ABSTRACT

Title of thesis: DEVELOPMENT OF A QUADCOPTER TEST ENVIRONMENT AND RESEARCH PLATFORM

Christian Patrice De Prins, Master of Science, 2015

Thesis directed by: Professor Nuno C. Martins
Dept. of Electrical and Computer Engineering

This thesis first uses a model-based systems engineering approach to model, design, and implement a quadcopter test environment and research platform (TERP). TERP provides quadcopter state information, using a motion capture system, which can be used with custom feedback strategies to enable controlled flight.

Next, it makes use of control theory to develop two controllers for quadcopter flight trajectory tracking: one based on linear quadratic regulation (LQR) and one based on model reference adaption. Simulations of both controllers are done in MATLAB using Simulink and seek to demonstrate the improved performance of the adaptive controller over the LQR controller in flight trajectory tracking with payload uncertainties. Flight tests with the LQR controller are then done to validate the TERP System.

DEVELOPMENT OF A QUADCOPTER TEST ENVIRONMENT AND RESEARCH PLATFORM

by

Christian Patrice De Prins

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2015

Advisory Committee:
Professor Nuno C. Martins, Chair/Advisor
Professor Mark Austin
Professor Sarah Bergbreiter

Dedication

The work enclosed in this thesis is dedicated to all students following in the lab after my departure. May this work serve as a platform for future research.

Acknowledgments

Thank you to my parents, Christian and Rebecca, and my brother, Laurent, for their constant support during the development of this thesis; my advisor, Dr. Nuno Martins, for his guidance and support throughout my time at Maryland; my committee members, Dr. Sarah Bergbreiter and Dr. Mark Austin, for their service and feedback; Dr. Krishnaprasad, for his help in teaching me many aspects of control theory; my coworkers at the Department of Fraternity and Sorority Life at the University of Maryland, for their support and encouragements; the men in Zeta Psi who have encouraged me and worked late nights alongside me during my stay at Maryland; my friends, for keeping my spirits high and always providing me with great company, especially the following: Matt Bahsen, for his help working on implementation aspects of the TERP system in the Cooperative Autonomy Lab; Bryan Janaskie, for serving as my systems engineering consultant; Lena Krain, for encouragements and help with writing sessions and editing; Selleck LeBlanc, for being willing to donate a leg to me, but settling for letting me borrow his laptop when mine crashed during the last couple weeks before my thesis was due; Steve Burgoon and Matt Chenworth, for their support, getting me out for late night gym sessions and intramural sports; Ryan Faul and Andrew Manuel, for their support and encouragement. Thank you also to all of my family and friends that have helped me along the road to this point in my life. I am truly blessed to be a product of such great company.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Motivation

The quadcopter, a helicopter variant with four rotors, has become widely used across many disciplines. The consumer market has been marked by increased sales of ready-to-fly quadcopters as toys as well as recreational projects by companies such as DJI and Parrot [1, 2]. Scientists and engineers around the world are using quadcopters as platforms for research involving control theory, perception, mapping, path planning, manipulation, group dynamics, and machine learning among other things [3–7]. Quadcopters have been used in filming and photography such as the shooting of New York City by Nicholas Doldinger in [8]. Lockheed Martin has developed a quadcopter for police and military use in surveillance as well as search and rescue missions [9]. An example of police use of a quadcopter for human search can be found in [10]. DHL and Amazon are developing the platforms for package deliveries [11, 12].

Each of these uses employs a control system for the quadcopter. This wide array of quadcopter use thus provides an ideal avenue to gain a more fruitful understanding of the application of control theory. The movement toward using adaptive controllers for quadcopters interests us most due to the increased use of quadcopters

for flights involving environments with uncertainties such as wind and structural failures, as well as moving and manipulating objects.

In our lab, we currently have quadcopters, computers, cameras, and software that should be able to accommodate the implementation of control systems to be studied. The hope is to develop a platform for testing and research so that feedback strategies can be used for controlled flight. It is often true that difficulties arise in determining and creating the best interfaces and uses for each component in the set-up of such a development.

The field of systems engineering relies on methods to simplify and organize the engineering design process in order to address the difficulties that arise thereby assuring the development of a valid product built in a verifiable way. Programs in universities around the nation have been created in recent times to specifically target the research and development of systems engineering methods [13–17]. Model based systems engineering is the primary method of focus for many of these program as well as the program at the University of Maryland [17]. This method focuses on developing the system through the use of computer-linked models to aid in all aspects of the design process.

1.2 State of the Art

The study of control system development for quadcopters has taken place around the world in laboratory environments such as the one we hope to develop. Stanford University's Stanford Testbed of Autonomous Rotorcraft for Multi Agent

Control (STARMAC), Massachusetts Institute of Technology's Real-Time Indoor Autonomous Vehicle Test Environment (RAVEN), University of Pennsylvania's General Robotics, Automation, Sensing, and Perception (GRASP) Lab's Micro-UAV Test Bed, and ETH Zurich's Flying Machine Arena (FMA) stand out as key examples of designed environments that lead to ease of control system development and testing [18–21]. Researchers at these institutions have tested many controllers based on ideas stemming from [3] which use PID, LQR, sliding mode control among others and then extend to newly developed control theories.

Stanford's STARMAC is designed to do waypoint navigation with quadcopters using a ground station for sending the GPS and waypoint data to each quadcopter and a computer cluster running MATLAB for control algorithm processing. The Draganfly X-4 quadcopter is used with a custom PCB for integration into their testbed. The onboard stabilization data updates at 76Hz making for great control in low vibration environments. The main difference between this platform and the following three is that it does not make use of a camera system for positional tracking data. It also operates primarily on GPS signals and thus flights take place outdoors [18].

MIT's Raven is designed to study multi-vehicle algorithms along with single vehicle autonomy. They use Draganfly V quadcopters along with a Vicon motion capture system. Each quadcopter has a dedicated ground station computer which processes current state information from the Vicon system along with mission data in order to command the quadcopters via a remote control interface. The computer is connected to the RC via a USB cable. The separation of the mission data, which

includes control algorithms, from the structure of Raven enables users to implement different controllers without requiring an entire test bed redesign. All control is done off-board using LQR control methods in the example included in [19].

UPenn's GRASP lab also uses a Vicon motion capture system along with computers to control quadcopter flights. The AscTec Hummingbird quadcopter, along with Zigbee wireless modules, fly by command from the ground stations. In this case, the on-board sensors provide data to an onboard controller for primary attitude control, and then off-board processing and controllers provide positional control for the flights [20].

ETH Zurich's FMA seems to be very similar in structure to UPenn's GRASP lab. The Vicon motion capture system, Hummingbird quadcopter base, and computers are used. It includes on-board electronics for on-board processing for attitude control, and computers which make use of wireless communication for off-board processing for positional control. State estimation occurs to help with latency concerns which makes high speed and quick maneuver flights possible. The philosophy for their communication command of the quadcopter involves sending the most updated command often to the quadcopter, which may not always reliably receive it, such that communication is unidirectional and simplified. The event of a missed command does not compromise the control of the quadcopter as a follow-up command is received so soon after [21].

1.3 Objectives

There are three objectives for this thesis. The first is to use a systems engineering approach to model and develop a quadcopter test environment and research platform by making use of the hardware and software available in the Cooperative Autonomy Lab at the University of Maryland - College Park. This includes using MATLAB and Simulink for simulations, and using an Ascending Technologies Hummingbird quadcopter and OptiTrack 6-DOF motion-capture camera system along with a computer for flight tests. MATLAB will also be used for the implementation of the controller for flight tests.

The second objective involves using developed control theory to design quadcopter controllers for flight trajectory tracking. A linear quadratic regulator (LQR) controller will be developed first as a baseline for comparison with a model reference adaptive controller (MRAC). The simulated flights with the adaptive controller will be compared to flights with the baseline LQR controller to determine margins of improvement of using the adaptive controller over the baseline LQR controller.

Future usability of the validated system through accurate, understandable documentation is the final objective of the work in this thesis. The goal for future students to design their own controllers and to easily use or modify the system to their needs will depend on the success of this thesis in providing a working product, which has been validated through the implementation of an LQR controller for flight tests, and clear documentation.

1.4 Outline of Thesis

Chapter 1 sets up the motivation for the work and covers state of the art for several quadcopter test environments. It continues with a section on the objectives of this thesis as well as an overview of the chapters.

