tracker. When met, the tracker sends the following to the mapper:

- The current image's greyscale pyramid
- The camera pose when the image was obtained
- A list of FAST corners
- A list of map points that have been matched to some FAST corner

Upon reception of this information, the mapper will then abort its current task and begin the process of appending the new keyframe to the map.

As a first step, the mapper revisits the existing map points that should be visible in the image, but were not included in the list from the tracker. It is possible for the mapper to perform a more thorough search since it does not have the tracker's real-time constraint. These points are re-projected onto the new image and added to the list of they are observed in the image.

Next, the set of image features is reduced to only the most salient features in each pyramid level. This is done by applying a non-maximal suppression and threshold based on each feature's Shi-Tomasi score, as described in section 4.2.1. The points that remain after this reduction are then considered for addition to the map.

Taken alone, the candidate points have no depth information. However, if the candidate points can be identified in another keyframe, then the disparity between the two occurrences gives the depth information. In particular, pixel patches around corner points that lie on the epipolar line are compared to the candidate point using zero-mean sum squared difference. This potentially means searching along long lines in many

keyframes to match a single candidate. To make the search more tractable, the following simplifications are used:

- Only the nearest keyframe is checked for correspondence.
- No template warping is preformed.
- Only equal pyramid levels are searched.
- The epipolar line is limited to a segment based on the range of depth where the feature is likely to occur

If a match has been found, the candidate point is triangulated. If that triangulation yields a result that is not close to a point that is already in the map, then the candidate point is added to the map.

Bundle Adjustment

As keyframes are added to the map, small errors in the pose of the keyframes can accumulate and make the map less and less accurate. To combat this, PTAM uses a method called *bundle adjustment* to realign the pose of the keyframes to minimize the projection error of the map points in each of the keyframes. PTAM uses the method presented in appendix 6.6 of [16], but augments it with the Tukey M-estimator.

The complexity of this problem increases with at least the cube of the number of keyframes. Furthermore, this process will be interrupted whenever a request for the addition of a new keyframe arrives.

To make this more tractable, the mapper splits the task into a regional and global optimization. For the local optimization, bundle adjustment is performed on the latest keyframe plus the 4 nearest keyframes. The mapper only proceeds onto global bundle adjustment after the local bundle adjustment has converged.

4.3 Application to Hexacopter Navigation

As mentioned in section 4.2, PTAM was originally developed to track a hand-held camera for augmented reality. However, its ability to run real-time on relatively light computing packages makes it desirable for use onboard a small aerial vehicle such as a hexacopter. In such a case, the camera is fixed to the body so that there is a simple relationship between the pose of the camera and the position and orientation of the vehicle.

4.3.1 PTAM Limitations

While PTAM is very attractive due to its computational efficiency, it does have several limitations. Some of the limitations discussed below are inherent in the PTAM algorithm, others are a consequence of deviating from the intended purpose of tracking a hand-held camera in a small space comes with some penalties.

Map Scale and Orientation

Perhaps the single largest hurdle to using PTAM (or any monocular SLAM algorithm) for navigation is the ambiguity of the map scale and orientation with respect to the earth frame.

PTAM uses the stereo disparity between two images of an unknown environment to set the initial map. Because the environment is unknown, no information can gained about the scale of the scene or the orientation of the points with respect to the earth frame. The scale is set by arbitrarily setting the stereo baseline to 10 cm, and the orientation is chosen such that the z world-axis is normal to the dominant plane in initial tracked features (i.e. it is assumed that the scene is a roughly horizontal plane), and the y world-axis is parallel to the y camera-axis at the end of the initialization. Therefore, extra information is required to align the scale and orientation of PTAM's world frame to the earth frame.

Map Size Limitation

The complexity of optimizing the map via bundle adjustment grows with at least the cube of the number of keyframes. At some point, the number of keyframes becomes too large for the bundle adjustment to be performed in real-time.

Drift and Loop Closure

In reality, when a new keyframe is added to the map, the pose estimate of that keyframe has some error. That error is propagated to the pose estimate of future keyframes, which will introduce further error. The effect is that the map becomes less accurate with distance from the map origin. For example, if the hexacopter flew in a large circle, the end of the map would most likely not coincide with the map at the start of the circle.

Camera Model

Similar to the above problem, a small difference between the true and modeled camera distortion leads to a distortion of the triangulated map points. In such cases, perfectly flat translation results in a curved map.

Dependence on Fixed Contrasting Features

The use of the FAST corner detector means that the scene must contain features that have contrast. A uniform surface, such as a finished concrete slab, offers no contrasting features to track.

Scenes with moving features will also confuse the tracker. For example, rippling water or waving grains might offer features, but those features are not fixed to the earth frame. In addition, the vehicle's own shadow may confuse the tracker.

Finally, the absence of light would inhibit the ability of the camera to gather any information from the scene. (This is essentially the same as the problem of the uniform surface)

Saliency Transitions

PTAM uses a fixed threshold for the Shi-Tomasi score to determine whether or not a feature should be included in the map. However, the saliency of features in the scene can vary widely between environments. For example, a rocky path contains many high-contrast features, while a concrete sidewalk has few. Setting the threshold for a high-saliency environment will lead to almost features will be rejected if the camera moves to a low-saliency environment. Alternatively, setting the threshold for a low-saliency environment means that the mapper will be overwhelmed with features if the camera transitions to a high-saliency environment.

Manual Initialization

PTAM's initialization routine requires user interaction. This is mainly because initialization requires smooth translation, so the user interaction is a way of notifying PTAM that a suitable movement is about to be performed. Also, success of the initialization must be verified. The initialization often either aborts due to lack of features or stereo baseline, or results in poor tracking. Either case is easily identifiable by the user who must then re-initiate the initialization.

4.3.2 Modifications

This section contains a brief summary some of the modifications that were made to PTAM to address some of the limitations for the purposes of hexacopter navigation. Daniel Henell of the University of Tokyo's Ishikawa-Oku Lab was the chief architect of these changes, which are described in detail in his thesis, [17].

Alignment to the Earth Frame

As mentioned in section 4.3.1, it is not possible to determine the scale and orientation of an unknown environment with a monocular camera. However, if an object of known

size and orientation can be identified in the environment, then a transformation can be constructed to align the map frame to the earth frame.

The object settled on for this purpose is a *fiducial marker*, a 2-dimensional symbol that has been designed to be easily identifiable by computer vision. The open-source library ARToolKit [27] is used for the specific implementation, and the particular fiducial marker used is shown in figure 4.2.

ARToolKit was used to find the image coordinates of the marker's corners. Virtual rays are then constructed from the camera center through each of the marker corners. It is assumed that the plane of the marker coincides with PTAM's ground plane, so the distance, in the map frame, between the intersection of the rays and the ground plane is compared to the known size of the fiducial marker to construct a scale factor.

The assumption that the marker plane coincides with the ground plane means that z direction already matches in both the map frame and the earth frame. Then the direction of the fiducial sides is used to determine a simple rotation transformation about the z axis.

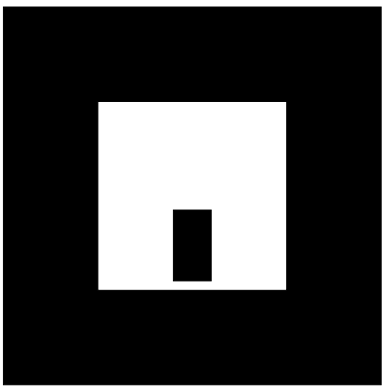


Figure 4.2: The ARToolKit marker employed for aligning the scale and orientation of the map frame to the earth frame.

Architecture Restructuring

To improve performance and take advantage multi-thread processors, the program was divided among 3 more threads, 5 in total. It was found that tracking performance could be increased considerably when rendering was disabled, so the code was refactored so that tracking could be run in a separate thread and therefore isolated from the computational requirements of rendering and other system functions.

Next, it was found that the video capture uses a buffer to ensure smooth video playback. When a video frame is requested, the oldest frame of the buffer is delivered. This can cause delayed measurements. To combat this, video capture was placed in another thread that constantly grabs the latest frame.

Finally, a 5th thread was created to handle all tasks related to control of and communication with the hexacopter. This is discussed in section 4.4.

4.4 Extension for Control of a Hexacopter

Let us assume that a hexacopter has a minimal sensor suite consisting of a set of 3-axis gyros and 3-axis accelerometers. Combined with an observer, this can give good estimates of the linear and angular accelerations and angular rates. With some extra assumptions, one might also be able to extract some reasonable estimates of linear speed and attitude. This information allows for the design of a quite well-behaved attitude and vertical acceleration controller with excellent disturbance rejection. Position control, however, is out of the question since drift in the speed and attitude estimates would wreak havoc on any position estimates.

Appending the sensor suite with computer vision gives drift free measurements of position and attitude allowing very robust estimates of the full state.[1] As an added

[1] In section 4.3.1 it was mentioned that PTAM's estimates drift as the camera moves farther from the origin. This drift is a function of position whereas the drift from integrating the accelerometers and gyros is a function of time. If we assume bounds on the distance that the hexacopter will travel from the map origin, then we can infer that the drift will also be bounded.

benefit, the fact that computer vision is computationally intensive means that a relatively powerful processor must be employed, which can easily handle some processing that might be burdensome on a microcontroller. This benefit is exploited by performing extended Kalman filtering on the vision computer and transmitting the results back to the microcontroller.

Chapter 5

The Hexamac Testbed

In this chapter, the University of Tokyo's Hexamac testbed is introduced. The Hexamac is a modified Mikrokopter Hexa XL with custom control software, paired with a custom computer-vision package hosted on a stripped-down Mac mini computer.

5.1 MK Hexa XL Platform

The Mikrokopter Hexa XL is a relatively high-lift multirotor developed by the German company HiSystems GmbH. Table 5.1 shows some relevant specifications of the Hexa XL.

Although the Hexa XL is capable of producing 64 N of thrust, which could theoretically allow for hovering with a payload of 4 kg, maneuvering with a payload of that weight would not be possible since the actuators would be fully saturated (the motors

Mikrokopter Hexa XL	
Empty weight (minus battery)	1.44 kg
Operational weight	2.15 kg
Maximum thrust (at hover)	64 N
Flight time (no payload)	30 min

Table 5.1: Some relevant specifications of the Mikrokopter Hexa XL

are already operating at maximum speed). In practice, the payload should be well below the lifting capacity to ensure maneuverability.[1]

5.1.1 Actuation

Actuation is achieved via 13-inch (0.33 m) propellers driven by brushless DC electric motor (BLDC) motors. The motors employ 12 stator coils and a rotor with 14 magnets. The motors do not include Hall effect sensors or a rotary encoder. Because the controller must direct the rotor rotation, the controller requires some means of determining the rotor's orientation relative to the stator coils. In the absence of Hall effect sensors or a rotary encoder to directly measure the rotor's position, the back EMF in the undriven coils is measured to infer the rotor position. As a consequence, the controller cannot determine the orientation of the rotor when not rotating. Initiating rotation is typically accomplished by beginning rotation from an arbitrary phase, and then skipping to the correct phase if it is found to be wrong. This can cause the motor to run briefly backwards, adding even more complexity to the startup sequence.

The motor controllers are capable of delivering up to 35 Amps of continuous current to each of the motors. Each of the motor controllers employs an ATMEGA168 microcontroller. However, the embedded software is not open and therefore it is not modifiable.

With the motors off, the FlightCtrl, motor controllers, radio control (R/C) receiver, and light emitting diodes (LEDs) draw approximately 0.28 A at 16 V. The battery employed is a 4 cell lithium ion polymer (LiPO) battery with a capacity of 6.6 A·h and the ability to deliver 132 A of continuous current. With this battery, the Hexa XL can fly about 30 minutes without a payload. This implies that the motors consume the vast majority of the battery's charge.

[1] The Mikrokopter website recommends limiting the payload to 1.0 kg.[14]

5.1.2 Structure

The structure of the Mikrokopter Hexa XL is mainly composed of 6 aluminum arms and a pair of carbon-fiber connector plates.

The arms have a hollow square cross section with wall width of 1 millimeter. Each arm is 0.36 meters in length. A pair of fairly large holes is included in the arms to allow the passage of motor and LED wires. Experience showed that deformation occasionally occurred in the area around the holes closest to the center of gravity during particularly "hard landings," which could necessitate the time consuming process of replacing the arms (complicated by the fact that the motor wires are passing through the center).

The carbon-fiber plates act as a means of binding the 6 arms together so that they are all coplanar with their endpoints lying at the vertices of a hexagon. The plates are bound to the arms with bolts and locking nuts.

Other parts of the Hexa XL can be viewed as attachments to the main structure. These include the battery cage, which is hung under the structure, the 3 legs, which are attached three of the arms, and the controllers, which sit above the center of gravity (CG) of the main structure.

The Hexa XL structure was modified slightly. First, several connectors were added so that the arms could be separated, making it possible to disassemble the Hexa XL for easy transport. Second, a pair of mounting brackets was added below the battery cage to allow the addition of a custom payload (i.e. the vision package described in 5.4).

5.2 MK FlightCtrl v2.1

The Hexa XL comes with a compact control electronics circuit board called the FligtCtrl v2.1. This board enables the manual control of the hexacopter's attitude, thrust, and height via R/C. It also offers some communication ports for the addition of supplemental electronics. One such option is an add-on board called NaviCtrl, which adds the capa-

bility to perform automatic position heading control. For the research presented here, a custom electronics package was connected to the FlightCtrl in lieu of the NaviCtrl. This package is discussed in section 5.4.

For the purposes of this research, it was desired to use the hardware components of the FlightCtrl board, but embedded software was nearly completely replaced with custom software that is optimized for research purposes. Therefore, while the FlightCtrl is an off-the-shelf product, some detail of its hardware is presented here as a basis for explaining how the custom embedded software was developed.

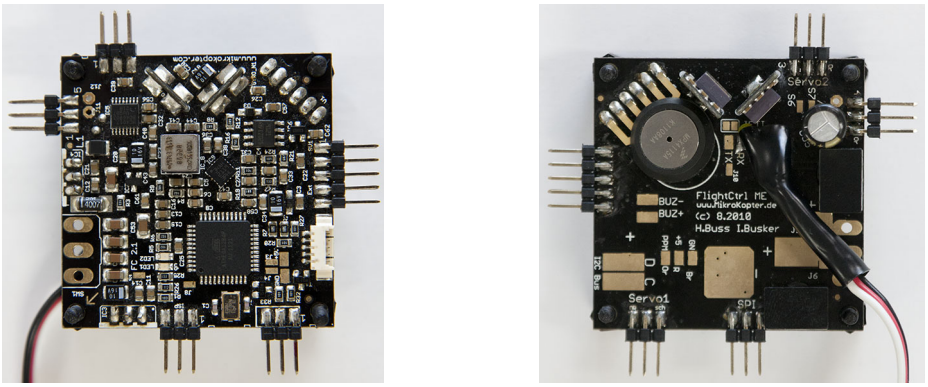
Table 5.2 lists the main components of the FlightCtrl V2.1, which can be seen in figure 5.1.

Quantity	**Description**	**Part**
2	Voltage Regulators	R-785.0-1.0
1	Microcontroller	ATMega 1284p
1	3-axis Accelerometer	LIS344ALH
3	Gyros	ADXRS620
1	Pressure Sensor	MPX4115
2	LED Drivers	TS912D
1	Servo Driver	74HC4017PW
4	Communications Ports	headers or pads

Table 5.2: The main component of the FlightCtrl v2.1 board.

In the most basic sense, the NaviCtrl performs the following functions:

- Read the sensors
- Receive commands from the radio control or other external device
- Compute propeller speed commands from the R/C commands plus feedback from the sensors
- Send propeller speed commands to the motor controllers
- Manage the modes of operation (e.g. engine start/stop, calibration, etc.)



(a) top (b) bottom

Figure 5.1: The components of the FlightCtrl board.

- Monitor for problems
- Provide physical status indicators

5.2.1 Microcontroller

The brains of the FlightCtrl is the ATMega 1284p microcontroller. This microcontroller includes all of the following components on a single chip [4]:

- A 20 MHz, 8-bit CPU
- 128 Kb of flash memory (program memory)
- 4 Kb of electronically erasable programmable read-only memory
- 16 Kb of random access memory
- 2 8-bit timers
- 2 16-bit timers
- An 8-channel 10-bit analog-digital converter
- A two-wire-interface device

- 2 universal synchronous and asynchronous serial receivers and transmitters
- A serial peripheral interface device
- A watchdog timer
- 32 I/O pins

The majority of the above-mentioned features are indeed employed in the control of the hexacopter. The central processing unit (CPU) is extremely light. Because the processor can only perform 8-bit mathematics, floating point operations require algorithms that are prohibitively computationally expensive and should be avoided wherever possible. This, combined with the slow clock speed, means that advanced processing cannot be achieved on this chip given the real-time control requirements.

5.2.2 Sensors

All of the sensors on the FlightCtrl (accelerometers, gyros, and pressure sensor) output an analog voltage in proportion to the quantity they are sensing. Those outputs are read via the microcontroller's analog-to-digital converter (ADC). For the accelerometers and gyros, a middle voltage corresponds to a reading of zero. This neutral voltage varies with each sensor and must be calibrated. This is done by reading the voltage output from each senor when the hexacopter is stationary on level ground. Those readings are then used to debias future readings.

Accelerometers

The accelerometers on the FlightCtrl measure proper acceleration (acceleration relative to free fall) along the x, y, and z body-axes independently. Because they measure relative to free fall, the accelerometers register an acceleration of 1 g straight up due to gravity when stationary in the earth frame. This is useful since the attitude of the hexacopter

can be accurately deduced from the direction of the gravity vector prior to takeoff. This attitude is used at the starting point for the future in-air attitude estimation.

The accelerometers are capable of measuring up to 2 g with an output relation of 0.6 V/g.[43] The microcontroller's ADC has a resolution of approximately 2.93 mV (1024 bits over 3 V) [4] which leads to a resolution of 0.00488 g/step.

It has been found that the magnitude of the z body-axis measurement is corrupted by vibrations in flight and therefore is not usable.[2]

Gyros

The gyros on the FlightCtrl measure the rotational rate along the x, y, and z body-axes independently. They are capable of measuring angular rates up to 300 deg/s (5.2 rad/s) with an output relation of 6 mV/(deg/s).[10] The microcontroller's ADC has a resolution of approximately 2.93 mV (1024 bits over 3 V) [4] which leads to a resolution of 0.814 (deg/s)/step (0.0142 (rad/s)/step).