Chapter 2 discusses background information related to systems engineering and the process of model based systems engineering. It also discusses quadcopter modeling and dynamics, specifically as it is related to the AscTec Hummingbird for development in Simulink.

Chapter 3 covers the systems engineering of the Quadcopter Test Environment and Research Platform (TERP) from needs, behavioral diagrams, structural diagrams, and requirements to implementation, verification and validation. Behavioral diagrams consist of use case diagrams and the activity diagram. Structural diagrams consist of system context as well as component and system architecture diagrams.

Chapter 4 guides the development of several flight controllers. The controllers are implemented and simulated in MATLAB Simulink. The chapter continues by showcasing the implementation of the LQR controller with the TERP System for actual flight tests in order to complete validation.

Chapter 5 provides a conclusion to the thesis as well as suggestions for future work.

Chapter 2: Background

2.1 Systems Engineering

Systems engineering is the intersection of engineering design and business management. Engineering design involves defining and solving a problem through the creation of an engineered solution. Business management focuses instead on the organization of processes and resources for the creation of a service or product [22]. The intersection occurs when a business carries out engineering design to include the full life cycle of the product from idea to disposal. Companies and researchers alike make use of systems engineering principles in hopes of increasing productivity and efficiency by acting as a liaison and creating a standard cross-disciplinary nomenclature [22].

Traditionally, document-based systems engineering involves keeping documents tracking the engineering design process up-to-date for all those involved [23]. Errors, specifically inconsistencies, in documents can arise, when document-based systems engineering occurs with members of different disciplines and in different buildings or even different countries. Advances in system complexity call for increased documentation without good methods for cross-referencing error detection [24].

Computers and system complexity led to the breakthrough of model-based

systems engineering (MBSE). MBSE involves the creation and upkeep of system models to track the design process that can be accessed by those involved in order to manage the development of the system. An accessible live model links all disciplines by a common structure and thereby improves communication among people from different areas of focus, possibly in locations around the world. The models created through the model- based approach are easily linked and cross-referenced such that inconsistencies are minimized in comparison to the document-based approach [23].

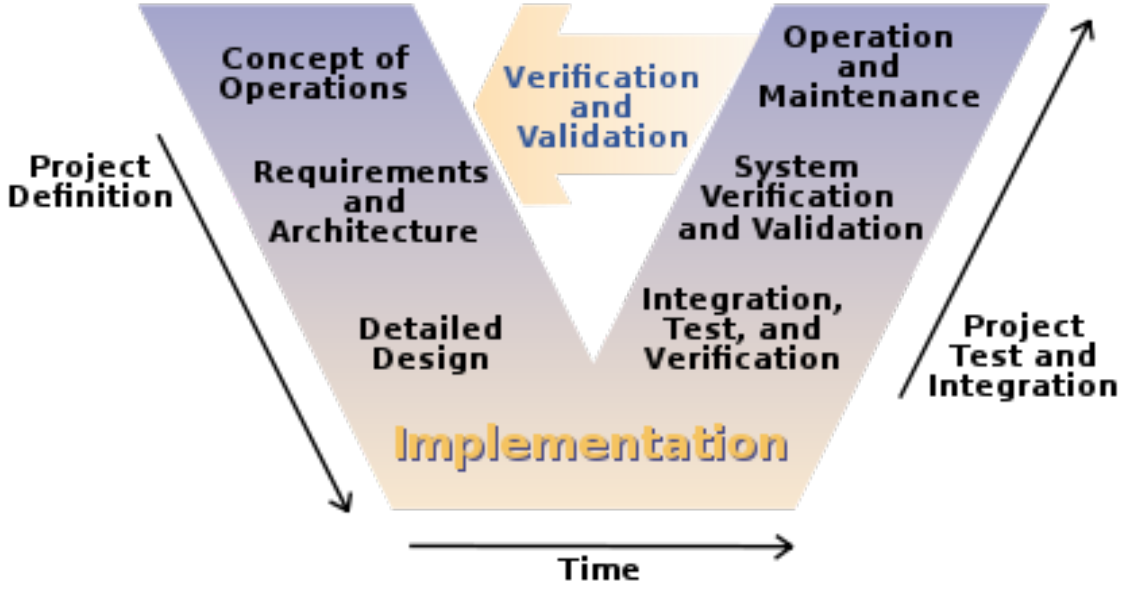

Figure 2.1: V-Model of System Development. [28]

The V-Model of system development in fig.2.1 describes the method of MBSE employed in this thesis. The goal of this model is to describe the process which leads engineers through the system development cycle. The project definition guides from concept to final design. Design implementation can then occur. This includes hardware construction and software coding. Component, subsystem, and system

testing and integration then takes place in order to verify and validate the system.

An accurate description of the system needs from stakeholders must be acquired in order to begin the development of the concept of operation. These needs are then supplemented with an analysis of the system domain and available component overviews. The system domain is a representation of the environment in which the system will operate. Component overviews include functional descriptions along with ports which enable interfacing possibilities. These overviews are modeled through use of blocks in block definition diagrams, and parts, represented as blocks as well, are assigned in hierarchical fashion to owner blocks.

The concept of operation can be modeled through use case development and further through representation of the use cases in activity diagrams. A use case diagram enables visualization of system boundaries, actors on the system, inclusions, and extensions. The activity diagrams then represent the interactions between the actors and system, and even possibly between subsystems that are described within the use cases. This additional visualization enables use case narrative composition in which the implied design requirements can be first defined. Use cases and activities are classified as behavioral models [25].

Design requirements can be categorized into two groups: functional and performance. Functional requirements describe intended operations or constraints and performance requirements describe the measures at which operations shall occur. There are three levels of requirements: system, subsystem, and component. Through architecting, one can develop requirements for each level. System context, or system level design, is the first architecture that synthesizes from the behavioral models.

The system context includes inputs, outputs, mechanisms, controls, and parts of the system which help in the development of system-level requirements. Mechanisms are items or structures which enable operation of the system. Controls are typically constraints to the system.

In a similar fashion, the development of the subsystem architecture leads to subsystem requirements, and the development of component architecture leads to component requirements. Architecting is the process of designing interactions between parts of the level being created through use of ports by considering the functionalities of those parts. Internal block diagrams are used to model architectures of systems, subsystems, and components. These internal block diagrams are linked to blocks from the block diagram which contain the parts and ports used [26].

Requirements development and architecting at any level can cause changes in requirements and architecture at other levels. Detailed design which occurs at the lowest level based on all requirements can also cause engineers to see the necessity to modify architectures and requirements at higher levels. Detailed design may also require the creation of components or parts that do not yet exist. Project definition can be a highly iterative process [27].

It is also true that verification and validation can cause engineers to iterate the project definition process as well based on results of failed tests. Verification involves constructing tests for requirements. Validation involves constructing tests showing successful functional integration of components into subsystems, functional integration of subsystems into the system, and functional capabilities of the system to the customer.

What results from using MBSE is a verified and validated system along with models which can be used for reproduction. These models can also be used to help engineers understand how to operate, troubleshoot, or upgrade the system. The clear trail of documentation left through MBSE provides these benefits to engineers as well as system users.

2.2 Quadcopter Modeling

The quadcopter structurally consists of a '+' shaped frame with motors at the end of each leg. Attached to each of the four motors is a propellor. The motor-propellor assembly generates a lift force, and torque on the quadcopter. Additionally there are effects of blade flapping, and gyroscopic forces that exist due to the structure of the quadcopter. The motor-propellor assemblies alternate spin direction to balance the effect of torque generated on the frame. Figure 2.2 shows the structure, reference frame orientations, and forces and moments which we consider to act on the quadcopter.

The typical operation of the quadcopter involves changing the collective thrust for moving in the z direction. Changing the difference in opposing motors changes the roll and pitch of the quadcopter. Yaw can be changed by changes in the collective torque on the quadcopter due to the motors. Changes in x and y are made through thrust vectoring in which thrust is used along with pitch and roll changes to move the quadcopter in those translational directions.