Pressure (Altitude) Sensor

The FlightCtrl uses a pressure sensor to determine altitude by assuming the following relation between change in barometric pressure and change in altitude: pressure(kPa) = 100 - 0.012 * altitude(m).[13] The voltage output from the pressure sensor follows the following relation: Vout = 0.045 * pressure(kPa) - 0.48.[39] This leads to the relation: Vout = 4.5 - 0.00054 * altitude(m). So, for a change in altitude of 1 m, the voltage output from the sensor only changes by 0.00054V.

However, the microprocessor's ADC has a measurement range of 3 Volts in steps of approximately 2.93 mV (1024 bits).[4] This implies a resolution worse than 5 meters per ADC step. To increase the resolution, a non-inverting operational amplifier is inserted

[2]This may also be true for the x and y body-axis measurements, but is difficult to confirm since it requires the ability to apply a force of known magnitude along one of those axes while in flight to compare to the resulting measurement.

between the pressure sensor and the ADC. This op-amp has a gain of 30.12, which leads to a resolution of 0.18 meters per step.

By increasing the resolution, the measurable range becomes much smaller. At 0.18 m resolution, the 10-bit ADC can only capture a range of 180 meters. In order to operate the Mikrokopter at locations of various elevations (e.g. 5 meters in Odaiba vs. 600 meters at Mt. Takao), and to compensate for meteorological variations in barometric pressure, the measurable pressure range must be pre-shifted so that current pressure lies within the measurable range. To accomplish this, the operational amplifier is wired such the that its output can be biased via a pair of additional voltages. One of the voltages provides coarse adjustment and the other fine adjustment. The voltages are controlled via a pair of PWM signals generated from the microcontroller.

Battery Voltage

For a LiPO battery, voltage decreases with both increased current draw and decreased capacity. If the voltage is ever allowed to drop below approximately 3.0 V per cell (used in a series combination), the battery will subsequently no longer accept a full charge and may experience problems holding voltage under load.[46] To prevent this, the remaining ADC channel is used to monitor battery voltage so that warnings and/or limits can be implemented.

5.2.3 Communication

The FlightCtrl board makes it possible to communicate with external devices via several communication ports. Table 5.3 lists the communication ports and their uses.

Motor Controllers

Communication with the 6 motor controllers is accomplished through use of the two wire interface (TWI) communication protocol. Using this protocol, many components

Port	Wires	Typical Communications
TWI	2	Motor Controllers (bi-directional)
SPI	3	NaviCtrl (bi-directional)
USART0	2	NiviCtrl or External Computer (bi-directional)
USART1	2	Serial R/C Receivers (bi-directional)
Timer1	1	PPM R/C Receivers (RX only)

Table 5.3: The FlightCtrl v2.1 communication ports and the devices typically connected to those ports.

can communicate with each other over a single pair of wires, so it is well suited to the case of the FlightCtrl communicating with all of the motor controllers. This is achieved by beginning each transmission with the address of the destination device.

Typically, the FlightCtrl will only send a speed command to each of the motor controllers, and in response will receive information about the present current draw, board temperature, and maximum allowable command from each of the motors. It is also possible to get and set each of the motor controller parameters, which include limits on operating temperature, current draw, and maximum command.

Radio Control

In order to accommodate a variety of brands of radio controllers, the FlightCtrl includes a port to decode radio control receivers that output servo commands in a serial protocol (e.g. Futaba S.Bus, Jeti, JR), and a port to decode receivers that output servo commands using pulse-poisition modulation (PPM). In addition, with receiver models, it is possible to return telemetry data to the radio controller. This is accomplished through serial output.

5.2.4 Other Features

Physical Indicators

The FlightCtrl board includes one port for driving a buzzer and two ports for driving LEDs. These outputs are useful for giving physical indications of FlightCtrl status and warnings in the form of beeps, flashing lights, color changes, etc.

Servo Control

The FlightCtrl board also includes circuitry that can control up to 7 external servos. This is made possible via a timing chip that decodes PPM commands from the microcontroller into separate pulse-width modulation (PWM) commands for servos. A separate power regulator is also provided so that an overdraw from the servos won't affect the critical FlightCtrl functionality.

These servo outputs are most often used to drive a stabilized camera mount. Servos are not used in this research.

5.3 Custom Embedded Software

The stock embedded software included with FlightCntrl is mostly open source.[3] However, it was found that much of this code is poorly laid out, inefficient, not well documented, and not readily modifiable for the purposes of controls research. For these reasons, it was decided to almost completely rewrite the embedded software. This section describes the methodology and architecture of the custom embedded software.

The software can be roughly separated into three parts:

1. Initialization

[3] A pre-compiled binary is included with the FlightCtrl source code. This binary must be linked with the rest of the source code. The functionality of this binary has not been identified, but the embedded software will not execute without it.

2. The main program

3. Interrupting functions

After initialization, the main program is repeated at 128 Hz. The main program is constantly interrupted by high priority, fast-running, functions. Even so, the computationally intensive main program always finishes before its 128 Hz frame expires. Each of these 3 parts will be described in detail in the following subsections.

5.3.1 The Main Program

The main program is generally responsible for the computationally intensive tasks. It processes incoming information, manages the various modes of operation, executes the control law, and prepares data for output. It is executed at 128 Hz and takes several milliseconds to process. It is the lowest priority task, so any interrupting function will (briefly) take the focus of the CPU. The following subsections describe the sub-functions of the main program in order of execution.

Main 1: Process Serial Inputs

If any message has been received over the serial connection, it will have been placed into a buffer and needs to be processed. This function first decodes the message and then makes the appropriate response according to the address and command identifier in the message header.

For example, a command identifier “b” is a request for the debug output stream. This particular message does not carry any further data, but will be responded to by initiating the debug output stream.

As another example, a reception addressed to the FlightCtrl with a command identifier “w” indicates that the data following the message header is a set of motor controller

parameters. The message is responded to by forwarding the parameters to the motor controllers over the TWI bus.

Main 2: Process Serial R/C Inputs

The FlightCtrl can interface with R/C receivers that output servo-channel commands in either a PPM or serial format. For PPM signals, the command at each channel can be deduced directly from the pulse width and no extra processing is required. For serial signals, the data must be decoded which requires extra computation time. This function checks if a new set of serial R/C data has been received and if so proceeds to decode the individual channel commands from the string of bytes received.

Main 3: Process Sensor Readings

When a new sensor reading is taken it is placed into a ring array (the newest data always replaces the oldest data). For each of the sensors, this function averages the values of the ring array and then applies a neutralizing bias if necessary.

Main 4: Mode Transitions

This function interprets various R/C and external inputs to do mode control. This includes starting/stopping the motors, calibrating the sensors, and heuristics for determining whether or not the hexacopter is in the air.

Main 5: State Estimation

This function estimates the state vector so that state feedback may be used in the control law. The convoluted way in which the stock embedded software performed state estimation was one of the main motivating factors for implementing custom embedded software. The ease with which sensor data can be accessed in this custom software makes modification to this function for the purpose of research very user-friendly. A variety

of state estimators can be implemented such as an optimal observer, or some kind of Kalman filter.

Main 6: Control Law

This function combines inputs from the R/C or some other external component, with the state estimates to produce propeller speed commands. Like the state estimate function, it is very likely to be changed for the purposes of research.

Main 7: Initiate Transmission of Propeller Speed Commands

This function initiates the TWI device, which will transmit the speed setpoint to each of the motor controllers and receive current, maximum command and temperature information in return. This function only initiates the transmission. Actual loading of the transmit buffer and reading of the responses occurs upon interrupt from the TWI device. This is discussed in section 5.3.2.

Main 8: Initiate Transmission of Serial Outputs

This function checks for requests for data transmissions in order of priority. If any request is found, it takes the following action:

1. Assembles the data for transmission
2. Applies a header including destination address and command identifier
3. Encodes the data using the algorithm described in section 5.3.3
4. Computes a value for the circular redundancy check (CRC)
5. Loads the first byte into the UART transmit register

Subsequent byte transmissions will be done upon interrupt from the UART transmitter, as is discussed in section 5.3.2.

Priority	Interrupting Device	Indication
1	TIMER2	10 kHz timer tick
2	TIMER1	R/C PPM pulse received
3	USART0	UART byte received
4	USART0	UART byte sent
5	ADC	ADC sample complete
6	TWI	TWI device ready
7	USART1	R/C serial byte received
8	TIMER3	128 Hz timer tick

Table 5.4: The interrupting tasks in the custom embedded software, listed by priority.

5.3.2 Interrupting Functions

The custom embedded software for the FlightCtrl uses an interrupt-based architecture. In such an architecture, the functions that must be run at the highest rate, are given the highest priority so that they can interrupt any lower-priority function to execute when necessary. For example, in the case of the FlightCtrl, it is desired to have the ADC measure the sensors as often as possible since higher sampling frequency means a larger average of samples can be computed without any penalty in time delay. The ADC on the FLightCtrl's microprocessor can perform readings faster than 10 kHz and function for storing and initiating the next reading takes only a few clock cycles. On the other hand, the function for computing the propeller speed commands from sensor feedback and R/C commands requires several ms of computation, but is only required to be updated around 100 Hz. Clearly, it would be detrimental if the ADC would have to wait for the propeller speed controller to finish before initiating the next computation. Therefore, an interrupt is issued when an ADC reading is complete so that the very short process of storing that reading and initiating the next one can be performed immediately, after which the processor returns to computing the propeller commands.

Table 5.4 lists the various functions that the embedded software performs in order of priority.

Interrupt 1: 10 kHz Timer Tick

This timer (TIMER2) ticks (posts an interrupt) every tenth of a millisecond, but is primarily used as a millisecond counter. The millisecond counter from this timer is used as the basis of several timing functions including wait, delay, and stopwatch functions. The wait function blocks execution of the calling function for a specified time. The delay functions do not block execution and allow an easy check if a specified amount of time has elapsed since the setting of the delay. Stopwatch functions also do not block execution and allow the measurement of the amount of time that has passed since the start o the stopwatch.

Interrupt 2: R/C PPM Pulse Received

Various brands/models of R/C receivers employ different communication protocols. However, the communication is generally encoded into PPM or some serial protocol. In either case, data is received in chunks. For example, for PPM, the duration between each pulse indicates the value of each channel. Therefore, a series of pulses is required to transmit the values of all channels. Similarly, using the serial protocol, the values of all channels are encoded into a series of bytes and each byte is received one at a time.

When a new PPM pulse received, an interrupt is triggered and the time that time that elapsed since the previous pulse (the position) is computed. This duration is converted into a command value via a simple offset.

Interrupt 3: UART Byte Received

When the USART0 device receives a new byte of data, it triggers an interrupt to the CPU. When the CPU responds, it copies the received byte into an input buffer. It then checks that byte for the end character “
r,” indicating the end of a string of bytes. If the byte matches the end character, the CPU sets a flag indicating that there is new data in the buffer that is ready to be

processed. This processing will be done by the main program during its next iteration. The processing is computationally intensive, so performing it in the interrupt handler might block other interrupts.

Interrupt 4: UART Byte Sent

After the USART0 device has sent a byte of data, it triggers an interrupt to the CPU. When the CPU responds, it checks to see if there are any remaining bytes of the serial message to be sent. If so, it copies the next byte into the USART0 transmit register to initiate the next transmission. If there is no byte to be sent, it checks if there are any other requests for data transmission to be handled and proceeds accordingly.

Interrupt 5: ADC Sample Complete

After the ADC has successfully performed an analog-to-digital conversion (referred to as a sample), it triggers and interrupt to the CPU. When the CPU responds to this interrupt, it first records the sample, then sets the next channel (sensor) to be read, and initiates the analog-to-digital conversion on that channel. Readings are recorded into a ring array (such that the oldest sample in the array is replaced with the newest reading). The 6 high-priority sensors are recorded into 12-sample ring arrays and the 2 low-priority sensors (z-axis acceleration and battery voltage) are recorded into 4-sample ring arrays. This function uniformly steps through each channel so that all of the ring arrays are refreshed after 80 samples. The interrupt that triggers this function occurs at approximately 10.2 kHz so the ring arrays are completely refreshed at approximately 128 Hz.

Interrupt 6: TWI Device Ready

The FlightCtrl communicates with the motor controllers using the TWI protocol. This protocol uses two wires: one for a clock signal and one of data. Many devices can

be connected in parallel to the same two wires and data can be both transmitted and received across the same wire. The possibility of conflicts is extremely high with such a simple physical setup. Therefore a set of strict rules is applied for determining which device has control of the data line.

Whenever the TWI device requires direction for how to proceed, it triggers an interrupt to the CPU with a message indicating the current status of the TWI bus. For example, the following are some scenarios that the CPU must respond to:

- Communication has been initialized, send a destination address
- The addressed device acknowledged your transmission request, begin sending data
- The addressed device accepted your last byte, send next byte
- The addressed device didn't respond to your transmission request
- The addressed device won't accept more data
- The addressed device acknowledged your read request, data will follow
- Data received, more data will follow
- Data received, no more will follow
- The addressed device did not respond to your read request
- Some error occurred

In each case, the CPU will take a different response.

Interrupt 7: R/C Serial Byte Received

When an R/C receiver that outputs a serial signal is used, it is connected to the receiver of the USART1 device. Similar to when the USART0 device receives data, when the

USART1 device receives data, it also triggers an interrupt to the CPU. The CPU responds to this interrupt by placing the received byte into a buffer. If the CPU expects that this is the last byte in the message, is sets a flag indicating that new R/C serial data has been received and need to be processed.

Interrupt 8: 128 Hz Timer Tick

This timer (TIMER3) ticks (posts an interrupt) at 128 Hz. The primary purpose of this timer is to initiate the main program at regular intervals. However, this timer is also divided into lower frequencies to perform other low-priority tasks such as updating the LEDs and buzzer.

5.3.3 Serial Communication Protocol

Serial communication between the FlightCtrl and a computer or other external device occurs via universal asynchronous serial receiver and transmitter (UART). A UART uses 2 wires for data transmission: one for transmitting, and one for receiving. Data transfers may be started at any time (hence the "Asynchronous" in UART), but must occur at a pre-agreed rate, called the BAUD rate. Furthermore, transmit and receive can occur simultaneously (i.e. one does not impact the other). When no data is being transferred, the output is set to the high state (1). A single low bit (0) marks the start of a new data packet. For the FlightCtrl USART0 has been set to the following:

- 57600 BAUD (57.6 kbits / second)
- 8 bits of data follow the single start bit
- Data packets are closed with a single high bit and without a parity bit

With the above settings, each byte of data requires a 10-bit packet. This means a maximum data rate of 5.76 kB / second. However, MikroKopter uses an additional protocol to manage the data transfer. In this protocol, multiple bytes of data are wrapped

into a single message with a header and a footer. The header includes the following at the start of the frame:

- The start character "#"
- An address
- A command identifier

The footer includes the following at the end of the frame:

- 2 bytes containing the value for the Cyclic Redundancy Check (CRC)
- The end character "̊"

Finally, to guarantee identification of the start and end characters, data is separated into 3-byte chunks and each chunk is re-encoded into 4-bytes. Each of the encoded bytes has a minimum value of 61, so that it cannot be confused with the start character "#" (= 35) or the end character "̊" (= 13).

The following is a summary of the total overhead:

- 1 extra byte per 3-bytes of data in a frame for encoding
- 6 extra bytes per data frame for identification and error checking
- 2 extra bits per byte to mark the start and end of a transmitted byte

Here are some example maximum transfer rates for certain data sizes given the information above:

- 1-3 bytes: 576 Hz
- 4-6 bytes: 411 Hz
- 7-9 bytes: 320 Hz

- 22-24 bytes: 151 Hz
- 66 bytes: 61.3 Hz (this is the size of the DebugOut structure)

5.3.4 Initialization

When power is first applied to the FlightCtrl board, the microcontroller begins stepping through the main program. The main program is divided into two parts: an initialization part and a free-running loop. The following lists the various steps in order of execution that are performed during initialization.

1. set the functions each of the microcontroller I/O pins
2. start the 1/10th ms timer
3. set the serial communications BAUD rate and ready for data exchange
4. load saved parameters from EEPROM
5. ready the R/C receiver signal decoder
6. initialize communication with motor controllers and query for connected controllers
7. start the 128Hz timer
8. start the ADC
9. reset gyro and accelerometer neutral values
10. calibrate the pressure altitude sensor
11. check the number of LiPO cells connected

5.4 Custom Computing Package

This section describes the components of the custom computer vision package that was assembled to be flown aboard the hexacopter testbed. Figure 5.2 shows this system in detail.

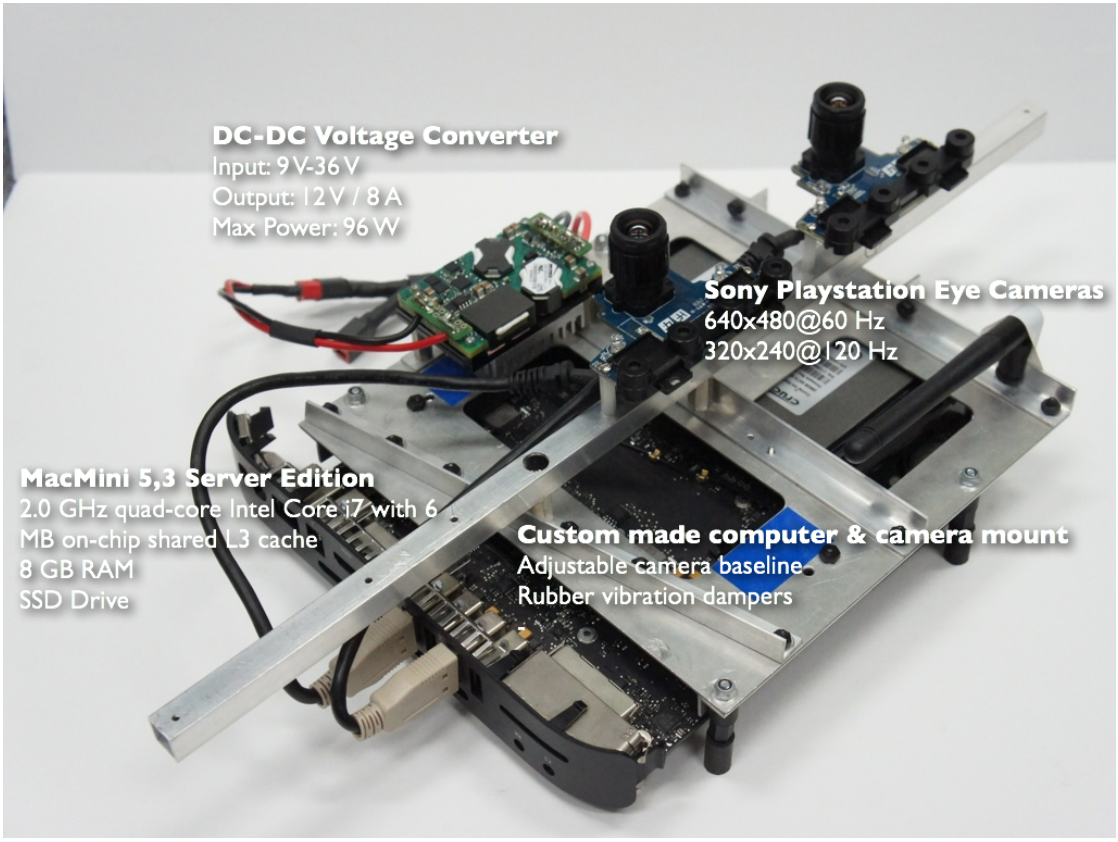


Figure 5.2: A detailed view of the custom computer vision package.

5.4.1 The Onboard Computer

Early in the research, the computer vision software was attempted to be hosted on a small arm-based mobile computing development board called a PandaBoard. However, this platform proved underpowered for high-speed vision. As a result, it was decided to move to a platform with as high a performance to weight ratio as possible under 750 kg.

After some comparison, it was decided to use a Mac mini server. The specifications for the specifications for this computer[4] are as follows computer is as following:

- 2.0 GHz quad-core Intel Core i7-2635QM CPU
- 4 GB 1333 MHz DDR3 RAM
- Intel HD Graphics 3000 GPU with 384 MB DDR4 SDRAM
- Two 500 GB 7200-rpm SATA hard drives

The weight of the off-the-shelf Mac mini (excluding cables) is 1.2 kg. In order to reduce weight the computer was removed from the casing. In addition, the AC-DC converter was discarded (since the computer will be powered from the onboard DC battery) and the 2 hard drives were replaced with one 500 GB solid state drive (SSD). Using the solid state drive both reduces the weight and makes the system less susceptible to vibration.

The stock AC-DC power supply for the Mac mini outputs 12 V at up to 6 A. The power source aboard the hexacopter is a 4-cell LiPO battery, the voltage of which fluctuates between 13.2 V and 16.8 V depending on the load and remaining capacity. Therefore, a power regulator is required to convert the voltage from the LiPO battery to a constant 12 V at a up to 6 A. A suitable part[5] was found from Murata Power Solutions Inc. The DC-DC converter accepts input voltages from 9 V to 36 V and outputs 12 V at a maximum of 9 A.

A custom mount was fabricated to combine the innards of the Mac mini, the camera, the power adapter, an antenna, and some status LEDs into a single unit. The weight of the whole unit is 749 g.

[4] the Mac mini5,3, mid-2011 model was procured as it was the highest performance model at the time
[5] UQQ-12/8-Q12PB-C

Resolution	Frame Rate
800x600	30 Hz
640x480	60 Hz
320x240	120 Hz

Table 5.5: The fastest frame rate for the PlayStation Eye at each of the possible resolutions.

5.4.2 Camera

The camera that is used for the vision package is a Sony PlayStation Eye. The PlayStation Eye is intended to be an accessory to the Sony PlayStation 3 game console, allowing players to interact with games using motion and color detection as well as sound through its built-in microphone array. The price-performance ratio is quite low for this camera. The PlayStation Eye has attached to it a long universal serial bus (USB) cable that provides the camera power and allows for two-way communication.

The native resolution of the PlayStation 3 Eye is 800x600 pixels, but the frame rate output by the camera can be increased by decreasing the resolution. Table 5.5 shows the fastest frame rate at each of the possible output resolutions.

In order to save weight, the housing and weighted base were removed from the PlayStation Eye. The USB cable was also shortened. The lens was removed and replaced with a standard M12 mount.

A number of lenses were compared. Wide angle lenses have the advantage of having many features in view and are less susceptible to motion blur. More narrow lenses give a more precise track. After some experimentation a lens that is slightly wider than the stock PlayStation Eye lens was settled on.

5.4.3 Communicating With FlightCtrl

In order for position and attitude estimates obtained by the computer vision package to be integrated into the rotor speed commands, it must be transmitted to the FlightCntrl.

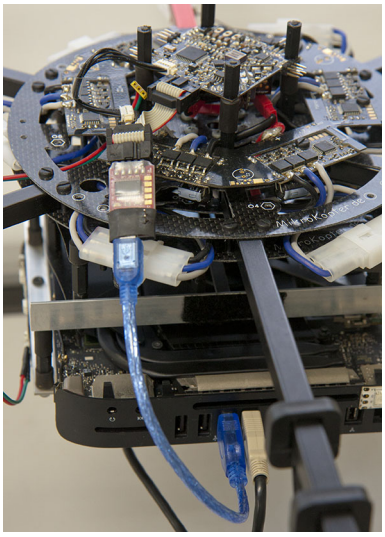


Figure 5.3: The blue cable in this figure is the two-way serial connection between the computer vision package and the FlightCtrl.

This is achieved by serial communication via an FTDI[6] chip that converts USB signals to UART signals and vice versa. Figure 5.3 shows this connection. When connected to the Mac mini, the chip appears to the system to be a serial (COM) port.

[6]FTDI is an acronym for Future Technology Devices International, the company that is the primary supplier of USB to UART chips

Chapter 6

Experimental Results

This chapter describes the various experiments that were performed to prove the capabilities of the algorithms and systems that have been proposed in this thesis. As a first step, the actual implementation of the nominal controller was flight tested for handling qualities. Next, a series of experiments were performed to test the performance of the proposed high-speed vision-based position and attitude sensor. Next, the nominal controller was extended to interface with the computer-vision package and simple position-hold was attempted. Finally, a fully autonomous mission was performed, including takeoffs and landings, using the fully integrated system.

Unfortunately, it was not possible to perform failure-tolerant control experiments for a variety of reasons. However, the proposed controller was tested in simulation and those results will be presented here.

6.1 Nominal Control Testing

Here, nominal control refers to the most basic level of control in which the user is manually controlling the vehicle via radio control (R/C). The nominal controller provides a foundation from which more advanced control techniques can be built. It is extremely

Roll	$k_{\dot{p}} = 2.15$	$k_p = 30$	$k_\phi = 100$	$k_{\phi i} = 150$
Pitch	$k_{\dot{q}} = 2.15$	$k_q = 30$	$k_\theta = 100$	$k_{\theta i} = 120$
Yaw	$k_{\dot{r}} = 0.775$	$k_r = 5$	$k_\psi = 5$	$k_{\psi i} = 1.25$

Table 6.1: The state feedback gains from LQG controller for position control.

important that this controller is robust with good handling qualities since, if anything goes wrong, the user should be able to immediately revert to the nominal controller to manually recover the vehicle.

As mentioned in section 2.6, this controller was designed using a linear model of the linear quadratic Gaussian (LQG) design method (which combines a linear quadratic regulator (LQR) with a Kalman filter). The parameters of this design are several weighting matrices that define the tradeoff between tracking performance and control effort, as well as matrices that describe the covariance of the process and measurement noise. Flight testing is then used to evaluate the handling qualities that result from the choice of the design parameters.

Following some trial and error, observer and state feedback gains were found that gave excellent handling qualities. Table 6.2 presents the state-feedback gains that are implemented in the attitude controller. Vertical control is achieved via open-loop bulk propeller speed command.

6.2 High-Speed Vision Evaluation

Two experiments tested the performance of the high-speed computer vision system's relative position estimation against a motion capture (MoCap) system and the global positioning system (GPS). In these experiments, a high-speed camera was attached to an aircraft which was flown in geometric patterns of known shape while simultaneously recording video and position information with one of the other systems. By algorithmically aligning the resulting trajectories, the error between the two could be estimated.

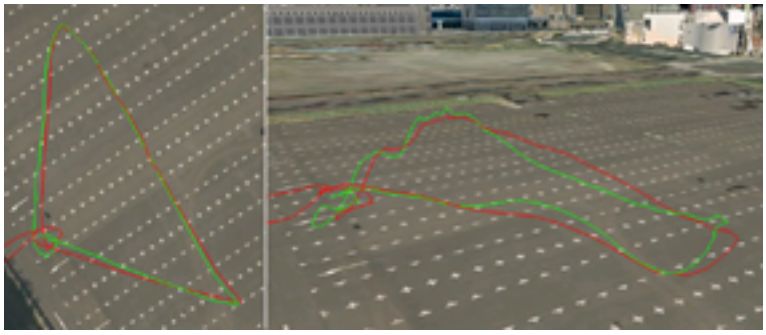


Figure 6.1: A rendering of the GPS position log (red) plotted against a trajectory estimated by PTAM (green). The left image shows an overhead view, while the right image shows a perspective view.

6.2.1 Comparison to GPS

To compare the performance of the high-speed vision-based position estimates to GPS, a high-speed camera was mounted on the hexacopter testbed (described in section 5.1), so that the camera's view was aligned with the hexacopter's z body-axis. In this experiment, the stock FlightCtrl embedded software was used in conjunction with the NaviCntrl add-on board (enabling automatic navigation via GPS and a digital compass). The hexacopter was commanded to fly under automatic navigation through a series of waypoints that formed a large triangle over a vacant parking lot. The values of latitude, longitude, and altitude reported from the on-board GPS sensor were logged to a micro-SD card at a rate of 2 Hz while simultaneously recording high-speed video for the duration of the flight.

The video was then post-processed with the modified version of the parallel tracking and mapping (PTAM) software that is described in section 4.3.2. The resulting flight track was then algorithmically aligned with the flight track from the GPS log for comparison.

Figure 6.1 shows a visualization comparing the trajectories from the GPS logs and the high-speed computer-vision system. The RMS error between the aligned tracks was found to be less than 2.5 m, which is on the order of typical GPS error.

6.2.2 Comparison to Motion Capture

In order to get a better estimate of the precision of the high-speed computer-vision system, is was compared to the position estimates from the MoCap system at Boeing's VEHICLE SWARM TECHNOLOGY LAB (VSTL) [35]. VSTL is a 30 x 15 x 6 m room that couples a MoCap system with the ability to control vehicles based on visual feedback.

The system can track an object of interest by employing multiple cameras to triangulate coordinated pules of visible light that are reflected from markers that are attached to the object. Position accuracy is sub-millimeter and attitude accuracy is sub-degree. Data provided at 100 frames per second with latency of approximately 10 ms. The VSTL MoCap system includes 44 cameras with 16 megapixel and 4 megapixel resolution variants running at a capture rate of up to 1000 frames per second.

A high-speed camera was mounted on a VSTL helicopter such that its view was aligned with the helicopter's z body-axis. The helicopter was then flown autonomously around the VSTL volume. The VSTL SwarmView operator interface commanded the vehicle to take off, follow a series of waypoints, and land. High-speed video was recorded for the duration of the flight and subsequently processed by PTAM offline to compute a flight track. The PTAM and MoCap flight tracks were then algorithmically aligned for comparison. As the 6-degree-of-freedom MoCap data was far more accurate than the data provided by PTAM, it was considered ground truth for validation and comparison.

Figures 6.2, 6.3, and 6.4 visualize the difference between the track acquired by PTAM and the track acquired by the MoCap system. From a 40 second segment of the flight, RMS error of 2.4 cm was calculated between the PTAM and MoCap tracks.

6.2.3 Experimental Results

Experimental results show that the modified version of PTAM is capable of precision as fine as several centimeters at rates as fast as 60 Hz. Both the precision and rate

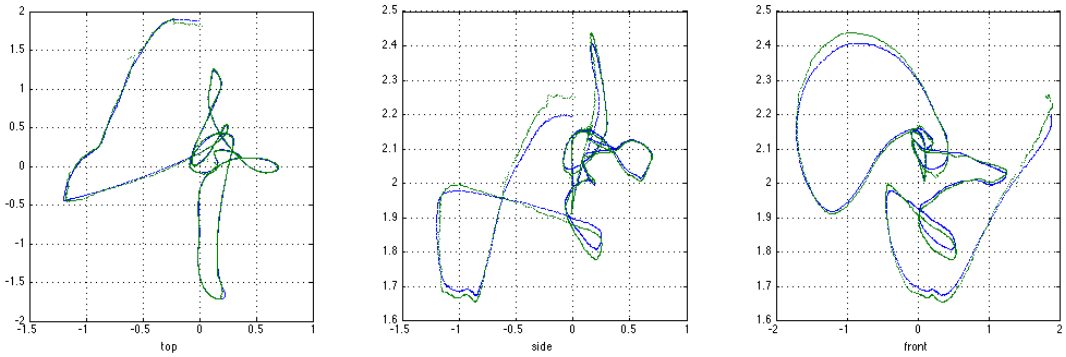


Figure 6.2: The flight paths recorded from MoCap (blue) and PTAM (green) plotted from the top, side and front view.

are orders of magnitude better than that of GPS. However, the scale and position of the map origin is not known a priori and requires additional information in order to be aligned to the earth frame. Currently, a marker of known size and position is used for this purpose.

6.3 High-Speed-Vision Based Navigation

With the capabilities of PTAM confirmed, the next step was to integrate its measurements with the hexacopter's onboard controller (FlightCtrl). As a first step, simple position hold was performed to evaluate the position control gains. With the control gains settled, the next step was to perform autonomous navigation.

6.3.1 Position Hold

The linear controller that was mentioned in section 2.6 and 6.1 was extended to include speed and position. LQG was again employed for the design method and, after some trial and error, design parameters that resulted in a good blend of tracking performance and control effort was found. Figure 6.2 lists the feedback gains that have been implemented in the position controller. Notice that the attitude gains are common with the attitude

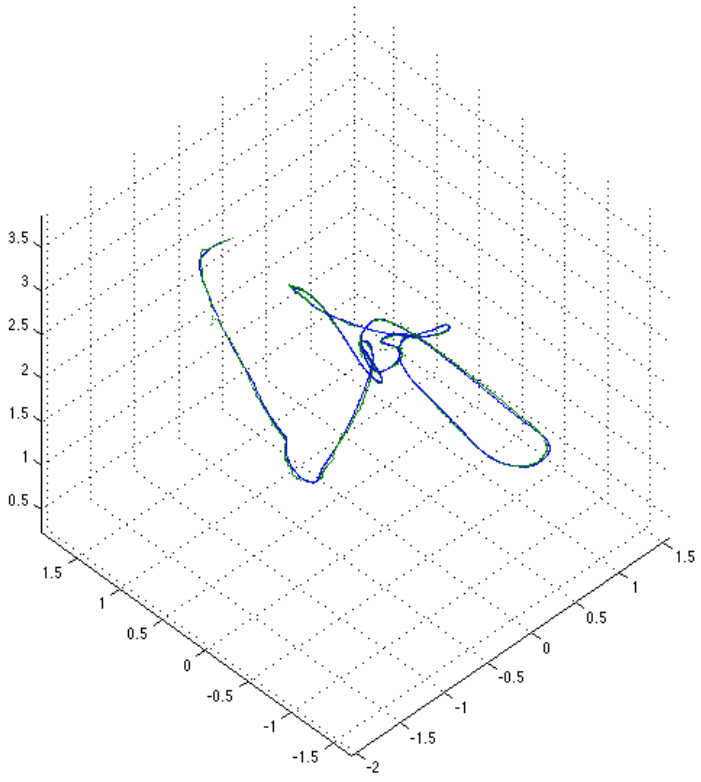


Figure 6.3: The flight paths recorded from MoCap (blue) and PTAM (green) plotted in 3D.

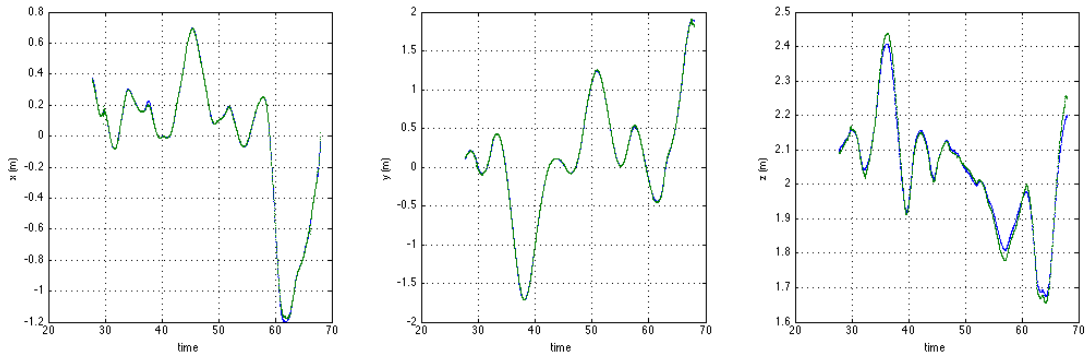


Figure 6.4: The flight paths recorded from MoCap (blue) and PTAM (green) plotted as a function of time. Units are meters.

controller. This ensures smooth transition between attitude and position control and vice versa.

Roll	$k_{\dot{p}} = 2.15$	$k_p = 30$	$k_\phi = 100$	$k_v = 18.9$	$k_y = 15$	$k_{yi} = 6$
Pitch	$k_{\dot{q}} = 2.15$	$k_q = 30$	$k_\theta = 100$	$k_u = 18.9$	$k_x = 15$	$k_{xi} = 6$
Yaw	$k_{\dot{r}} = 0.775$	$k_r = 5$	$k_\psi = 5$	$k_{\psi i} = 1.25$		
Altitude	$k_{\dot{z}} = 6.1$	$k_z = 8.6$	$k_{zi} = 1.72$			

Table 6.2: The state feedback gains from LQG controller for position control.

6.3.2 Autonomous Navigation

With the position control gains settled, the next step was to perform autonomous navigation. The experiment can be summarized as the following steps:

1. The high-speed-vision system is initialized and the tracker's frame is aligned to the earth-inertial frame using a marker of known shape and size.
2. The hexacopter is placed at some arbitrary location away from the marker.
3. The experimental routine is initiated and the hexacopter, under control from the computer-vision package, performs an aggressive takeoff and altitude hold at its current location.
4. Upon reaching the desired altitude, the hexacopter proceeds to the first of three predetermined landing sites.
5. Once it has arrived above the landing site, the hexacopter descends and lands.
6. The hexacopter takes off again within two seconds and proceeds to the next landing site.
7. This process is repeated indefinitely.

Figure 6.5 depicts the plan for this experiment.