Assumptions made in the development of this model and the development of

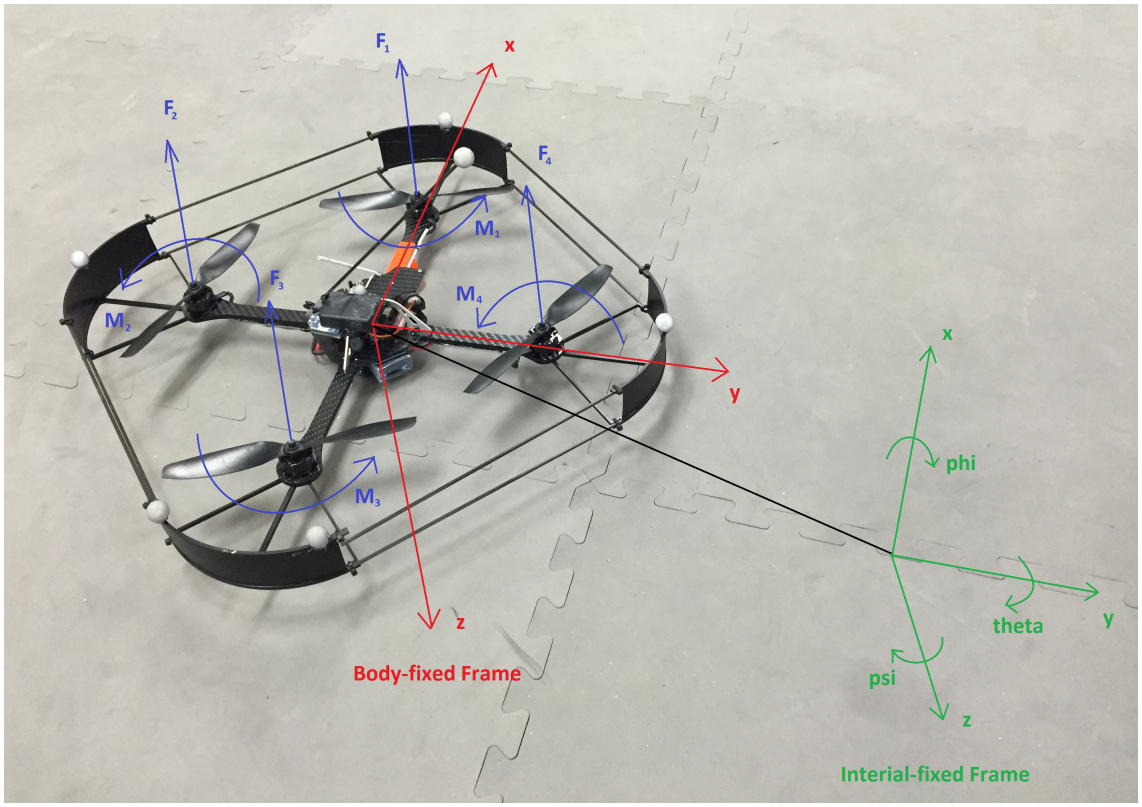

Figure 2.2: Quadcopter Orientation Diagram.

the controllers following in this thesis are the following which are similar to those in [3, 30–32]. The body center of gravity coincides with the origin of the body fixed frame. The structure of the quadcopter is symmetrical and rigid. The propellors are also assumed to be rigid. Additionally, the effects of blade flapping and propellor gyroscopic forces are to be ignored due to low speed operations of the quad as well as the assumption of small angle flights made similar to [29–31].

Newton-Euler formulation of the quadcopter model used in this thesis follows [31, 32] closely with the modification such that the orientation of the inertial-fixed and body-fixed frames are the aerospace standard, north-east-down (NED), as in

fig. 2.2. The Newton equation in the inertial-fixed frame is

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + {}^{I}R_{B} \begin{bmatrix} 0 \\ 0 \\ -\sum_{i=1}^{4} F_i \end{bmatrix} \tag{2.1}$$

where m is the quadcopter mass, g is acceleration due to gravity, F_i is the lift force due to motor i in the z direction of the body-fixed frame, r is the vector of x, y, and z position in the inertial-fixed frame, and ${}^{I}R_{B}$ corresponds to the rotation matrix from the body-fixed frame to the inertial-fixed frame. This rotation is derived from the X-Y-Z Euler angle formulation and has the following equation as in [32] with c_x and s_x corresponding to $cos(x)$ and $sin(x)$, respectively. The angles ϕ, θ, and ψ correspond to the roll, pitch, and yaw of the aircraft, respectively.

$$\begin{aligned}
{}^{I}R_{B} &= R_z(\psi)R_y(\theta)R_x(\phi) \\
{}^{I}R_{B} &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \\
{}^{I}R_{B} &= \begin{bmatrix} c_\psi c_\theta & c_\phi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}
\end{aligned}$$

The Euler equation in the body-fixed frame is

$$J\dot{\omega} = M - M_g - w \times J\omega \tag{2.2}$$

where J is the inertial tensor of the quadcopter around its center of mass, ω is the vector of ω_x, ω_y, and ω_z angular velocities from the body-fixed frame, M is

the matrix of moments due to motor torques, M_g is the matrix of moments due to propellor gyroscopic effects.

Neither equation above takes into account the effects of blade flapping. By our assumptions of symmetry and low speed operation, we ignore the effect of M_g and are left with a simplified

$$J = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}$$

and

$$M = \begin{bmatrix} l(F_2 - F_4) \\ l(F_1 - F_3) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix}$$

where J_x, J_y, J_z are the quadcopter moments of inertia about the x, y, and z axes of the body-fixed frame, l is the distance from the center of the propellor to the center of mass of the quadcopter, and F_is and M_is correspond to the forces and moments, respectively, due to individual motor thrusts 1-4 in the body-fixed frame.

These motors have force and moment relations to angular speed according to the equations $F = b\omega^2$ and $M = d\omega^2$, where ω is the motor speed in rotations per minute [32]. A conversion from u to specific individual motor speeds can then be carried out in order to control the actual quadcopter. Constants $b = 6.11 \times 10^{-8} \frac{N}{rpm^2}$ and $d = 1.5 \times 10^{-9} \frac{Nm}{rpm^2}$ from [31] are used for the Hummingbird quadcopter.

Additionally, we use the relation between angular velocities and the inertial-

fixed frame Euler angles as in [32] to arrive at the simplified nonlinear model

$$
\begin{aligned}
m\ddot{x} &= -(c_\psi s_\theta c_\phi + s_\psi s_\phi)u_1 \\
m\ddot{y} &= -(s_\psi s_\theta c_\phi - c_\psi s_\phi)u_1 \\
m\ddot{z} &= -c_\theta c_\phi u_1 + mg \\
J_x\dot{\omega_x} &= \omega_y\omega_z(J_y - J_z) + lu_2 \\
J_y\dot{\omega_y} &= \omega_x\omega_z(J_z - J_x) + lu_3 \\
J_z\dot{\omega_z} &= \omega_x\omega_y(J_x - J_y) + u_4 \\
\dot{\phi} &= \omega_x + s_\phi t_\theta\omega_y + c_\phi t_\theta\omega_z \\
\dot{\theta} &= c_\phi\omega_y - s_\phi\omega_z \\
\dot{\psi} &= \tfrac{s_\phi}{c_\theta}\omega_y + \tfrac{c_\phi}{c_\theta}\omega_z
\end{aligned}
\tag{2.3}
$$

with $u = [u_1\ u_2\ u_3\ u_4]^T$ where $u_1 = F_1 + F_2 + F_3 + F_4$, $u_2 = F_2 - F_4$, $u_3 = F_1 - F_3$, and $u_4 = -M_1 + M_2 - M_3 + M_4$.

The model described by (2.3) is then implemented in Simulink as in fig. 2.3 with intertial parameters for the AscTec Hummingbird quadcopter taken from [33] and our measured mass which are

$$
\begin{aligned}
&m = 0.668kg : \text{ mass of the Hummingbird quadcopter with battery,} \\
&J_x = 0.0039kg * m^2 : \text{ moment about x axis of Hummingbird quadcopter,} \\
&J_y = 0.0039kg * m^2 : \text{ moment about y axis of Hummingbird quadcopter, and} \\
&J_z = 0.0049kg * m^2 : \text{ moment about z axis of Hummingbird quadcopter.}
\end{aligned}
\tag{2.4}
$$

The model input is u and the output includes x, y, z, ϕ, θ, ψ, as well as the velocities of each in the inertial-fixed frame of reference.

Further, due to low speed flight and small angle approximation assumptions,

the nonlinear model in (2.3) can be simplified to

$$\begin{aligned}
\ddot{x} &= -g\theta \\
\ddot{y} &= g\phi \\
\ddot{z} &= -\tfrac{1}{m}\triangle u_1 \\
\dot{\phi} &= \tfrac{l}{J_x}u_2 \\
\dot{\theta} &= \tfrac{l}{J_y}u_3 \\
\dot{\psi} &= \tfrac{1}{J_z}u_4
\end{aligned} \tag{2.5}$$

by letting $u_1 = mg + \triangle u_1$ in a fashion similar to [30, 31]. This yields a linear model which will serve as a base for developing flight controllers in Chapter 4.