The only sensors employed in this experiment were a 3-axis MEMs gyro and our computer-vision system. Even so, we were able to repeat an aggressive set of takeoffs and landings at separated sites. The locations of the landings were repeatable, did not

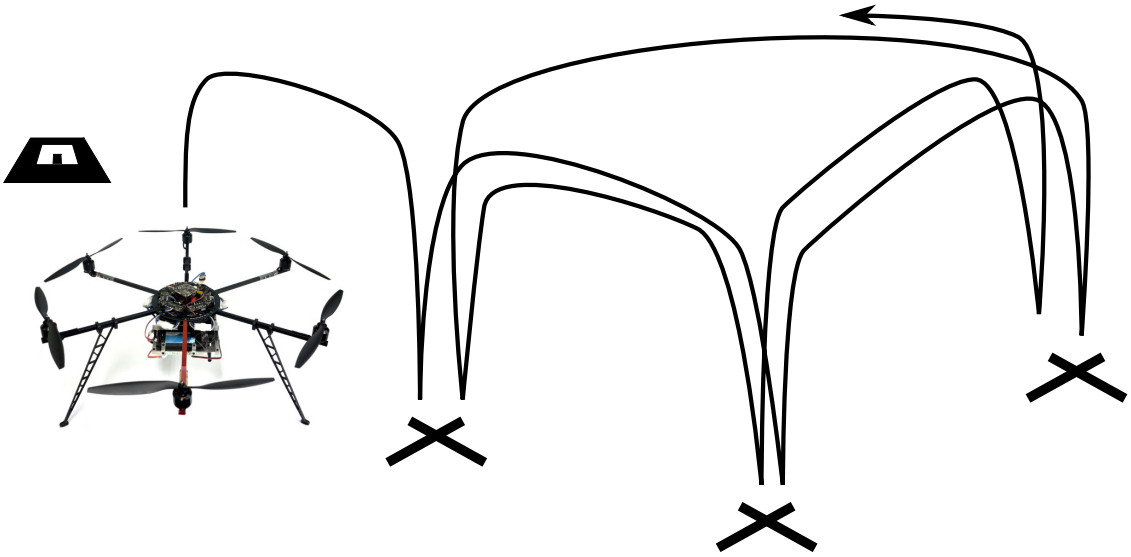


Figure 6.5: The plan for the autonomous navigation experiment.

suffer from drift, and recorded landing precision finer than 20 cm. Figure 6.6 plots the position data that was estimated from the high-speed-vision system.

6.4 Failure-Tolerant Control Simulation

Unfortunately, time did not permit to perform experiments with the fault-tolerant controller presented in chapter 3. The following tasks must be completed for the experiment to be possible:

- The fault-tolerant controller must be implemented in the FlightCtrl embedded software.
- An issue with the accelerometers on the FlightCtrl board must be resolved.
- It would be preferable to have a mechanism devised to physically remove the propeller.

In lieu of experimental results, the following pages present simulation results for various situations, demonstrating the capability and deficiency of this approach.

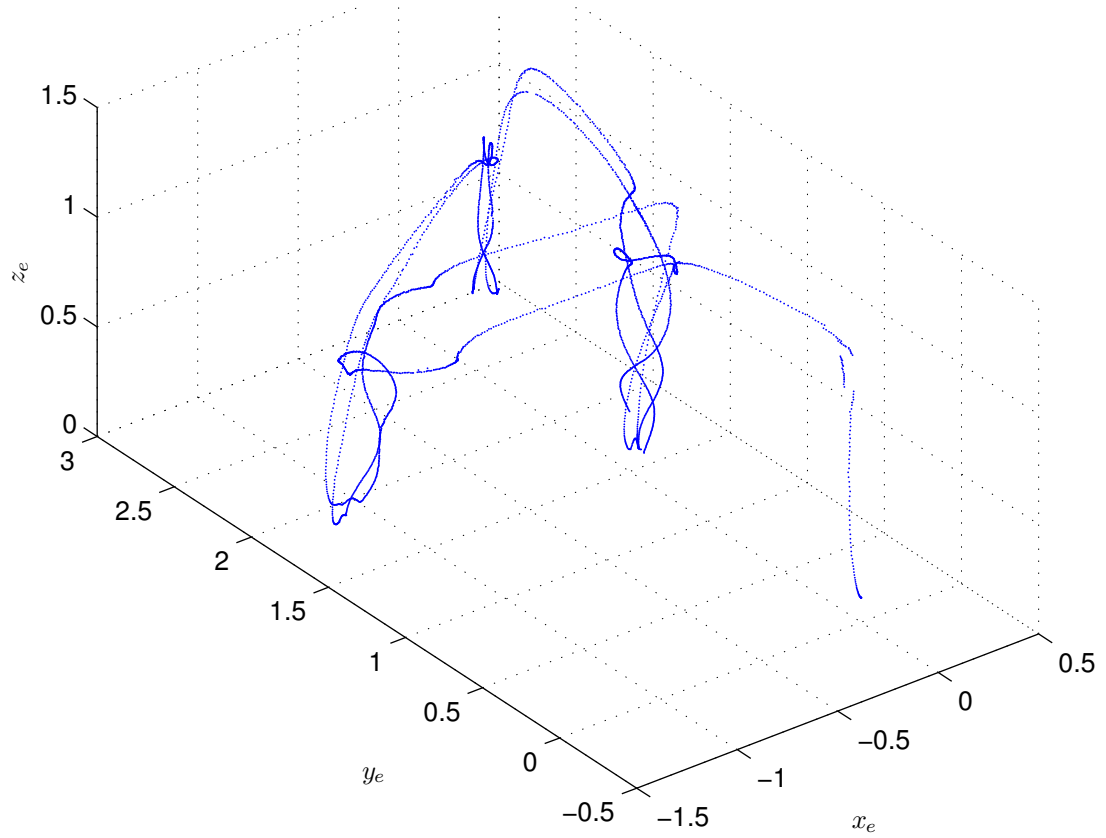


Figure 6.6: The position estimates from the high-speed-vision system for the autonomous navigation experiment.

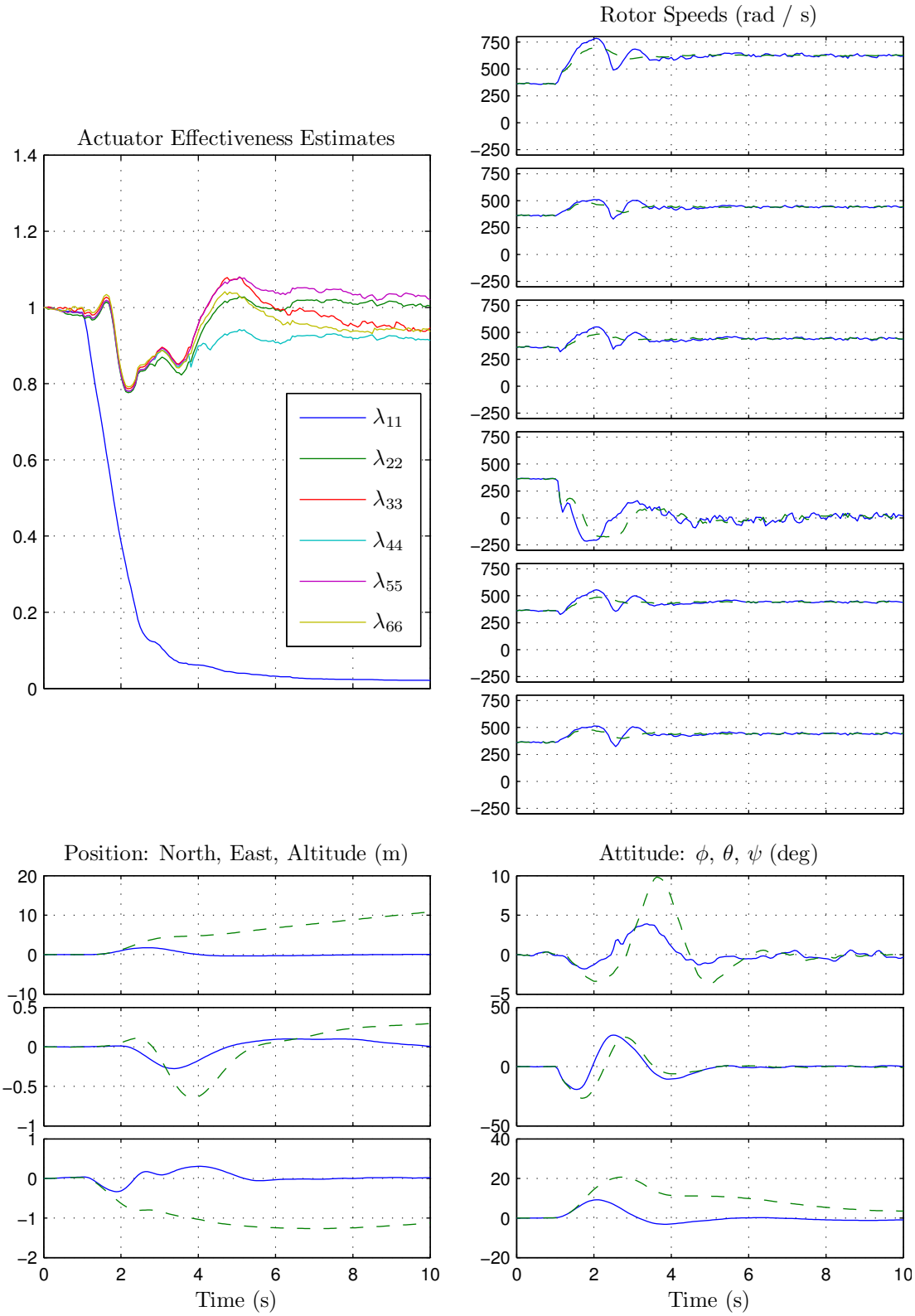


Figure 6.7: From steady hover. At $t = 1$ seconds, propeller #1 becomes completely ineffective. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

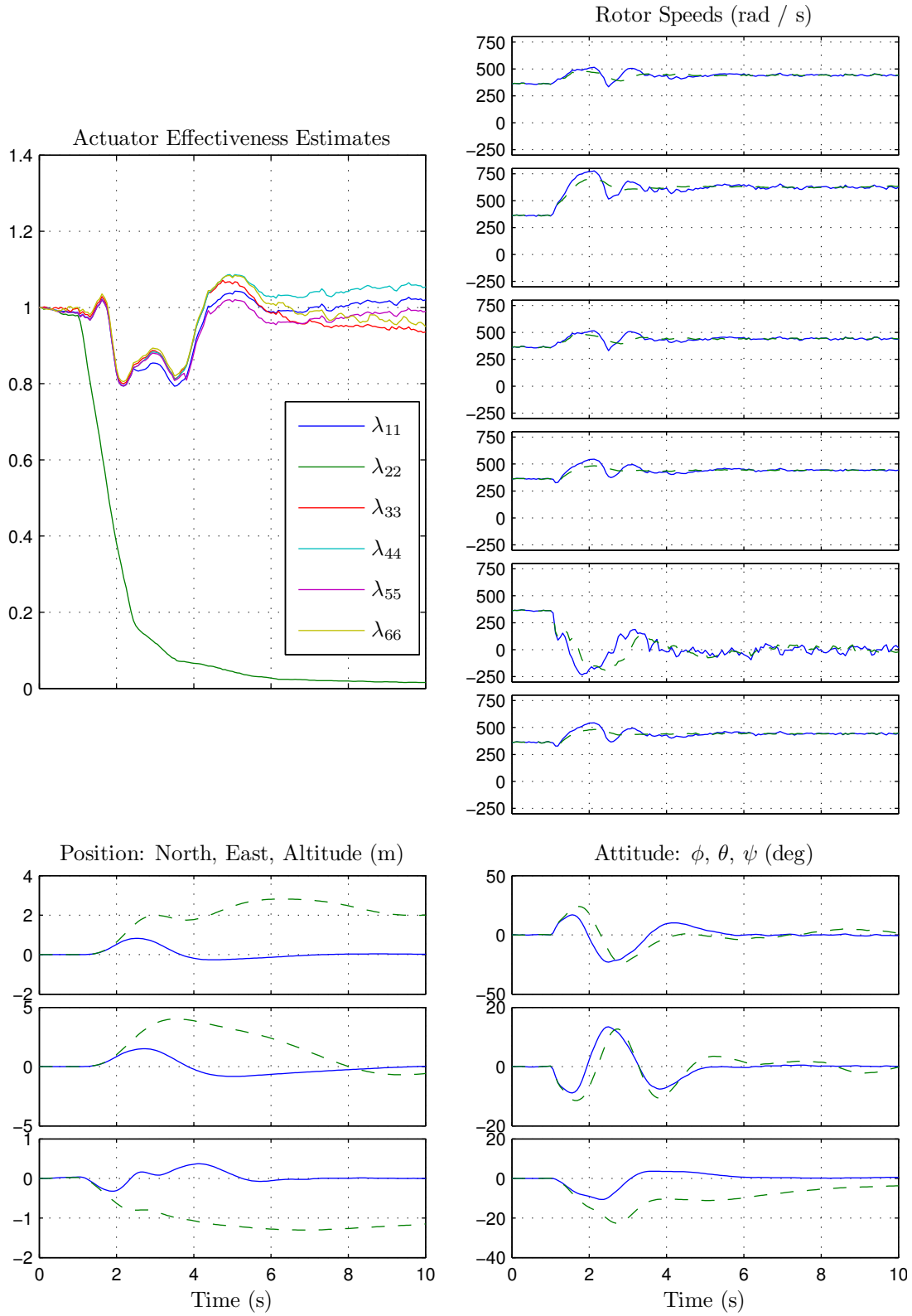


Figure 6.8: From steady hover. At $t = 1$ seconds, propeller #2 becomes completely ineffective. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

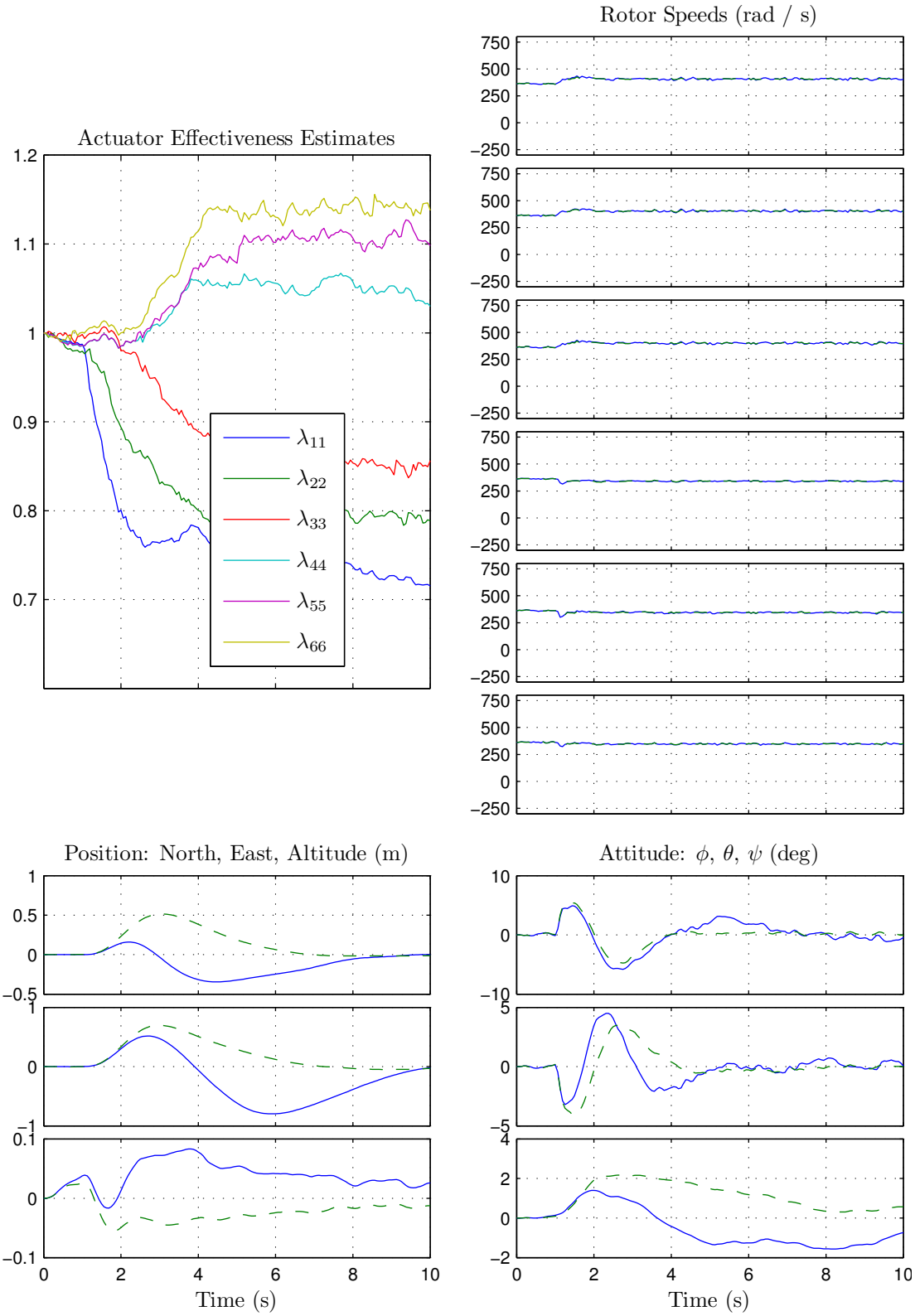


Figure 6.9: From steady hover. At $t = 1$ seconds, propeller #1 becomes 70% effective, propeller #1 becomes 70% effective, propeller #2 becomes 80% effective, propeller #3 becomes 90% effective, propeller #5 becomes 110% effective, propeller #6 becomes 120% effective. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