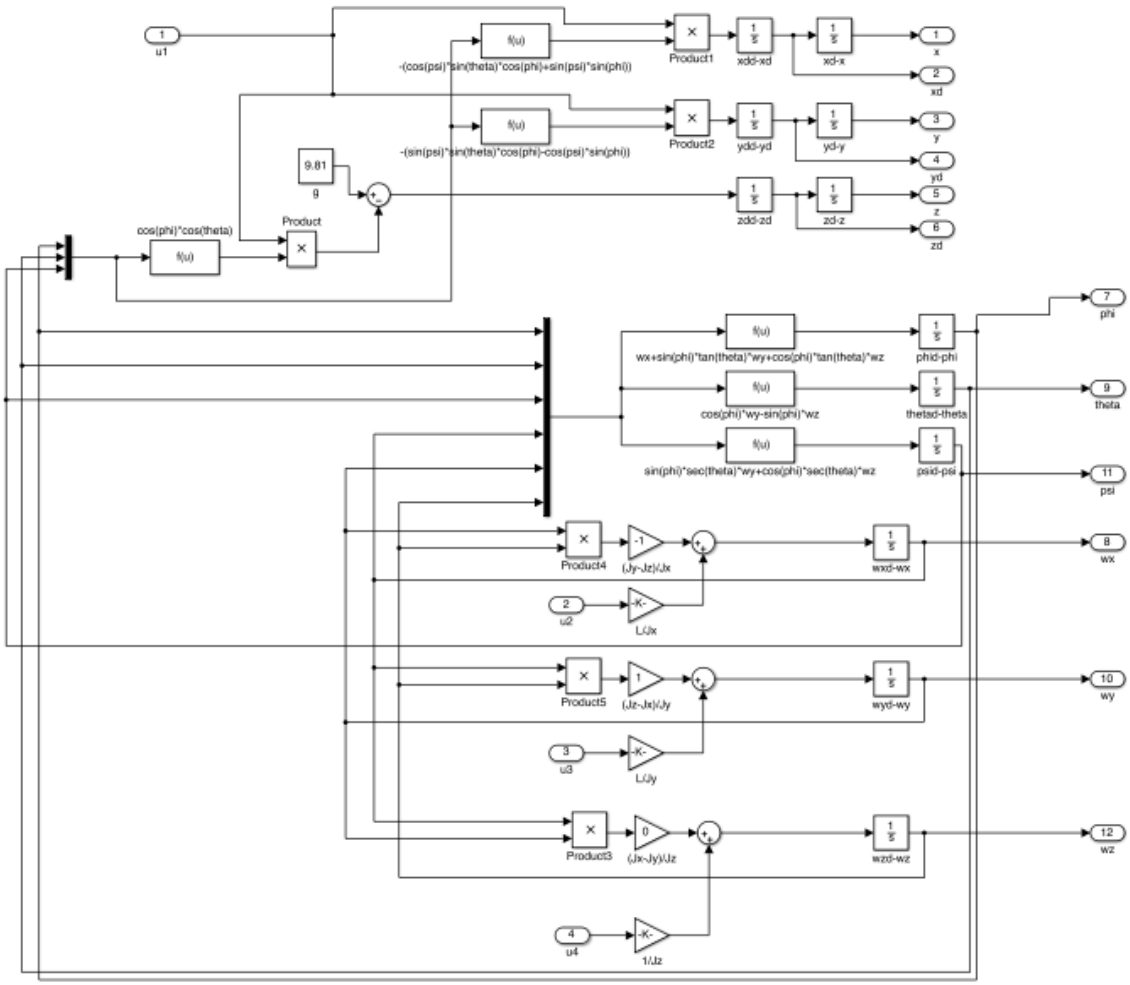

Figure 2.3: Simulink Model of the Hummingbird Quadcopter.

Chapter 3: Quadcopter T.E.R.P. System Design

3.1 Description of Needs

The stakeholders in this project are future students, Dr. Nuno Martins, spectators, and current students. In speaking with current students and Dr. Martins, and considering the viewpoint of future students and spectators, we developed a set of needs for the system.

- Design a testbed for quadcopter flights using available lab space.

- Be able to implement custom controllers.

- Use Hummingbird quadcopter with the ACI communication protocol.

- Use OptiTrack camera system with Motive software.

- Use Windows PC in the lab.

- All testing and flights need to be safe.

- Use MATLAB for controller implementation.

- Be able to mount a payload on the quadcopter.

- Be able to fly the quadcopter with the TC7 remote control.

This set of needs forms the informal basis for the design and development of the Quadcopter Test Environment and Research Platform (TERP) that transpires over the course of this chapter.

3.2 System Domain and Component Overview

The TERP System exists as a part of the lab domain, see fig. 3.1. The lab domain also includes the operator, the controller, and the payload. The physical environment for the quadcopter flight is left out in this case as it is assumed that the flight space is an inner part of TERP. The air and visual provisions are thereby provided by that flight space.

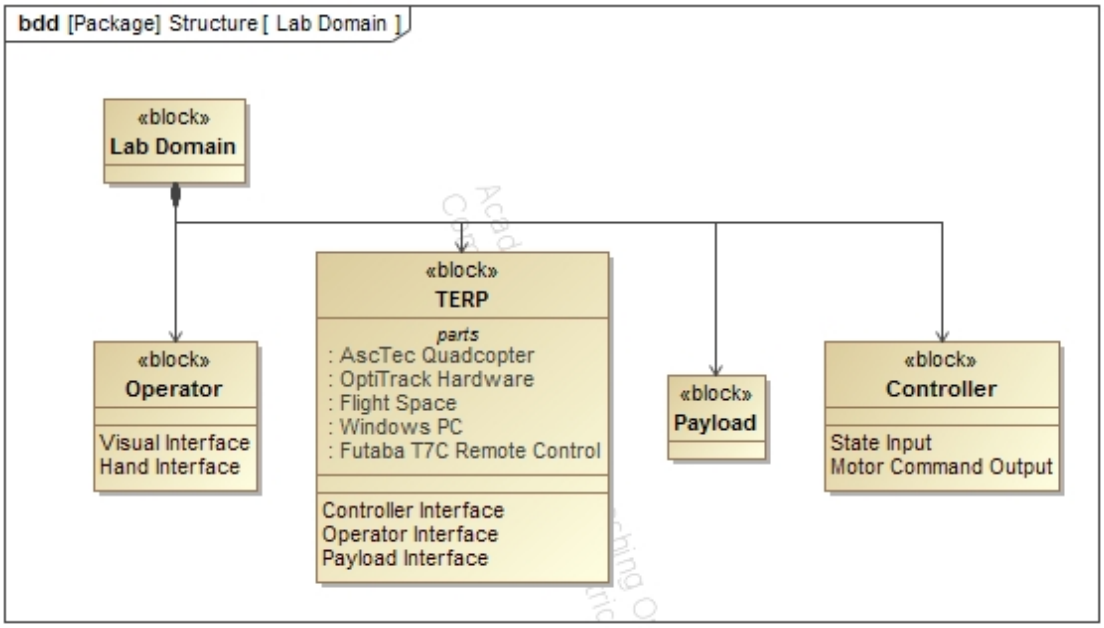

Figure 3.1: TERP System domain diagram.