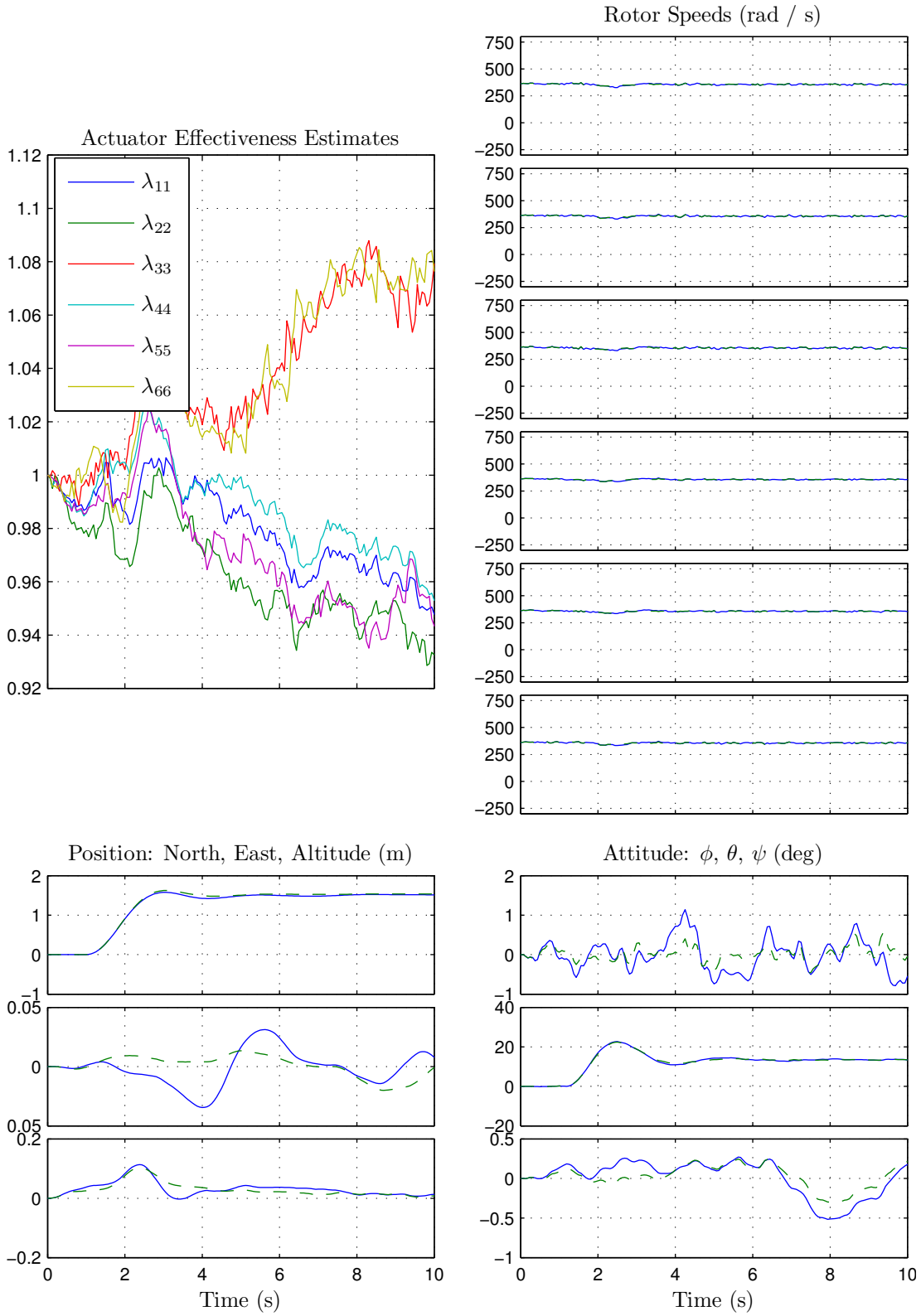


Figure 6.10: From steady hover. At $t = 1$ seconds, a 0.5 g acceleration is applied along the x body-axis. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

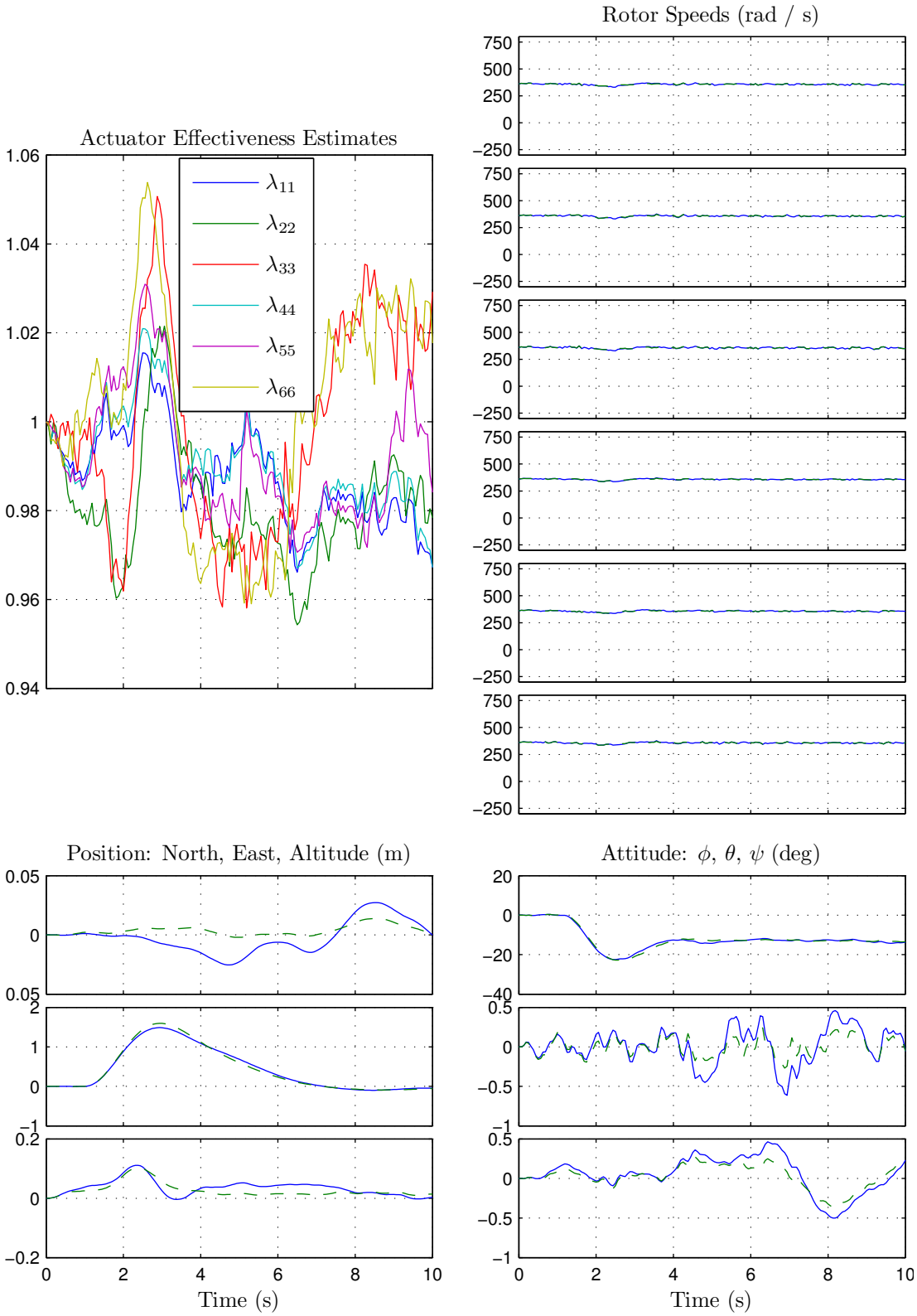


Figure 6.11: From steady hover. At $t = 1$ seconds, a 0.5 g acceleration is applied along the y body-axis. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

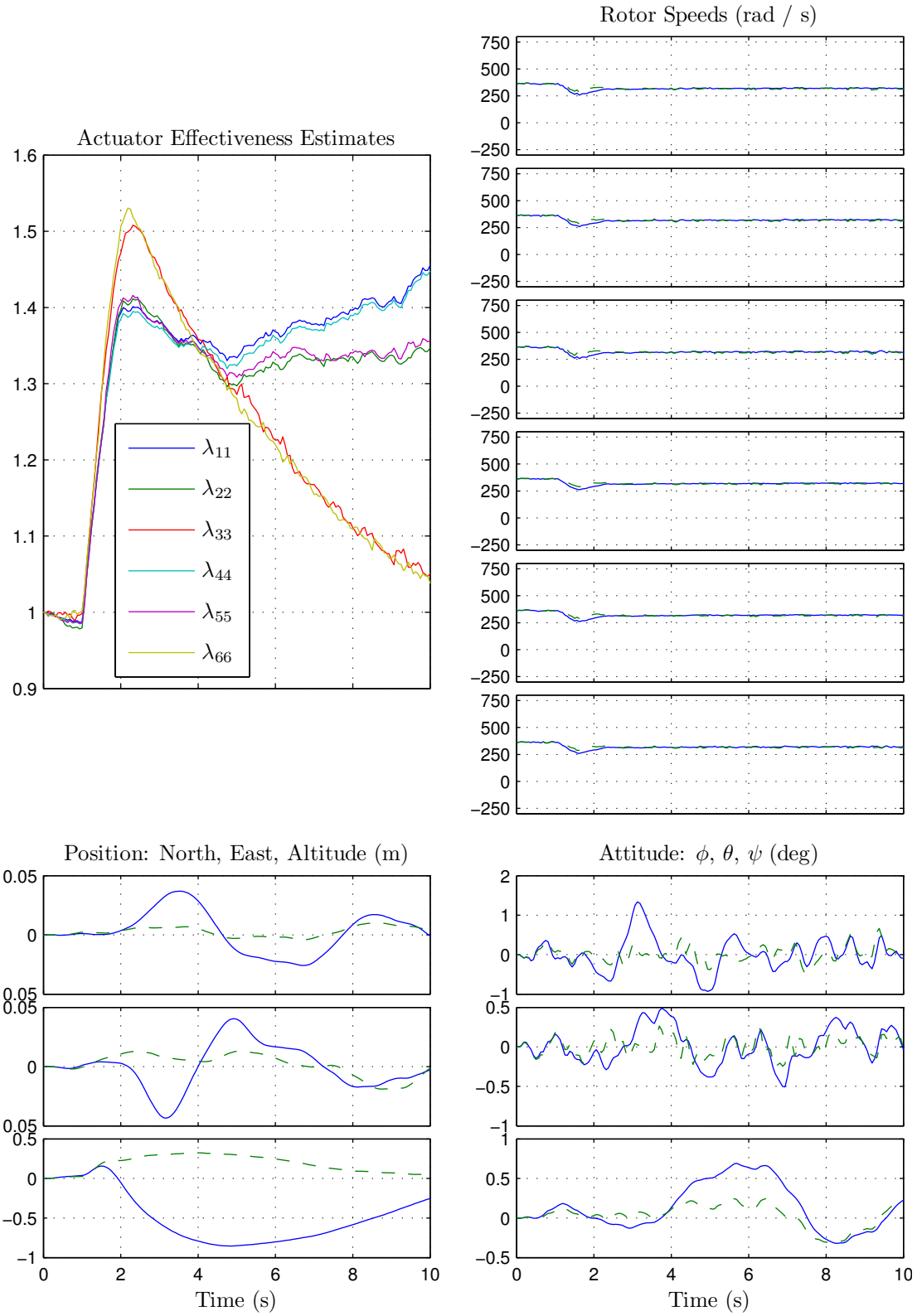


Figure 6.12: From steady hover. At $t = 1$ seconds, a 0.5 g acceleration is applied opposite the z body-axis. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

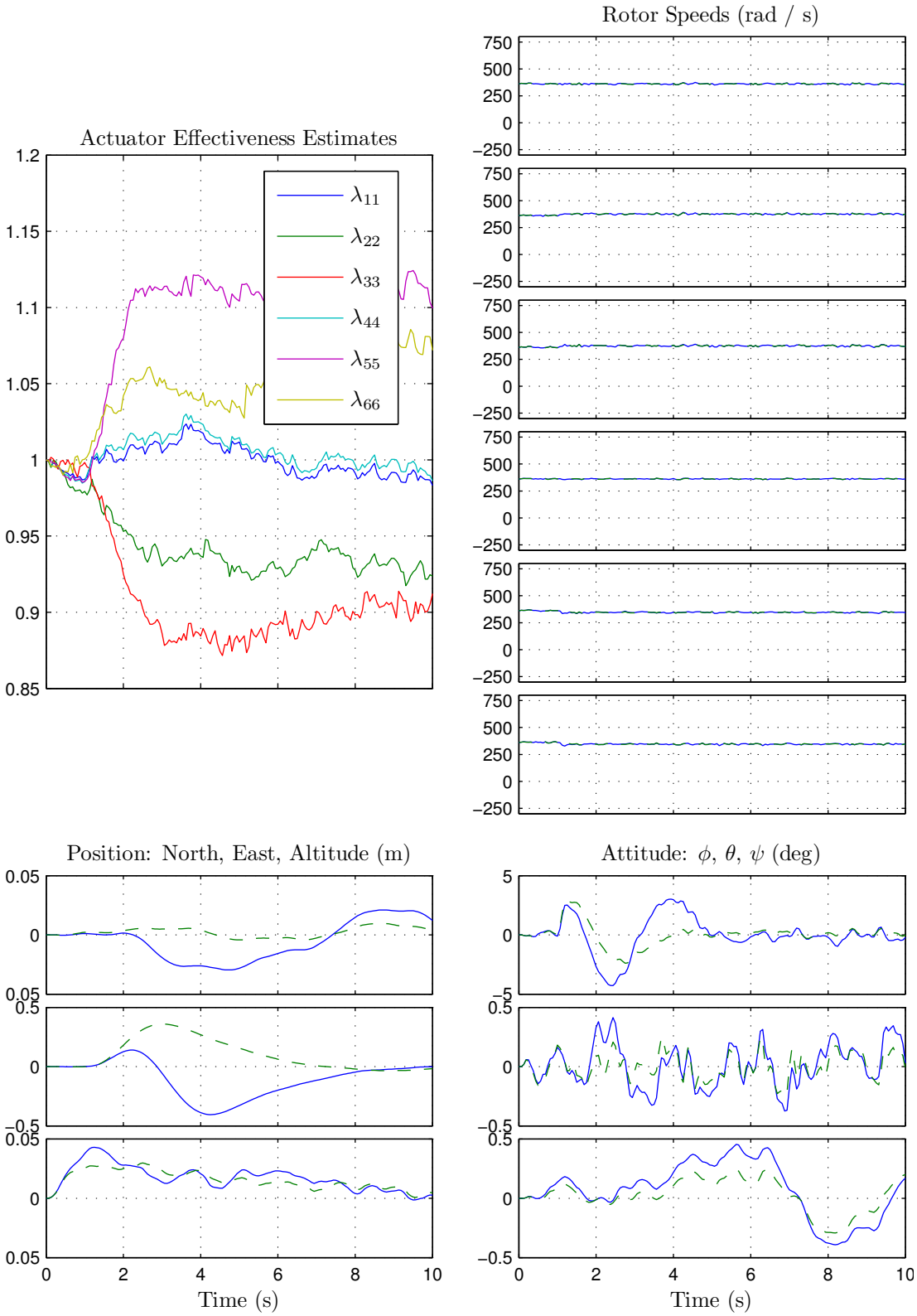


Figure 6.13: From steady hover. At $t = 1$ seconds, a 20 deg/s^2 roll acceleration is applied. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

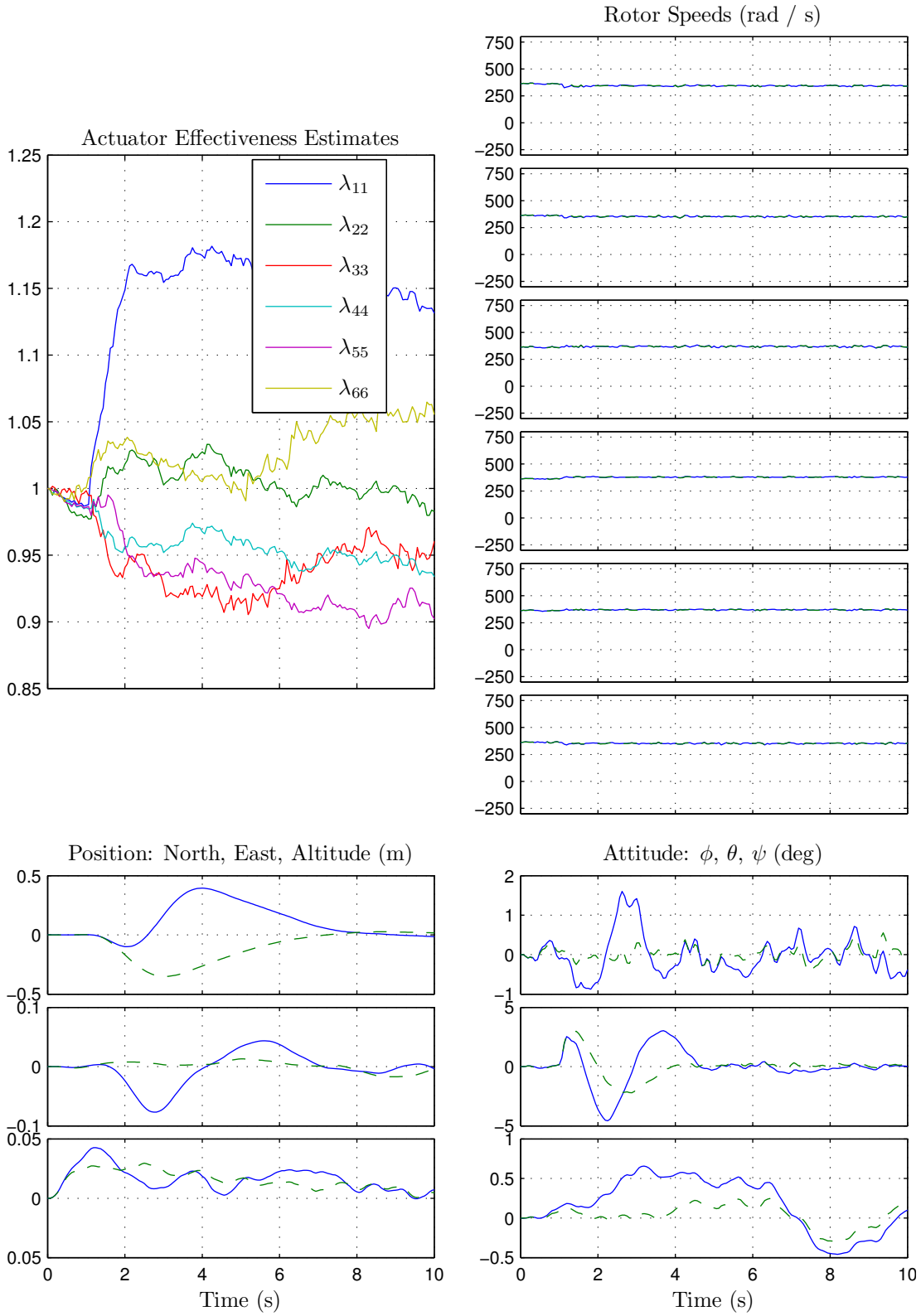


Figure 6.14: From steady hover. At $t = 1$ seconds, a 20 deg/s^2 pitch acceleration is applied. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