Five main parts compose the TERP System, see fig. 3.2. Both hardware and software comprise the currently available useable parts which we model in order better understand how they can be used to create respective interactions in the development of the system architecture. The quadcopter and PC are represented using block diagrams as they have components which we will need to use for sys-

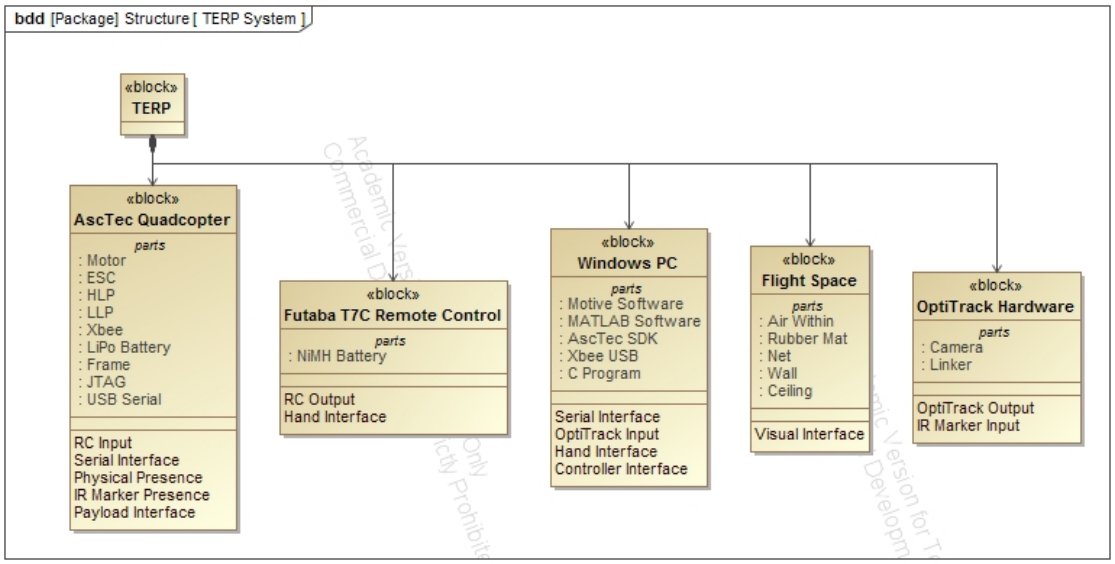

Figure 3.2: TERP System block diagram.

tem integration. There is no need to decompose the RC block as the Futaba TC7 remote control integration is completed as a part of the Ascending Technologies Hummingbird package. The flight space solely gives visual feedback, protection, and size constraints, therefore there is no need to further decompose the block. The OptiTrack camera system is also already integrated thus we simply model it as a part of the TERP System in order to show where positional data for the Motive software originates.

The Quadcopter to be used is the Hummingbird from Ascending Technologies (AscTec). A motor and electronic speed controller (ESC) is mounted on each of the four carbon fiber arms of the quadcopter frame. The arms are joined by an aluminum central body which houses the electronics on carbon fiber plates. The arms, central body, and plates form the frame of the quadcopter and provide the physical presence as well as the payload interface. Infrared marker presence is also

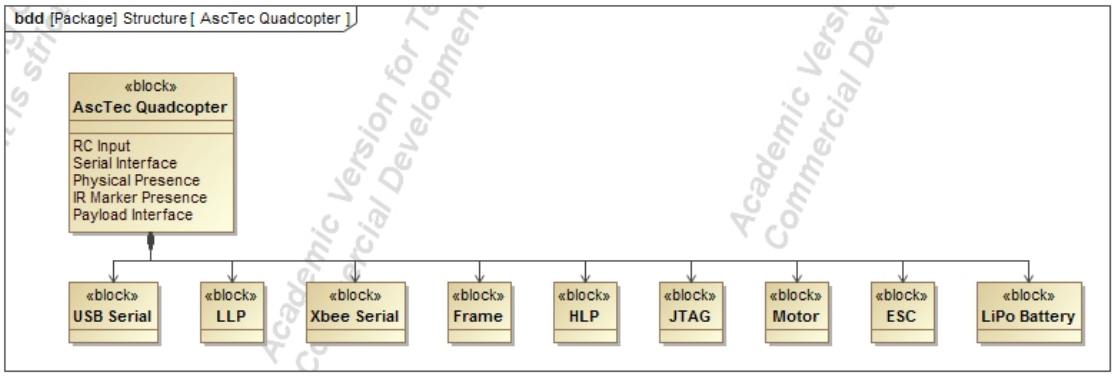

Figure 3.3: TERP System quadcopter block diagram.

based on placement on the quadcopter frame.

Serial communication ports are available wired through USB or the JTAG and wirelessly over XBEE. The wireless serial interface can be used with the ACI protocol to command motor speeds. HLP and LLP are the high level processor and low level processor respectively. The LLP controls communication with all sensors and motors. The HLP is available to be programmed via the JTAG and can run custom code which can instruct the LLP to generate motor commands and provide sensor data to the HLP. Control of XBEE communication is also part of the HLP. The processors run and update at a rate of 1kHz. A 2100mAh lithium-polymer (LiPo) battery powers the quadcopter [34].

The Futuba T7C remote control integration with the Hummingbird quadcopter includes six channels of use as well as some emergency protocols. Four channels used by the two joysticks control throttle, yaw, pitch, and roll as in fig. 3.4. The top left switch controls modes of operation and should remain in the position furthest away for manual control. More can be read about height control and GPS

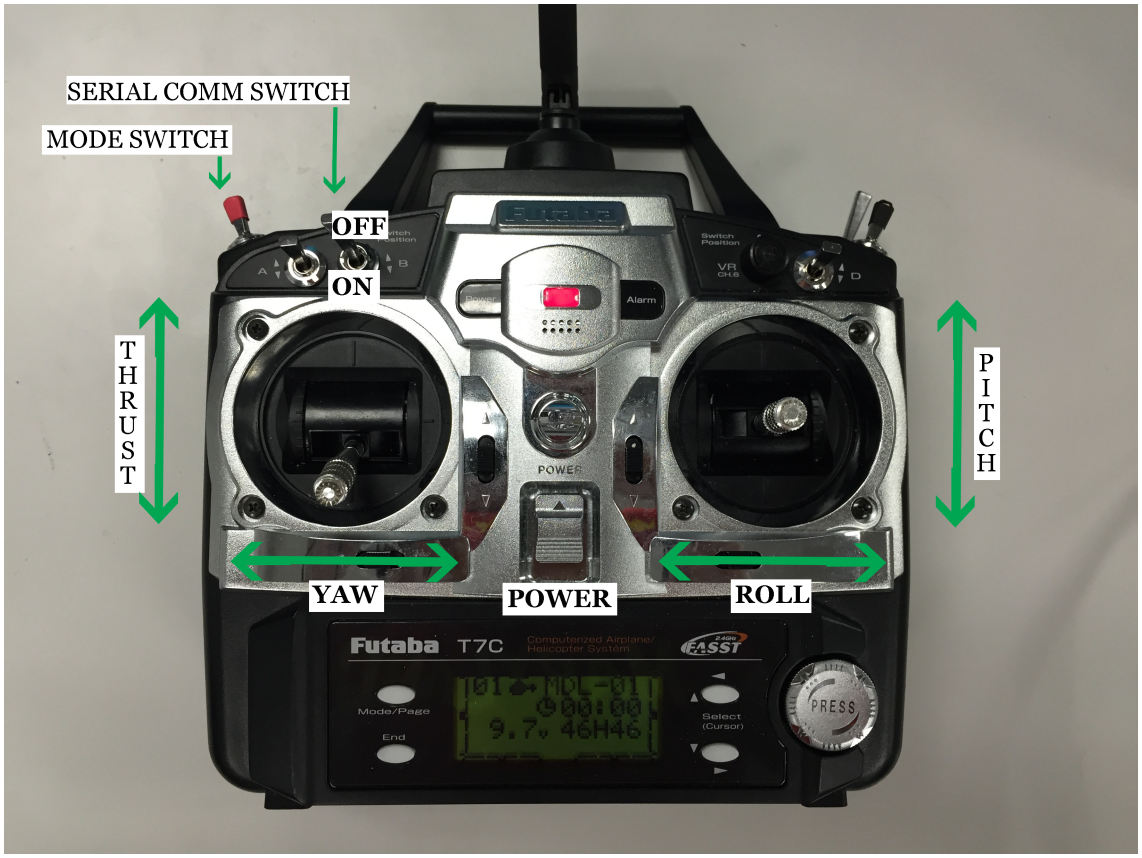

Figure 3.4: Futaba T7C Remote Control label with quadcopter aspects of control.

control modes in [34]. Switch B enables control of the LLP by the HLP which includes when the HLP receives wireless commands.

The integrated emergency protocol results in the quadcopter performing immediate stabilization based on the AscTec base controller built into the LLP when the signal from the remote control is lost. Communication from the remote control is done through a receiver on the quadcopter that is connected to the HLP. The receiver is not modeled in the system, but more information including what will be used to sync the remote control can be found at [35].

We use a Windows 7 PC with MATLAB, Motive and C program running capa-

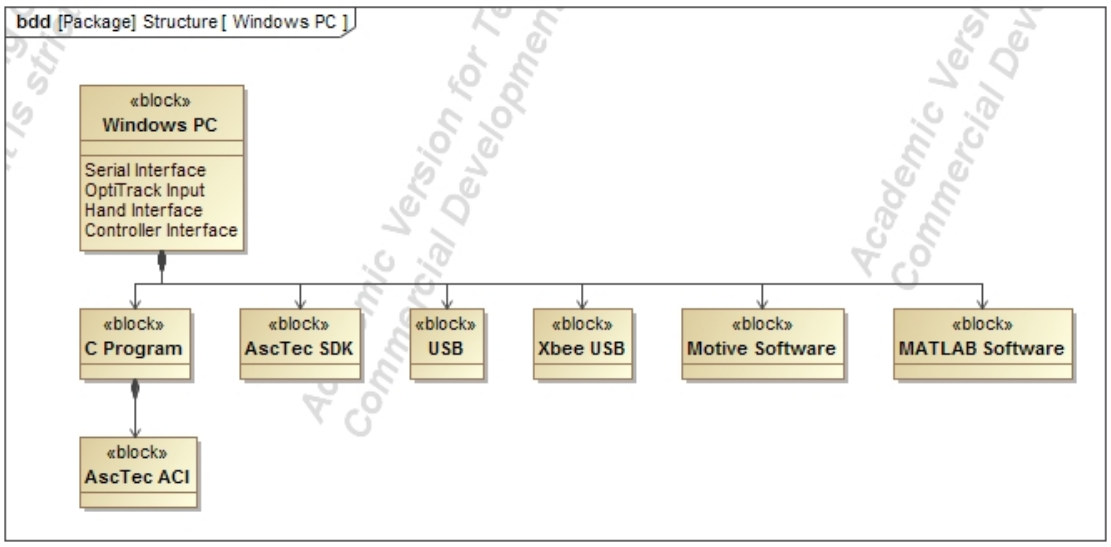

Figure 3.5: TERP System PC block diagram.

bilities. MATLAB software can be used to simulate controllers with the quadcopter model from chapter 2 in Simulink. MATLAB can also be used in conjunction with Motive to receive 6-DOF positional data from the quadcopter in the flight space. Motive processes camera data and generates 6-DOF data based on IR markers on the quadcopter. The PC also has serial communication capabilities wired through USB or wirelessly through XBEE USB. Serial communication with the quadcopter must adhere to the ACI communication protocol which is written in the C programming language.

The flight space is an enclosure in the lab with dimensions: 12ft x 18ft x 9ft, with two walls, a ceiling, two netted sides, and a matted floor serving as boundaries. The netted side which divides occupied lab space from the flight space includes doubled golf netting securely mounted by railings above and by eyebolts on each wall. The additional safety measures taken in protecting users and spectators from

dangers of quadcopter flights led to this double net boundary.

The camera system hardware, fig. 3.6, by OptiTrack consists of twelve cameras and two hubs connected via USB cables and an RCA cable link. Raw camera data is transmitted to the Motive software on the PC via the USB connection from the hubs to the PC. These inner connections are not modeled and more information can be found in [36]. Cameras can be organized in different orientations to create a visualized space in which shapes can be tracked using Motive software to process data based on the IR markers configured in a fixed shape. Calibration of the cameras for the space is done using Motive with an IR wand and coordinate system marker.

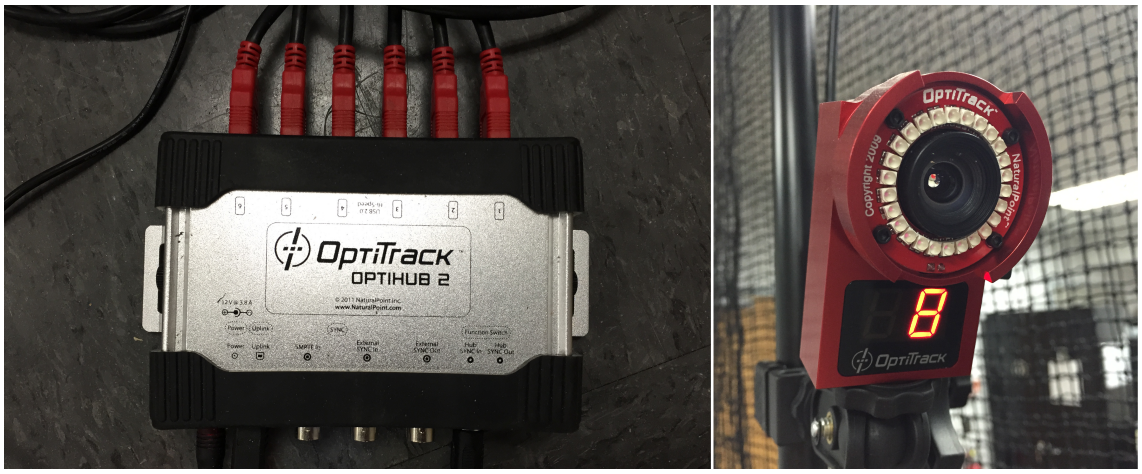

Figure 3.6: OptiTrack OptiHub 2 and Flex V100 R2 Camera

3.3 Use Cases and Activity Diagram

The use case diagram in fig. 3.7 includes the primary use case of 'Run Controller' as well as the case for loading the controller. The use case narratives which explain each of these in detail are in Appendix A. The activity diagram in fig. 3.8

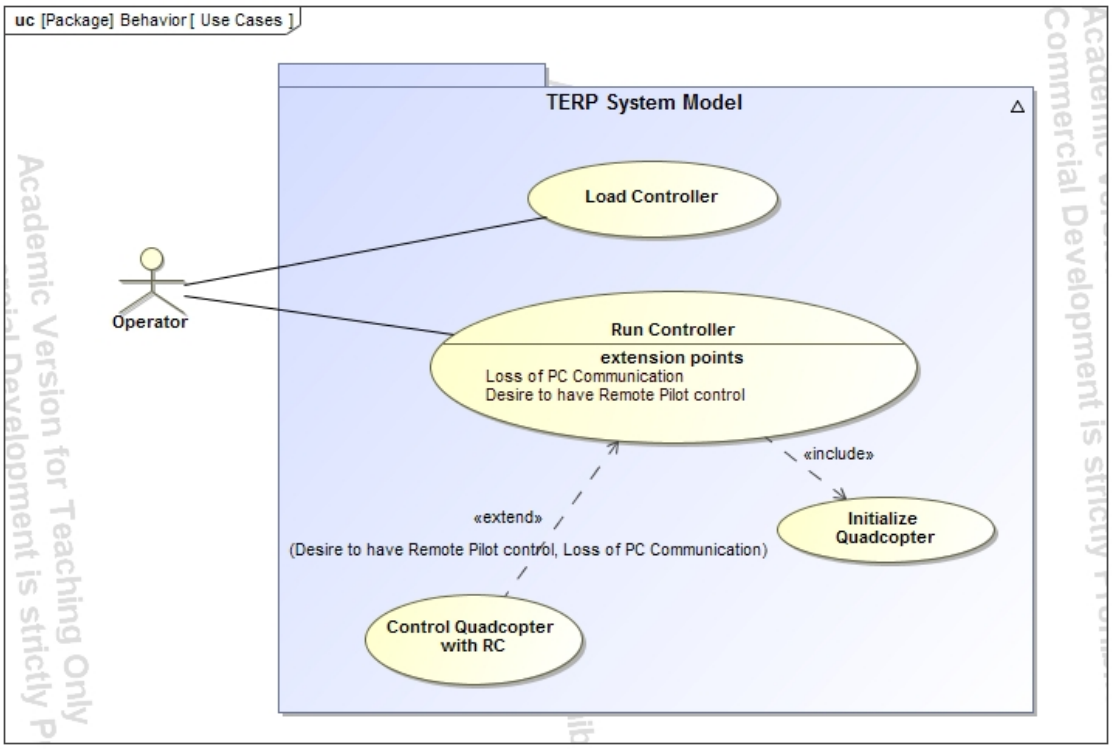

Figure 3.7: TERP System use case diagram.

enables a joint visualization of the TERP System use cases.

Each of the use cases starts with the system in an idle, but completely calibrated and operational state. The loading of the controller consists simply of the operator inserting a custom controller into the space of the system available for controller implementation.