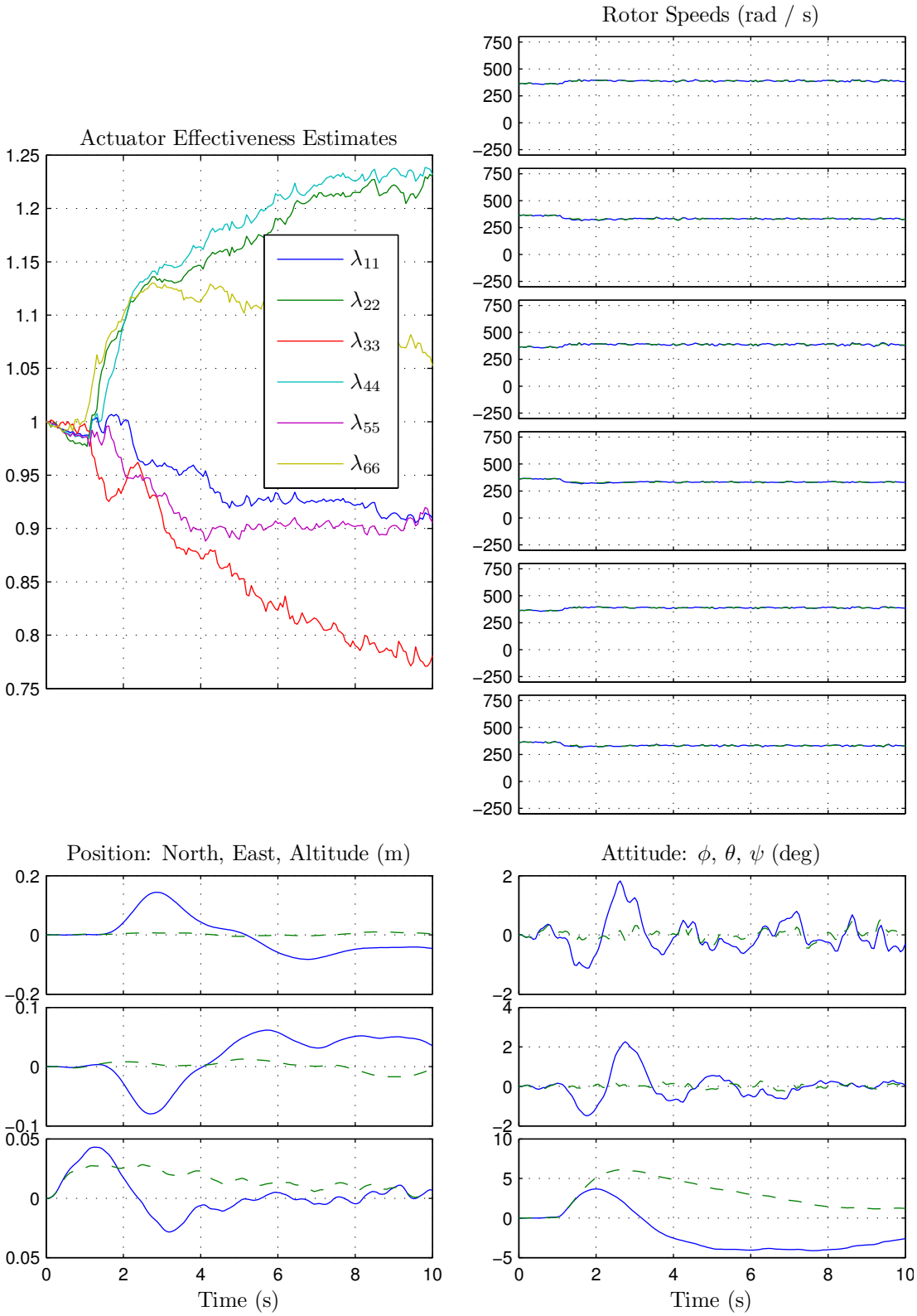


Figure 6.15: From steady hover. At $t = 1$ seconds, an 5 deg/s^2 yaw acceleration is applied. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

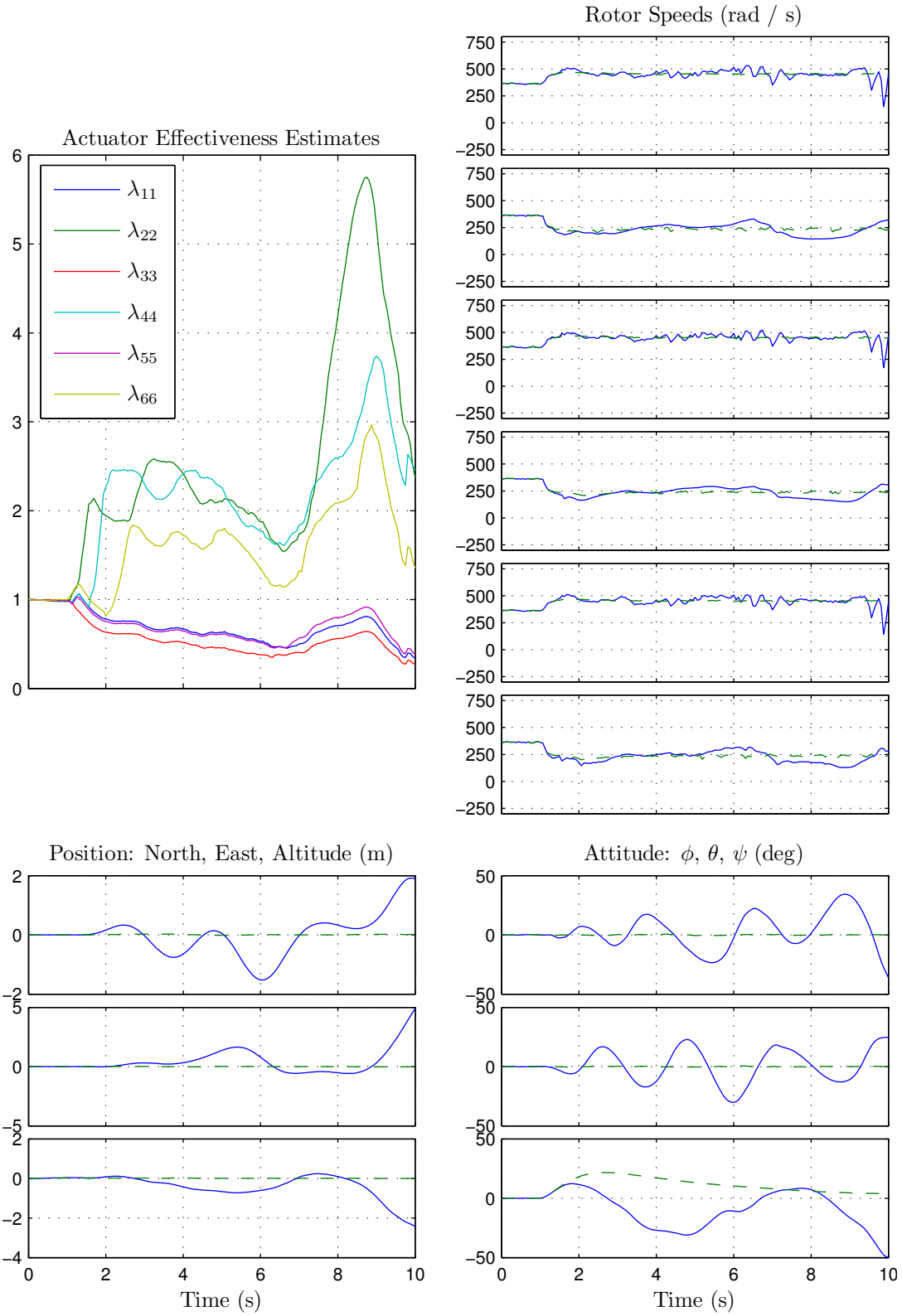


Figure 6.16: From steady hover. At $t = 1$ seconds, an 18 deg/s^2 yaw acceleration is applied. The solid line is the result using the adaptive controller, the dashed is without adaptive control. This response is unstable.

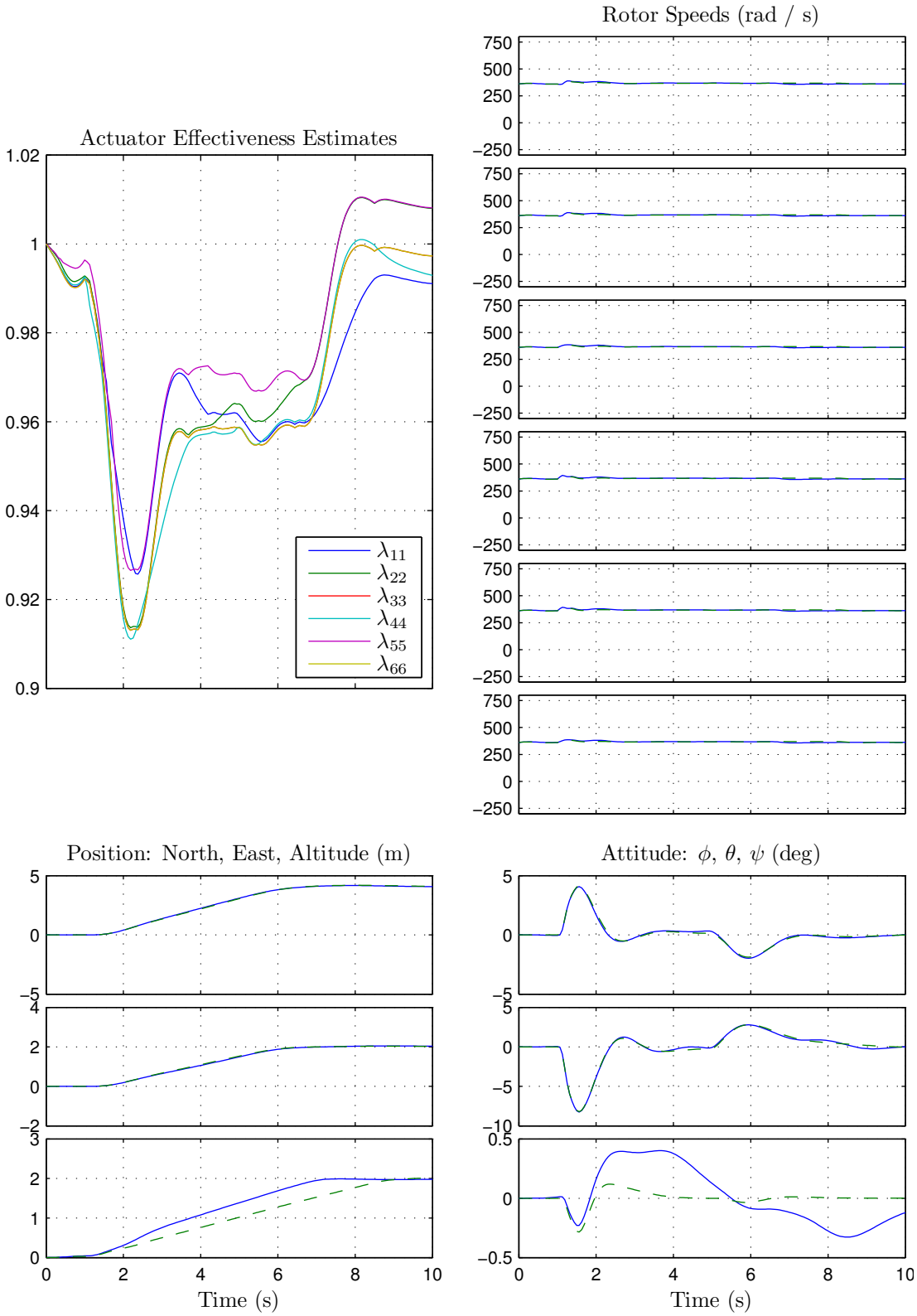


Figure 6.17: From steady hover. At $t = 1$ seconds, the hexacopter is commanded to travel 1 m up, 4 m north and 2 m east. The speed of transit is set to 1 m/s laterally and 0.5 m/s vertically. The solid line is the result using the adaptive controller, the dashed is without adaptive control.

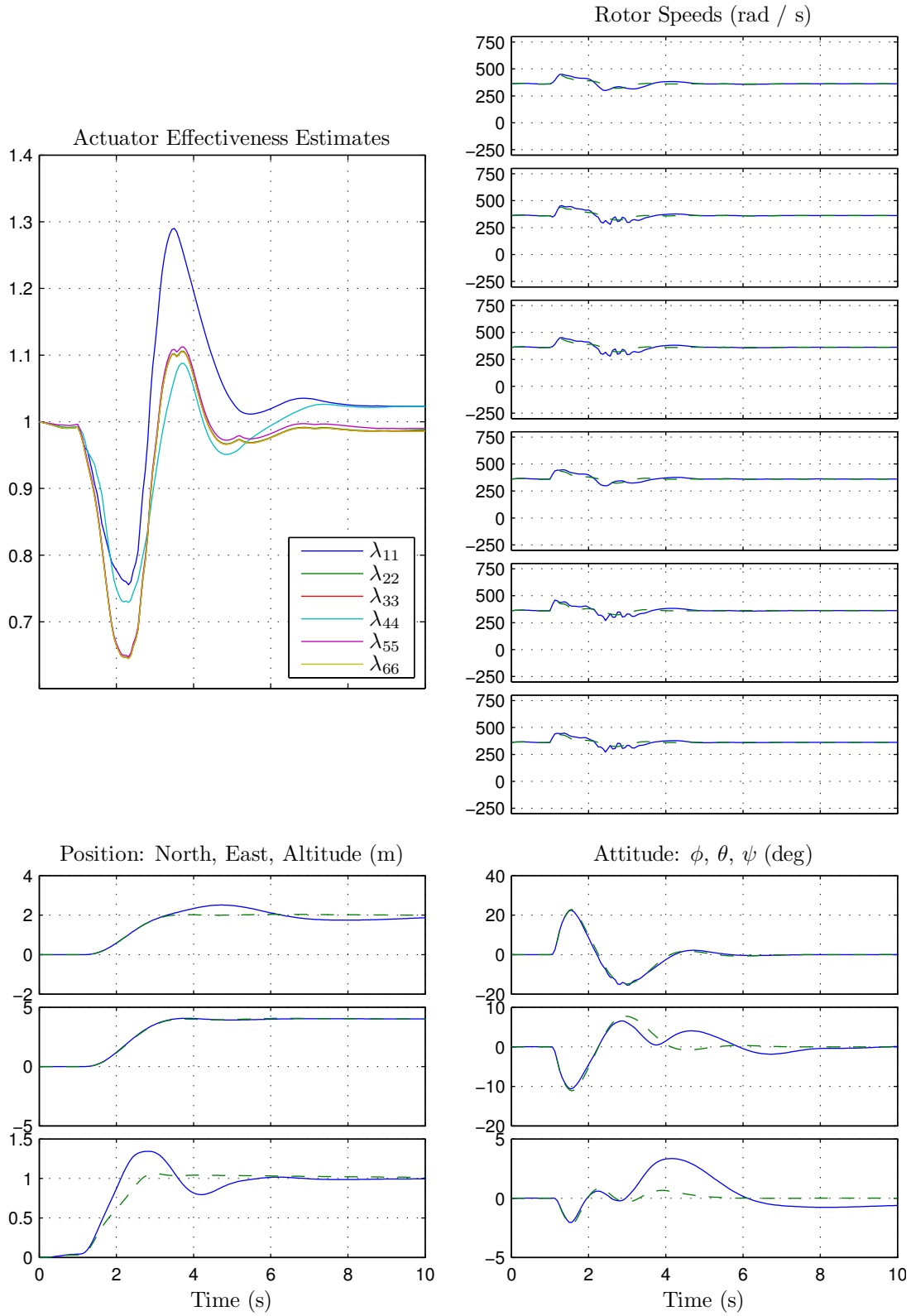


Figure 6.18: From steady hover. At $t = 1$ seconds, the hexacopter is commanded to travel 1 m up, 4 m north and 2 m east. The speed of transit is set to 2.85 m/s laterally and 1 m/s vertically. The solid line is the result using the adaptive controller, the dashed is without adaptive control. This response is nearly unstable.

Chapter 7

Conclusion

7.1 Summary

7.1.1 Failure-Tolerant Control Summary

This thesis proposed a controller to:

1. Decouple a hexacopter's dynamics
2. Simultaneously estimate and adapt to differences in real and design control effectiveness
3. Automatically reconfigure control when a propeller has become critically ineffective

The controller was demonstrated in simulation and was able to quickly identify and adapt to a partial and subsequent total actuator failure. It was analyzed for sensitivity to disturbances, measurement noise, and modeling uncertainty. A fundamental trade-off was found to exist between the speed of convergence and the sensitivity to disturbances, noise, and uncertainty in the state function, especially for small actuator commands. A sufficient condition that guarantees stability in the presence of modeling uncertainty in the input matrix could not yet be determined.

Experimentation using simulation revealed that the controller is able to detect and adapt to total failure of a single propeller. However, performance under normal circumstances seems to diminish qualitatively when the adaptive controller is employed. The adaptive controller is robust to disturbances, but may cause the vehicle to become unstable in extreme cases. The most troubling result is that aggressive maneuvering causes the controller to become unstable.

7.1.2 High-Speed-Vision-Based Navigation Summary

This thesis demonstrated a self-contained computer vision system with size, weight, and power that is compatible with flight on-board a small hexacopter. This sensor is capable of determining position and attitude in GPS-denied areas. The positional accuracy and update rate is at least one order of magnitude superior to that of uncorrected GPS. The ability to perform aggressive autonomous navigation, including takeoffs and landings, was successfully demonstrated.

7.2 Future Work

7.2.1 Failure-Tolerant Control Future Work

Further Investigation of Stability Bounds

As mentioned previously, analytical solutions to the stability bounds could not be determined, but were attempted to be defined experimentally. It was found that the most unacceptable behavior was the instability that occurred during aggressive maneuvering under normal operating conditions. In such cases, good performance was achieved by disabling the adaptive control. It is speculated that the degradation is due to the nonlinear effect of motion of the aircraft on thrust produced by the propellers, but more investigation must be carried out to verify this. Perhaps if the cause can be identified, then measures can be taken to prevent instability.

Explore the Parameter and Operating Space

The simulation results presented in section 6.4 employed a rather high adaptive gain, γ, of 2. However, the baseline controller is quite robust in the absence of adaptation, so the speed of adaptation may be eased, which should improve performance.

It is proposed to use Monte Carlo simulation to explore both the controller parameter space, and the operating space. Doing so should find parameter settings that are particularly robust, and expose operating conditions that are particularly destabilizing.