The running of the controller consists of an operator using the system to process OptiTrack data, to send the data to the controller, and then to take the controller's output and send motor commands to the quadcopter from the PC.

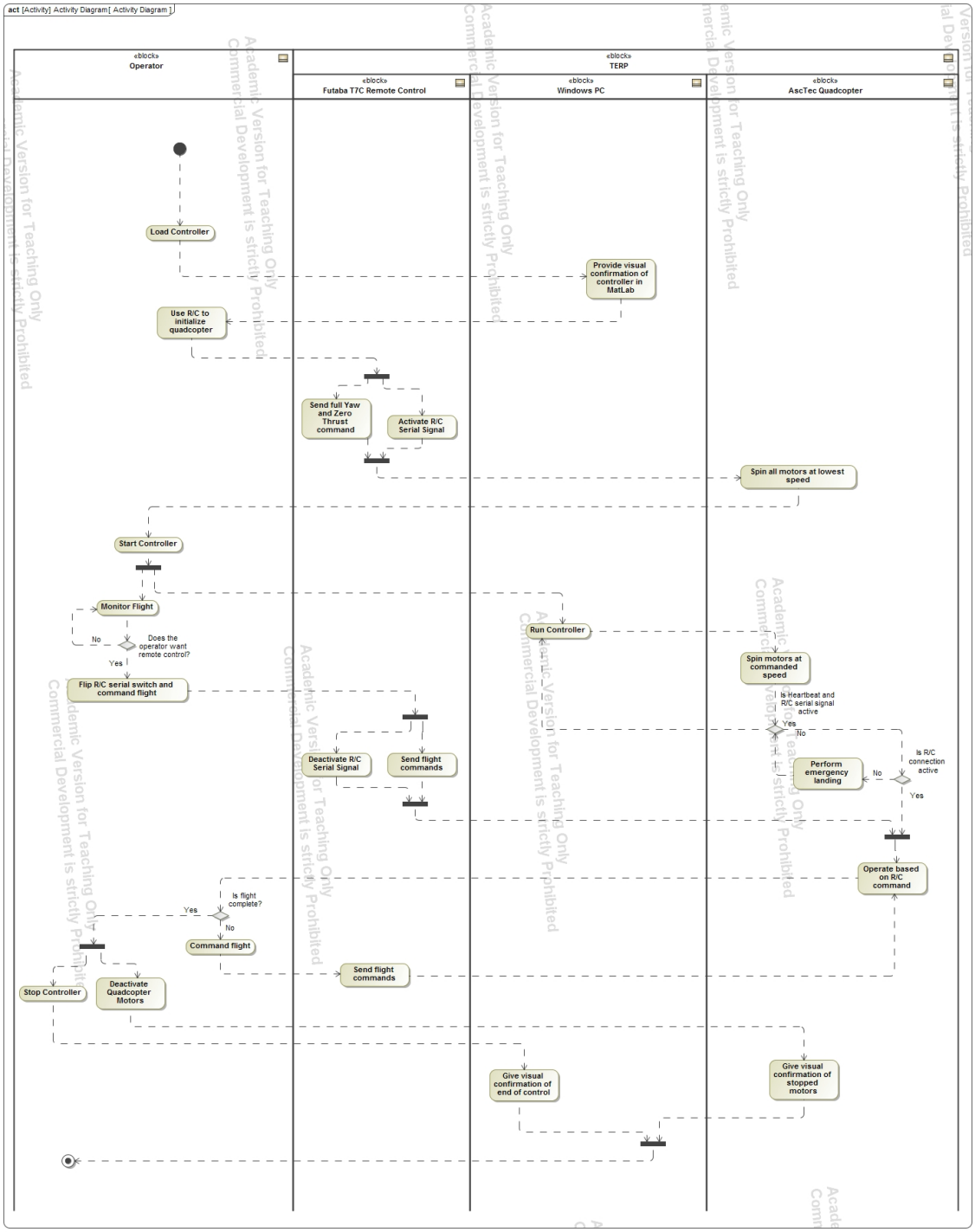

Figure 3.8: TERP System activity diagram.

3.4 Requirements and Architecture

The careful inspection of use cases and the developed activity diagram yields an architecture for interactions of parts of the lab domain of which the system belongs, see fig. 3.9. Each of those are then used to develop a set of system level requirements as well as preliminary sets of requirements for some of the subsystems and components. The set of system level requirements in fig. 3.10 leads to the system architecture in fig. 3.11 which describes the interaction interfaces between the Windows PC, OptiTrack Hardware, Flight Space, AscTec Quadcopter, and Futaba T7C Remote Control as subsystems and components of the TERP System.

The Windows PC and AscTec Quadcopter are represented as subsystems within TERP to facilitate the implementation of interfaces within them and the Futaba T7C Remote Control, OptiTrack Hardware, and Flight Space will be considered as components for the purpose of this system.

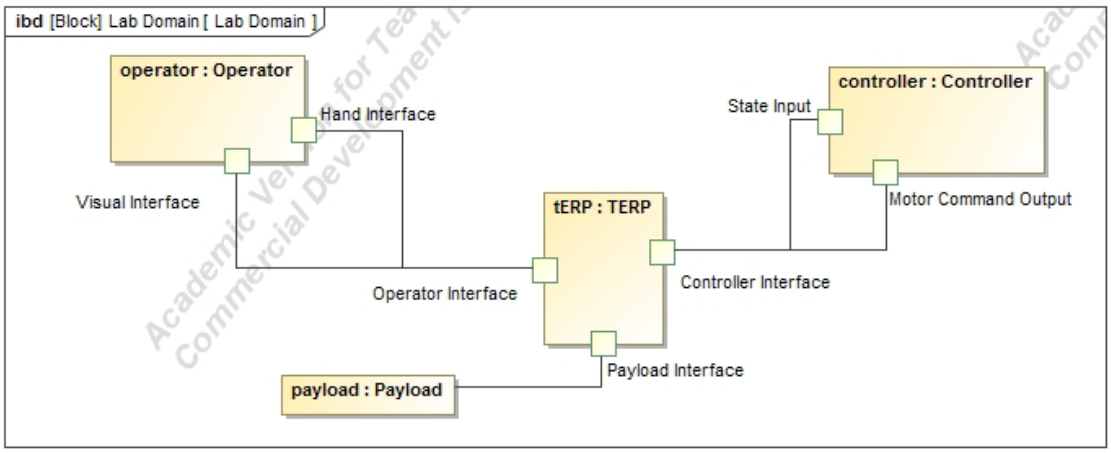

Figure 3.9: Lab domain architecture.

#	Id	Name	Text
1	1	Flight Space	System shall provide a flight space within the dimensions of the lab.
2	2	Quadcopter	System shall use AscTec Hummingbird quadcopter for flights.
3	2.4	Heartbeat	System shall send heartbeat command at a minimum rate of 10Hz during controller run.
4	3	Remote Control	System shall accept operator quadcopter commands via RC.
5	3.1	RC Hardware	System shall use Futaba TC7 Remote Control.
6	4	Controller	System shall run custom flight controller.
7	4.1	Windows 7 PC	System shall use Windows 7 PC in the lab to run controller.
8	5	OptiTrack	System shall use OptiTrack System for providing quadcopter state information to the controller.
9	6	Controller Command	System shall accept controller quadcopter motor commands.
10	6.1	Controller to PC	System shall accept motor commands on Windows 7 PC.

Figure 3.10: TERP System level requirements.