Refine Analytical Analysis

The analytical analysis of the feedback control scheme presented in this thesis should be revisited. The definition of reasonable analytical stability bounds would be helpful for the control designer.

Perform Flight Test

Proof of the ability to recover from total failure of a single propeller should be demonstrated with flight test. It is proposed to modify one of the arms of the hexacopter so that it can be physically discarded. Doing so has the dual benefit of unambiguously indicating the failure of the propeller and moving the cg away from the failed propeller. The later is important since it allows the propeller opposite the one that has failed to rotate in the positive direction. This is necessary since the motor controllers on the Hexamac testbed cannot rotate in the opposite direction.

Apply This Method to the Octocopter

In the course of studying this controller, it was found that an octocopter is much better suited for failure tolerance. This is because, upon failure of a single propeller, the lifting force required to maintain hover can be distributed among all of the remaining propellers. None of the propellers have to operate in the opposite direction to maintain control.

7.2.2 High-Speed-Vision-Based Navigation Future Work

Constrain the Feature Search with Motion Information

As explained in section 4.2.3, PTAM uses a simple motion model that extrapolates the previous speed of the camera to estimate the current position of the camera. Feature points from the point cloud are then searched for within some radius of the estimated position. That radius is unnecessarily large, reflecting the low confidence in the current position estimation. However, quite lot of information can be gathered about the motion of the camera from the onboard inertial measuring unit (IMU) coupled with knowledge of the aircraft dynamics. Feeding this information to PTAM would allow the search area to be greatly reduced, thereby increasing performance and decreasing the probability of incorrectly matching feature sets to the wrong location in space.

Automatic Initialization

The current implementation of PTAM relies on a user to perform the initialization. This involves a key press that commands PTAM to start the initialization routine, a horizontal movement to establish a stereo baseline, and another key press to declare the end of initialization. This could easily be automated through cooperation between PTAM and the attitude controller. In such an instance, the attitude controller could notify PTAM of "safe" conditions to attempt initialization. PTAM proceeds with initialization until it has detected some stereo baseline. The attitude controller could then verify that the disparity is not due to rotation. If any of those conditions fail, the automatic initialization would simply be repeated.

Set Scale On Ground

Currently, a fiducial marker is used to set the scale of PTAM's map. However, it should be possible to infer scale from the distance between the camera and the ground prior to

takeoff.

Sensor Fusion

Combining position estimates from PTAM with GPS offers substantial benefits. GPS can be used to align the scale and orientation of PTAM's map to the world frame. PTAM can be used to greatly increase the accuracy of position estimates and to maintain of source of position estimates upon loss of GPS, or vice versa.

Adaptive Camera Model

It was found that deficiencies in the camera model are responsible for distortion of PTAM's map. As a result, position estimates become less accurate as the camera moves away from its starting location. It may be possible to develop an adaptive camera model that will adjust its own parameters to combat the distortion of the map when detected.

Vision-Only Control

Given position and attitude estimates of high enough quality, it should be possible to achieve control from the use of computer-vision exclusively. The main obstacle is the need for estimates of angular acceleration at rates of approximately 50 Hz. This means taking the second derivative of attitude estimates. Any noise in the attitude estimate would be greatly amplified, so measurements must be made at a rate much higher than 50 Hz in order to be properly smoothed. It is speculated that vision-only control may be possible using a system that can provide estimates at 200 Hz.

Appendix A

Torque and Propeller Size

This section describes a simple approximation for calculating the torque required to accelerate a propeller as a function of propeller size. For this approximation, aerodynamic forces are ignored and it is assumed that the propeller can be represented as a bar with constant circular cross section as in figure A.1. It is also assumed that the radial cross-section increases proportionally with propeller length.

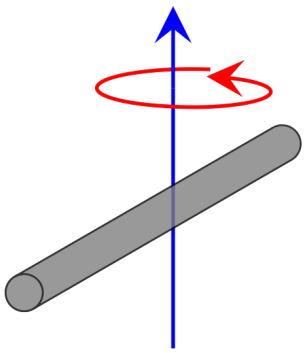


Figure A.1: A propeller represented as a solid rod being rotated about its center

Using the above simplifications, the mass and moment of inertia of the the rotor can be estimated using the following equations:

$$\begin{aligned} mass &= volume \times density \\ &= \pi \times radius^2 \times length \times density \end{aligned} \tag{A.1}$$

$$\begin{aligned} moment\ of\ inertia &= \frac{mass \times length^2}{12} \\ &= \frac{\pi \times radius^2 \times length^3 \times density}{12} \end{aligned} \tag{A.2}$$

Applying the second assumption, that cross-sectional radius increases proportionally with propeller length, allows *radius* to be replaced by $k \times length$, where k is some constant, giving:

$$\begin{aligned} moment\ of\ inertia &= \frac{\pi \times (k \times length)^2 \times length^3 \times density}{12} \\ &= \frac{\pi \times k^2 \times length^5 \times density}{12} \end{aligned} \tag{A.3}$$

Equation A.3 implies that the torque required to accelerate a propeller is proportional to the 5th power of the length of the propeller.

Miniature rotorcraft such as quadrotors and hexacopters typically achieve control by modulating the speeds of each of the propellers. Therefore, the control bandwidth is determined by the maximum achievable acceleration and so equation A.3 has strong implications for the performance of the vehicle.

For example, let us assume quadrotor A employs propellers that are twice the size of those employed for quadrotor B. In order for quadrotor A to have the same control bandwidth as quadrotor B, quadrotor A must be capable of accelerating its propellers at the same rate as quadrotor B. However, the moment of inertia of quadrotor A's propellers will be on the order of 32 times larger than quadrotor B's and so the motors and electronics employed by quadrotor A must be capable of providing 32 times the

torque of quadrotor B. This is not practical and so rotorcraft that achieve control by modulating propeller speed are limited to small propellers with small moments of inertia.

At present it seems that the upper limit for propeller size is on the order of about 20 centimeters.

Appendix B

Rotorcraft Failure Tolerance

Of the many possible configurations for small rotorcraft, some of the more popular are the traditional helicopter, tricopters, quadrotors, hexacopters and octocopters. In all of these configurations, the aim is to control the 3-dimensional position of the aircraft plus the heading direction. Also in all cases, this is achieved by applying a 3-dimensional moment to the body plus a thrust force along the z body-axis (ignoring the unintended side-force from the tail rotor of a helicopter or the swiveling propeller on a tricopter). However, the actuators responsible for manipulating these 4 degrees of freedom are different in each case, and so the consequence of a failed actuator is also different in each case.

Helicopters

Traditional helicopters achieve control decoupling through clever mechanical design. The blades of the main rotor are hinged so that they can be twisted about the axis from root to tip. The twisting is controlled through linkages to a swash plate. The swash plate resembles a pair of discs that are mechanically joined so that they remain parallel to each other, but may slip rotationally along the axis that is normal to the center of the discs. One of the discs rotates with the rotor about the z body-axis, the other is stationary with

the body. Linkages from the flight controls in the body lift and rotate the stationary part of the swash plate, which then lifts and rotates the rotating part, which in turn pushes the blade-twisting linkages.

In the common single main-rotor with anti-torque tail-rotor configuration, a smaller rotor is mounted on the tail with symmetric pitch control (i.e. no swashplate) to produce a force on a lever-arm long enough to generate a moment that exceeds the torque generated by the main rotor.

A helicopter can survive a drive failure through autorotation, in which the rotor is accelerated by twisting the blades downward so that, while descending, the upcoming air blows through the blades like a windmill. When near the ground, blades are turned back up, effectively exchanging the rotational inertia that is stored in the rotor to produce decelerating thrust.

However, other failures are not recoverable. Failure of any of the linkages or failure of the tail-rotor drive will lead to an irrecoverable instability.

Quadrotors

The quadrotor is probably the second most common configuration for small rotorcraft after the traditional helicopter. Unlike the traditional helicopter configuration, quadrotors are mechanically very simple. Quadrotors typically employ 4 fixed-pitch propellers arranged in the same plane so that their hubs coincide with the vertices of an imaginary square. The propellers are typically attached directly to motors, so that the motors are the only moving parts on the vehicle. Adjacent propellers spin in opposite directions.

Control of a quadrotor is typically achieved by modulating the speed of each propeller[1], thereby effecting the lift and torque that each generate. The effect of each propeller can be seen in the rigid body-dynamics, which when linearized about the hover

[1]The use of variable pitch rotors has been demonstrated[30]

condition, are given by:

$$\begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} \\ 0 & -\frac{l}{I_{xx}} & 0 & \frac{l}{I_{xx}} \\ \frac{l}{I_{yy}} & 0 & -\frac{l}{I_{yy}} & 0 \\ -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{B.1}$$

where T_n is the thrust from the n$^{\text{th}}$ propeller and k_Q is the constant of proportionality for propeller torque from thrust.

Equation B.1 shows that each propeller affects 3 of the 4 degrees of control freedom. However, the matrix in that equation is square with full rank, so it is readily invertible. Therefore, the thrust from each of the propellers that is required to achieve some desired acceleration can be solved for directly by inverting equation B.1. Propeller speed commands follow from the assumption that propeller thrust is generally proportional to the square of propeller speed[12].

Figure B.1 shows a typical configuration with a single failed propeller. Notice that hover is not an equilibrium condition in this case. As an intuitive proof, notice that the thrust from the propeller opposite the failed propeller would produce a moment about the arms of other 2 functional propellers. This moment cannot be countered by the those 2 propellers, so the the propeller opposite the failed propeller must be killed. However, since the remaining 2 propellers spin in the same direction, they will produce a yawing moment. This yawing moment will cause the quadrotor to experience yaw acceleration until it is counteracted by drag.

Raffaello D'Andrea of ETH Zurich has demonstrated control of a quadrotor in this condition using a high-speed motion capture system in [45]. However, such control is probably not achievable outside of a lab environment with sensors of the current state-of-the-art.

One might be tempted to try to construct a quadrotor so that the opposite propellers

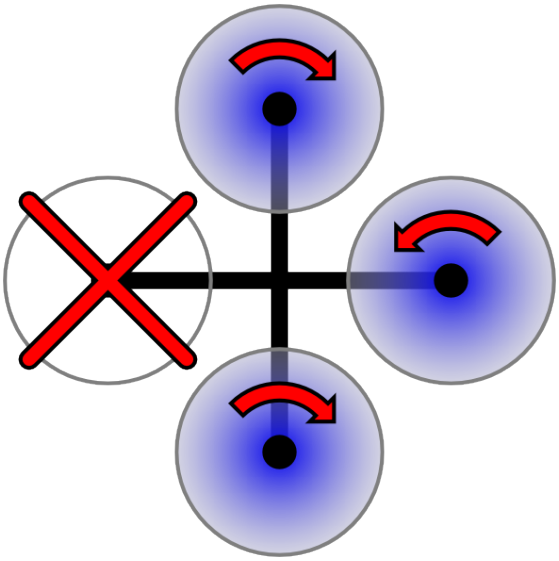


Figure B.1: A typical quadrotor configuration with a single propeller failure

spin in opposite directions, thereby removing the concern of the yaw acceleration in a failure scenario. In such a case, the rigid-body dynamics linearized about hover condition would be given by:

$$\begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} \\ 0 & -\frac{l}{I_{xx}} & 0 & \frac{l}{I_{xx}} \\ \frac{l}{I_{yy}} & 0 & -\frac{l}{I_{yy}} & 0 \\ -\frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{B.2}$$

The matrix in equation B.2 has a rank of 3 and therefore, it is not possible to decouple control of all 4 of the desired degrees of control freedom.

Tricopters

Tricopters typically employ 3 fixed-pitch propellers arranged in the same plane. At least one of the propellers is actuated so that it can be swiveled about the arm that it is mounted to. The swiveling of one of the propellers allows the thrust to be vectored to

control the yawing moment (though it will also produce some side force).

Figure B.2 shows an example of a typical tricopter configuration and a failed propeller. It is clear that failure of any of the propellers results in an immediate and catastrophic roll and/or pitching moment.

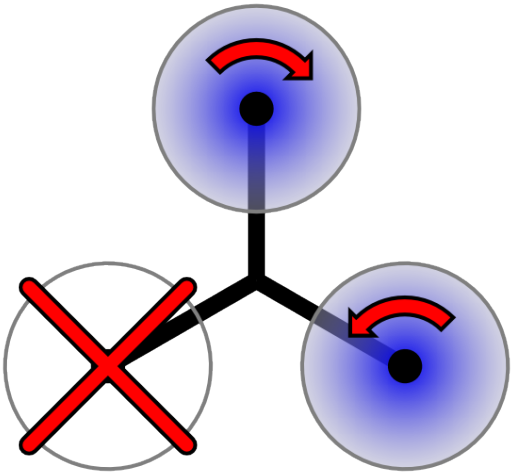


Figure B.2: A typical tricopter configuration with a single propeller failure

Hexacopter

Hexacopters typically employ 6 fixed-pitch propellers arranged in the same plane so that their hubs coincide with the vertices of an imaginary hexagon. Like the quadrotor, adjacent propellers spin in opposite directions. The rigid-body dynamics of a hexacopter

linearized about the hover condition are given by:

$$\begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} \\ 0 & -\frac{\sqrt{3}l}{2I_{xx}} & -\frac{\sqrt{3}l}{2I_{xx}} & 0 & \frac{\sqrt{3}l}{2I_{xx}} & \frac{\sqrt{3}l}{2I_{xx}} \\ \frac{l}{I_{yy}} & \frac{l}{2I_{yy}} & -\frac{l}{2I_{yy}} & -\frac{l}{I_{yy}} & -\frac{l}{2I_{yy}} & \frac{l}{2I_{yy}} \\ -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} \tag{B.3}$$

As in the case of the quadrotor, it is desired to invert these dynamics to solve for the propeller thrust that is required to achieve the desired accelerations. However, the matrix in equation B.3 is not square and therefore is not invertible. Rather, there exists an infinite number of pseudoinverse matrices (matrices that will result in the 4-by-4 identity matrix when multiplied on the right of the matrix in equation B.3). Using the Moore-Penrose pseudoinverse enforces the hexagonal symmetry of the hexacopter. In this way, a pure $\dot{w}$ command is distributed evenly across each of the propellers and angular acceleration commands about each of the arms result in identical thrust commands but rotated 60 degrees. The inverted dynamics using the Moore-Penrose pseudoinverse is given by:

$$\begin{bmatrix} T_{1cmd} \\ T_{2cmd} \\ T_{3cmd} \\ T_{4cmd} \\ T_{5cmd} \\ T_{6cmd} \end{bmatrix} = \begin{bmatrix} -\frac{m}{6} & 0 & \frac{I_{yy}}{3l} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6} & -\frac{\sqrt{3}I_{xx}}{6l} & \frac{I_{yy}}{6l} & \frac{I_{zz}}{6k_Q} \\ -\frac{m}{6} & -\frac{\sqrt{3}I_{xx}}{6l} & -\frac{I_{yy}}{6l} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6} & 0 & -\frac{I_{yy}}{3l} & \frac{I_{zz}}{6k_Q} \\ -\frac{m}{6} & \frac{\sqrt{3}I_{xx}}{6l} & -\frac{I_{yy}}{6l} & -\frac{I_{zz}}{6k_Q} \\ -\frac{m}{6} & \frac{\sqrt{3}I_{xx}}{6l} & \frac{I_{yy}}{6l} & \frac{I_{zz}}{6k_Q} \end{bmatrix} \begin{bmatrix} \dot{w}_{cmd} \\ \dot{p}_{cmd} \\ \dot{q}_{cmd} \\ \dot{r}_{cmd} \end{bmatrix} \tag{B.4}$$

Figure B.3 shows a typical configuration with a single failed propeller. Notice that for the hover condition, the propeller opposite the one that has failed must be operated near zero speed. As a proof, notice that any thrust from the propeller opposite the one

that has failed produces a corresponding moment due to its lever-arm. To counteract this moment, the propellers adjacent to the failed propeller must produce more thrust than the combination of the other 3 propellers. Since those two propellers rotate in the same direction, this implies that they will also produce more yawing moment than the combination of the other 3 propellers, resulting in an yaw acceleration. Therefore, the yawing moment can only be balanced when the propeller opposite the one that has failed is not producing any thrust or yawing moment.

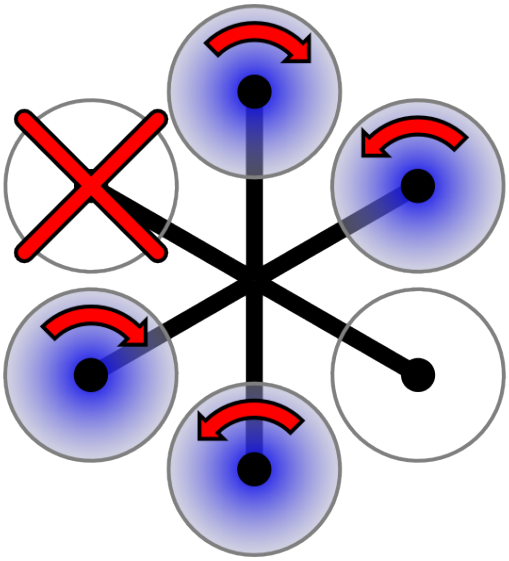


Figure B.3: A typical hexacopter configuration with a single propeller failure

Octocopter

Octocopters employ 8 fixed-pitch propellers and can be found in a variety of shapes. Octocopters are well-suited for failure tolerance. As an example, let us assume that the propeller hubs coincide with the vertices of an imaginary octagon. Then the rigid-body

dynamics linearized about the hover condition are given by:

$$\begin{bmatrix} \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} \\ 0 & -\frac{1}{\sqrt{2}}\frac{l}{I_{xx}} & -\frac{l}{I_{xx}} & -\frac{1}{\sqrt{2}}\frac{l}{I_{xx}} & 0 & \frac{1}{\sqrt{2}}\frac{l}{I_{xx}} & \frac{l}{I_{xx}} & \frac{1}{\sqrt{2}}\frac{l}{I_{xx}} \\ \frac{l}{I_{yy}} & \frac{1}{\sqrt{2}}\frac{l}{I_{yy}} & 0 & -\frac{1}{\sqrt{2}}\frac{l}{I_{yy}} & -\frac{l}{I_{yy}} & -\frac{1}{\sqrt{2}}\frac{l}{I_{yy}} & 0 & \frac{1}{\sqrt{2}}\frac{l}{I_{yy}} \\ -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} & -\frac{k_Q}{I_{zz}} & \frac{k_Q}{I_{zz}} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \tag{B.5}$$

In the case of a single propeller failure, there are many possible ways to recover control of all 4 state rates, including reversion to a quadrotor-like configuration. However, it is also possible to maintain the use of all 7 remaining propellers. For example, if the first propeller fails, then the first column of the matrix in equation B.5 is replaced with zeros. The Moore-Penrose pseudoinverse can then be used to solve for the thrusts that will result in the desired accelerations. In the case of pure thrust along the z body-axis without angular rotation, the Moore-Penrose pseudoinverse results in the following contribution of thrust from each propeller:

$$\begin{bmatrix} 0\% & 17\% & 19\% & 8\% & 12\% & 8\% & 19\% & 17\% \end{bmatrix}^T$$

Appendix C

Inertial to Body-Axis Relations

This appendix details the modeling of the motion of a rigid body using a combination of an earth-fixed reference frame and a body-fixed reference frame. This is convenient for several reasons.

- The earth frame can be treated as approximately inertial.[1]
- Newton's law of inertia only holds in inertial frames. In other non-inertial the motion of the frame relative to an inertial frame must be accounted for.
- The moment-of-inertia matrix is constant in the body-fixed frame of a rigid body, but varies with rotation in the inertial frame.
- In the case of an aircraft, it is easier to account for most forces in in the body frame.
- The end goal is typically to control the attitude and/or position of the vehicle in the earth frame.

[1]The earth-fixed frame is not inertial due to its rotation, but the associated Coriolis effect can be assumed unlikeable in our case.

Euler Angles

As a preliminary step, let us develop a transformation from the earth frame to the body frame through the use of Euler angles.[2] The Euler angles represent a set of 3 elemental rotations that, when performed about the axes of a coordinate system, can fully describe the transformation from the original orientation of the coordinate system to any other orientation. In the aerospace convention, these angles are a rotation ϕ (heading angle) about the z earth-axis (pointing down), then a rotation θ (pitch angle) about the newly formed y axis, and finally a rotation ψ (roll angle) about the latest x axis. This series of rotations is demonstrated in

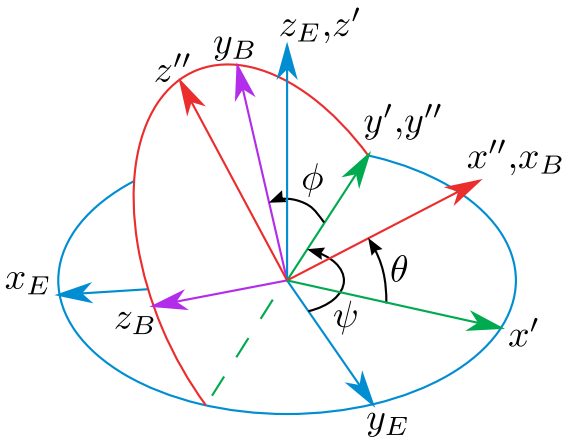


Figure C.1: The series of elemental rotations that define the Euler angles in the aerospace convention.

A rotation ψ about the z-axis is given by:

$$R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.1}$$

[2]While commonly referred to as "Euler angles" in aerospace, they are more precisely referred to as "Tait–Bryan angles" since they represent rotations about three distinct axes, while proper Euler angles use the same axis for both the first and third elemental rotations.

A rotation θ about the y-axis is given by:

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{C.2}$$

A rotation ϕ about the z-axis is given by:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{C.3}$$

Combing these three elemental rotations in the sequence of the aerospace convention gives the full 3-dimensional transformation matrix:

$$\begin{aligned} R = R_\phi R_\theta R_\psi = \\ = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \end{aligned} \tag{C.4}$$

The reverse of a rotation is simply the transpose of the rotation matrix. This can be seen intuitively in the case of the single axis rotation matrices in equations C.1, C.2, and C.3. To prove this for the full 3-dimensional matrix, let us construct the inverse as:

$$
\begin{aligned}
R^{-1} = (R_\phi R_\theta R_\psi)^{-1} = R_\psi^{-1} R_\theta^{-1} R_\phi^{-1} = R_\psi^T R_\theta^T R_\phi^T = \\
&= \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \\
&= R^T
\end{aligned} \tag{C.5}
$$

Newton's Second Law

Newton's second law for linear and angular motion of a rigid body about its center of gravity is well known as:

$$\mathbf{F}_E = m\mathbf{a} \tag{C.6a}$$

$$\mathbf{M}_E = I_E\boldsymbol{\alpha} \tag{C.6b}$$

where F_E, $\mathbf{M}_E$, $\mathbf{a}$, and $\boldsymbol{\alpha}$ are the force, moment, linear acceleration, and angular acceleration vectors respectively, all given in the earth frame (considered inertial). I_E is the moment of inertia matrix of the rigid body with respect to the earth frame. m is the mass (independent of reference frame). Note that the moment-of-inertia matrix varies in the earth frame varies with the rotation of the rigid body.

It is desired to transform these equations into the body frame since the moment of inertia matrix I is constant in the body frame and the applied forces and moments are generally more easily accounted for in the body frame.

Let us first consider the linear version of Newton's second law and attempt to transform it to the body axis by distributing the rotation matrix, R.

$$R\mathbf{F}_E = mR\dot{\mathbf{V}}_E$$
$$\Rightarrow \mathbf{F}_B = mR\dot{\mathbf{V}}_E \tag{C.7}$$

Now, let us recognize that velocity transformed from the inertial (earth) frame to the body frame can be described by:

$$\mathbf{V}_B = R\mathbf{V}_E = R\dot{\mathbf{X}} \tag{C.8}$$

Taking the derivative of velocity in the body axis gives:

$$\dot{\mathbf{V}}_B = \dot{R}\mathbf{V}_E + R\dot{\mathbf{V}}_E$$
$$\Rightarrow R\dot{\mathbf{V}}_E = \dot{\mathbf{V}}_B - \dot{R}\mathbf{V}_E \tag{C.9}$$

Using the definition of a derivative, we can rewrite the last term in equation C.9 as:

$$\begin{aligned}\dot{R}\mathbf{V}_E &= \lim_{\Delta t\to 0}\frac{R(t+\Delta t)-R(t)}{\Delta T}\mathbf{V}_E \\ &= \lim_{\Delta t\to 0}\frac{R(t+\Delta t)\mathbf{V}_E - R(t)\mathbf{V}_E}{\Delta T}\end{aligned} \tag{C.10}$$

Note that $\dot{R}$ is equivalent to the rotational velocity, $\mathbf{\Omega}_B$, of the body frame with respect to the inertial frame, and that limit of the vector difference in the numerator of equation C.10 is equivalent to $-(\mathbf{\Omega}_B \times \mathbf{V}_B)\Delta t$ since $-(\mathbf{\Omega}_B \times \mathbf{V}_B)$ is the tangential component of the vector $\mathbf{V}_B$ due to the rotation. Therefore we arrive at the result:

$$\dot{R}\mathbf{V}_E = -(\mathbf{\Omega}_B \times \mathbf{V}_B) \tag{C.11}$$

Substituting equation C.11 into equation C.7, we arrive at Newton's second law for linear motion of a rigid body with all terms given in the body frame:

$$\mathbf{F}_B = m\left(\dot{\mathbf{V}}_B + \boldsymbol{\Omega}_B \times \mathbf{V}_B\right) \tag{C.12}$$

Substituting the body-frame vector quantity $I\boldsymbol{\Omega}_B$ for $m\mathbf{V}_B$ the above process gives a similar result for Newton's second law for angular motion of a rigid body:

$$\mathbf{M}_B = I\dot{\boldsymbol{\Omega}}_B + \boldsymbol{\Omega}_B \times I\boldsymbol{\Omega}_B \tag{C.13}$$

Euler Angle Rates

Revisiting equation C.11 by expanding the cross product into a matrix multiplication and substituting equation C.8 for $\mathbf{V}_B$ gives the following relation:

$$\dot{R}\cancel{\mathbf{V}_E} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} R\cancel{\mathbf{V}_E} \tag{C.14}$$

Examining equation C.14 element $\dot{R}_{31}$ gives the following relationship:

$$\begin{aligned} -\dot{\theta}\cos\theta &= \begin{bmatrix} 0 & r & -q \end{bmatrix} \cdot \begin{bmatrix} -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}^T \\ \Rightarrow \dot{\theta} &= \frac{-1}{\cancel{\cos\theta}}(r\sin\phi\cancel{\cos\theta} - q\cos\phi\cancel{\cos\theta}) \\ \Rightarrow \dot{\theta} &= q\cos\phi - r\sin\phi \end{aligned} \tag{C.15}$$

Using the same procedure for element $\dot{R}_{32}$ plus a substitution of equation C.15 for $\dot{\theta}$ gives:

$$\dot{\phi}\cos\phi\cos\theta - \dot{\theta}\sin\phi\sin\theta = \begin{bmatrix} -r & 0 & p \end{bmatrix} \cdot \begin{bmatrix} -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}^T$$

$$\Rightarrow \dot{\phi} = \frac{1}{\cos\phi\cos\theta}\left[r\sin\theta + p\cos\phi\cos\theta + (q\cos\phi - r\sin\phi)\sin\phi\sin\theta\right]$$

$$\Rightarrow \dot{\phi} = p\frac{\cos\phi\cancel{\cos\theta}}{\cancel{\cos\phi}\cos\theta} + q\frac{\cancel{\cos\phi}\sin\phi\sin\theta}{\cancel{\cos\phi}\cos\theta} + r\frac{(1-\sin^2\phi)\sin\theta}{\cos\phi\cos\theta}$$

$$\Rightarrow \dot{\phi} = p + q\sin\phi\tan\theta + r\frac{\cos^{\cancel{2}}\phi\tan\theta}{\cancel{\cos\phi}}$$

$$\Rightarrow \dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{C.16}$$

Using the same procedure for element $\dot{R}_{23}$ gives:

$$\dot{\psi}\cos\theta\cos\psi - \dot{\theta}\sin\theta\sin\psi = \begin{bmatrix} 0 & r & -q \end{bmatrix}$$

$$\cdot \begin{bmatrix} \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \end{bmatrix}^T$$

$$\Rightarrow \dot{\psi} = \frac{1}{\cos\theta\cancel{\cos\psi}}\left[r(\cos\phi\cancel{\cos\psi} + \cancel{\sin\phi\sin\theta\sin\psi}) + q(\sin\phi\cancel{\cos\psi} - \cancel{\cos\phi\sin\theta\sin\psi})\right.$$

$$\left. + (\cancel{q\cos\phi} - \cancel{r\sin\phi})\cancel{\sin\theta\sin\psi}\right]$$

$$\Rightarrow \dot{\psi} = p\cos\phi\sec\theta + q\sin\phi\sec\theta \tag{C.17}$$

Assembling equations C.16, C.15, and C.17 into a single matrix equality gives:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{C.18}$$

References

[1] Markus Achtelik et al. "Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments". In: *IEEE ICRA. http://iarc. angelstrike. com/oldauvs/5th_mission/2009SymposiumPapers/2009MIT. pdf* (2009).

[2] Spencer Ahrens et al. "Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 2643–2648.

[3] Erdinç Altuğ, James P Ostrowski, and Camillo J Taylor. "Control of a quadrotor helicopter using dual camera visual feedback". In: *The International Journal of Robotics Research* 24.5 (2005), pp. 329–341.

[4] Atmel. *8-bit Atmel Microcontroller with 16/32/64/128K Bytes In-System Programmable Flash*. 8272D–AVR–05/12. Datasheet. May 2012.

[5] Radek Baránek. "Attitude control of multicopter". In: *electroscope* (5 2012). ISSN: 1802-4564.

[6] Fernando Caballero et al. "Vision-based odometry and SLAM for medium and high altitude flying UAVs". In: *Unmanned Aircraft Systems*. Springer, 2009, pp. 137–161.

[7] Abbas Chamseddine et al. "Model Reference Adaptive Fault Tolerant Control of a Quadrotor UAV". In: *Proc. of AIAA InfoTech@ Aerospace 2011: Unleashing Unmanned Systems* (2011), pp. 29–31.

[8] Jaime Del-Cerro et al. “An autonomous helicopter guided by computer vision for inspection of overhead power cable”. In: *Proceedings of the Workshop WS6 Aerial Robotics, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems-IROS*. 2002, pp. 69–78.

[9] Cédric Demonceaux, Pascal Vasseur, and C Regard. “Omnidirectional vision on UAV for attitude computation”. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE. 2006, pp. 2842–2847.

[10] Analog Devices. *ADXRS620*. ADXRS620. Datasheet. Version B. Sept. 2010.

[11] Damien Eynard et al. “UAV altitude estimation by mixed stereoscopic vision”. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 646–651.

[12] Gary Fay. *Derivation of the aerodynamic forces for the mesicopter simulation*. 2001.

[13] HiSystems GmbH. *Altitude Sensor*. URL: `http://www.mikrokopter.de/ucwiki/en/heightsensor` (visited on 07/18/2013).

[14] HiSystems GmbH. *MK Basicset Hexa XL*. URL: `https://www.mikrocontroller.com/index.php?main_page=product_info&products_id=561` (visited on 07/18/2013).

[15] Nicolas Guenard, Tarek Hamel, and Robert Mahony. “A practical visual servo control for an unmanned aerial vehicle”. In: *Robotics, IEEE Transactions on* 24.2 (2008), pp. 331–340.

[16] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Vol. 2. Cambridge Univ Press, 2000.

[17] Daniel Henell. “Airborne SLAM Using High-Speed Vision”. MA thesis. University of Tokyo, 2013.

[18] Stefan Hrabar and Gaurav Sukhatme. "Vision-based navigation through urban canyons". In: *Journal of Field Robotics* 26.5 (2009), pp. 431–452.

[19] Stefan Hrabar and Gaurav S Sukhatme. "Omnidirectional vision for an autonomous helicopter". In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 1. IEEE. 2003, pp. 558–563.

[20] Peter J Huber. *Robust statistics*. Springer, 2011.

[21] Hojjat A Izadi, Youmin Zhang, and Brandon W Gordon. "Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation". In: *World Congress*. Vol. 18. 1. 2011, pp. 6343–6348.

[22] Andrew Johnson, James Montgomery, and Larry Matthies. "Vision guided landing of an autonomous helicopter in hazardous terrain". In: *Robotics and automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international conference on*. IEEE. 2005, pp. 3966–3971.

[23] Hassan K Khalil. *Nonlinear systems*. Vol. 3. Prentice hall Upper Saddle River, 2002.

[24] Evgeny Kharisov, Naira Hovakimyan, and Karl J Åström. "Comparison of several adaptive controllers according to their robustness metrics". In: *AIAA Guidance, Navigation and Control Conference*. 2010.

[25] Georg Klein and David Murray. "Parallel tracking and mapping for small AR workspaces". In: *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE. 2007, pp. 225–234.

[26] Georg Klein and David Murray. "Parallel Tracking and Mapping on a Camera Phone". In: *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*. Orlando, Oct. 2009.

[27] Philip Lamb. *ARToolKit*. URL: `http://www.hitl.washington.edu/artoolkit` (visited on 07/18/2013).

[28] Tong Li. "Nonlinear and fault-tolerant control techniques for a quadrotor unmanned aerial vehicle". PhD thesis. Concordia University, 2011.

[29] B Ludington, EN Johnson, and GJ Vachtsevanos. "Vision based navigation and target tracking for unmanned aerial vehicles". In: *Advances in Unmanned Aerial Vehicles*. Springer, 2007, pp. 245–266.

[30] Buddy Michini et al. "Design and flight testing of an autonomous variable-pitch quadrotor". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2978–2979.

[31] Pratap Misra, Brian P Burke, and Michael M Pratt. "GPS performance in navigation". In: *Proceedings of the IEEE* 87.1 (1999), pp. 65–85.

[32] Alison A Proctor, Eric N Johnson, and Thomas B Apker. "Vision-only control and guidance for aircraft". In: *Journal of Field Robotics* 23.10 (2006), pp. 863–890.

[33] Hugo Romero, Ryad Benosman, and Rogelio Lozano. "Stabilization and location of a four rotor helicopter applying vision". In: *American Control Conference, 2006*. IEEE. 2006, 6–pp.

[34] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[35] Emad Saad et al. "Vehicle swarm rapid prototyping testbed". In: *Proc AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*. 2009.

[36] Iman Sadeghzadeh et al. "Fault-tolerant trajectory tracking control of a quadrotor helicopter using gain-scheduled PID and model reference adaptive control". In:

Annual Conference of the Prognostics and Health Management Society. Vol. 2. 2011.

[37] Srikanth Saripalli, James F Montgomery, and Gaurav S Sukhatme. "Visually guided landing of an unmanned aerial vehicle". In: *Robotics and Automation, IEEE Transactions on* 19.3 (2003), pp. 371–380.

[38] Srikanth Saripalli et al. "Detection and tracking of external features in an urban environment using an autonomous helicopter". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 3972–3977.

[39] Freescale Semiconductor. *Integrated Silicon Pressure Sensor Altimeter/Barometer Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated*. MPX4115. Datasheet. Version 5. Aug. 2006.

[40] Jianbo Shi and Carlo Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE. 1994, pp. 593–600.

[41] Brian L Stevens and Frank L Lewis. "Aircraft control and simulation". In: (2003).

[42] Henrik Stewenius, Christopher Engels, and David Nistér. "Recent developments on direct relative orientation". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 60.4 (2006), pp. 284–294.

[43] STMicroelectronics. *LIS344ALH*. LIS344ALH. Datasheet. Version 3. Apr. 2008.

[44] NSTB/WAAS T&E Team. *Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report*. Technical Report. Federal Aviation Administration, Jan. 31, 2012, p. 23.

[45] TED. *Raffaello D'Andrea: The astounding athletic power of quadcopters.* URL: http://http://www.ted.com/talks/raffaello_d_andrea_the_astounding_athletic_power_of_quadcopters.html (visited on 07/18/2013).

[46] Wikipedia. *Lithium polymer battery — Wikipedia, The Free Encyclopedia.* 2013. URL: http://en.wikipedia.org/w/index.php?title=Lithium_polymer_battery&oldid=562985418 (visited on 07/18/2013).

[47] Brian Williams, Georg Klein, and Ian Reid. "Real-time SLAM relocalisation". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE. 2007, pp. 1–8.

[48] Xiaobing Zhang et al. "Fault tolerant control for quadrotor via backstepping approach". In: *The 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, Orlando, Florida, USA*. 2010.

[49] Youmin Zhang and Abbas Chamseddine. "Fault Tolerant Flight Control Techniques with Application to a Quadrotor UAV Testbed". In: *Automatic Flight Control Systems-Latest Developments, InTech* (2012).