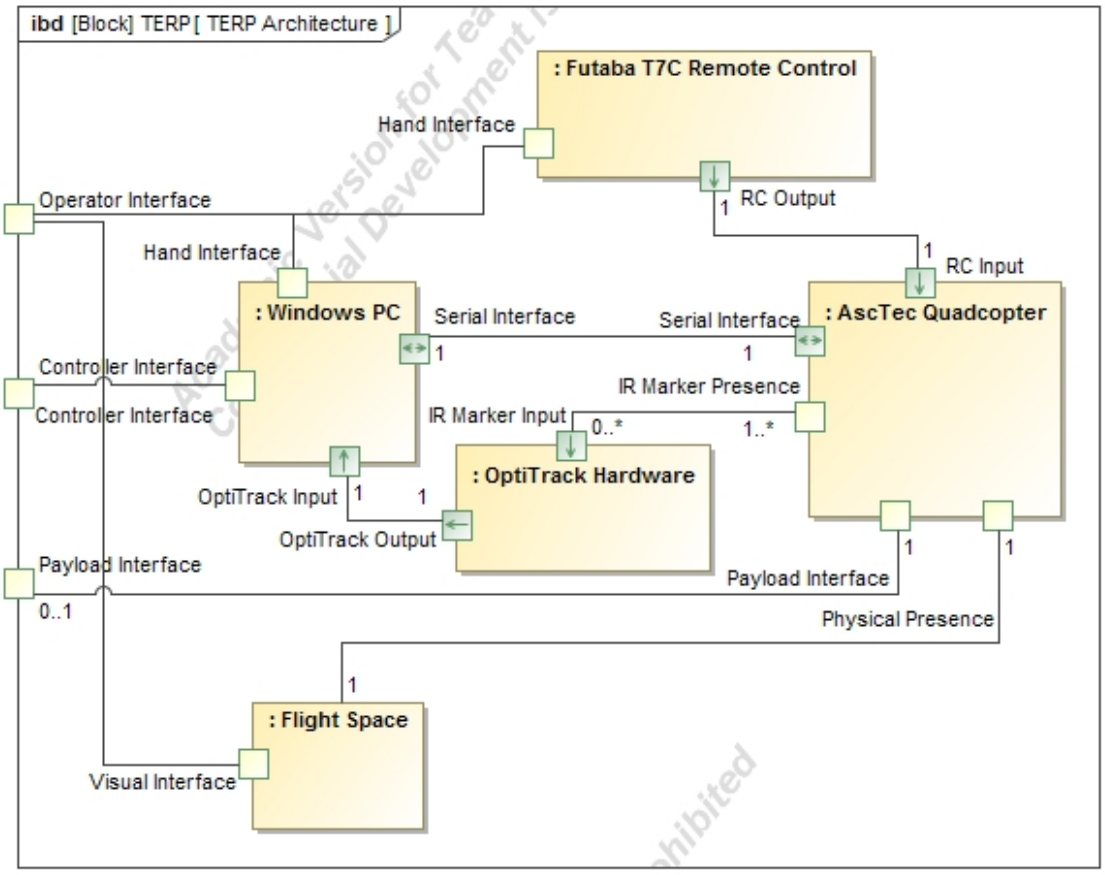

Figure 3.11: TERP System architecture.

Based on the preliminary requirements, and the understanding of the quadcopter architecture as modeled in fig. 3.12, additional requirements are developed for successful integration of the quadcopter into the TERP System. All quadcopter specific requirements are listed in fig. 3.13. The AscTec Communication Interface

protocol will be used to perform wireless communication with the quadcopter. From the architecture, it can also be seen how the relationship between the LLP and HLP requires the use of the serial communication control switch activation for the HLP commands to be accepted as valid for motor control by the LLP. The USB serial is modeled as part of the quadcopter architecture for the purpose of showing that direct communication can be made with the LLP of the quadcopter when connected using the AscTec Control Pilot Tool or Research Update Tool for monitoring data or calibrating the quadcopter and remote control based on instructions in [34]. The ESC, Battery, and Motor illustrate motor command to movement translation.

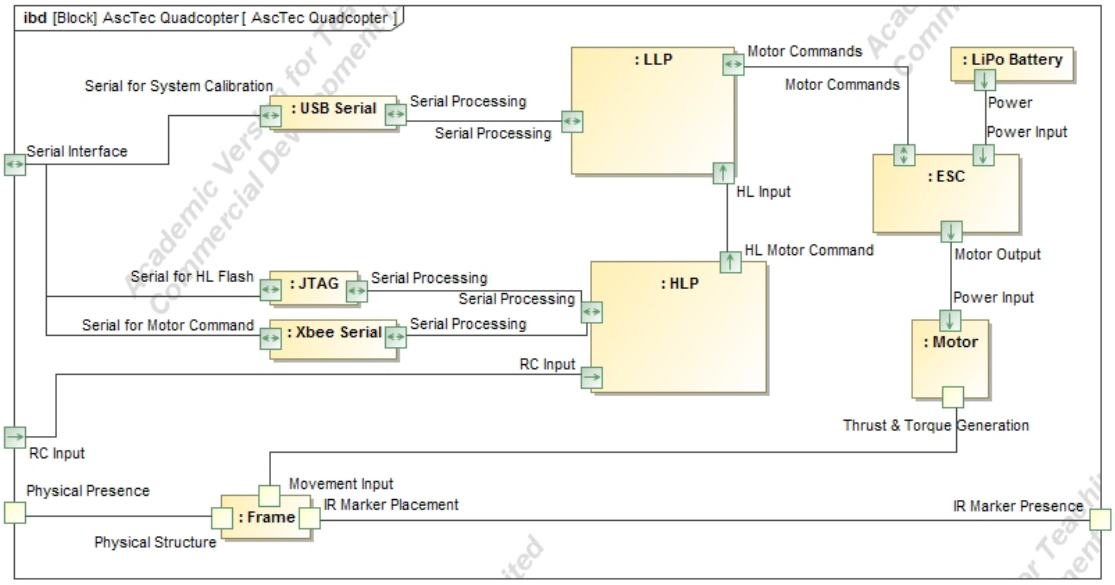

Figure 3.12: Quadcopter architecture.

#	Id	Name	Text
1	2.1	RC Quad	Quadcopter shall operate by commands via RC.
2	2.2	Payload	Quadcopter shall have payload mount.
3	2.3	Wireless Quad	Quadcopter shall operate by commands via XBee.
4	2.4	Heartbeat	System shall send heartbeat command at a minimum rate of 10Hz during controller run.

Figure 3.13: Quadcopter requirements.

Similarly is done for the PC which is modeled in fig. 3.14 with requirements in fig. 3.15. Serial interfaces are illustrated for connections to the various methods on the quadcopter. Most importantly, is the chain of programs which dictate the operation of flights. Motive processes positional data, MATLAB uses Motive information and the loaded control to determine motor commands which are sent wirelessly using the C program.

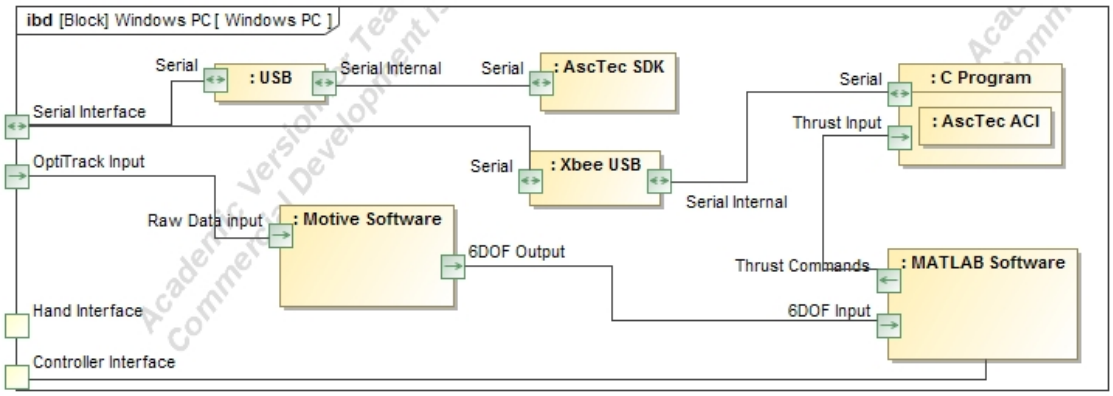

Figure 3.14: PC architecture.

#	Id	Name	Text
1	4.1	Windows 7 PC	System shall use Windows 7 PC in the lab to run controller.
2	4.2	Controller Build	Controller shall run on Windows 7 PC platform.
3	5.3	Motive Software	Motive Software shall be used on the Windows 7 PC to process camera data.
4	6.1	Controller to PC	System shall accept motor commands on Windows 7 PC.
5	6.3	Wireless PC	PC shall send quadcopter motor commands via Xbee.
6	6.4	Command Rate	PC shall send commands at a minimum rate of 80Hz
7	6.5	AscTec ACI	Ascending Technologies Communication Interface shall be used for sending commands from the PC to the quadcopter via Xbee.

Figure 3.15: PC requirements.

Additional requirements are developed for the components of the system and a full list will be referenced when system verification is completed in section 3.6.

3.5 Implementation

Implementation of the system involves making use of integration capabilities of the subsystems and developing interfaces for integration of subsystems. This section will outline the process and steps taken for full system implementation.

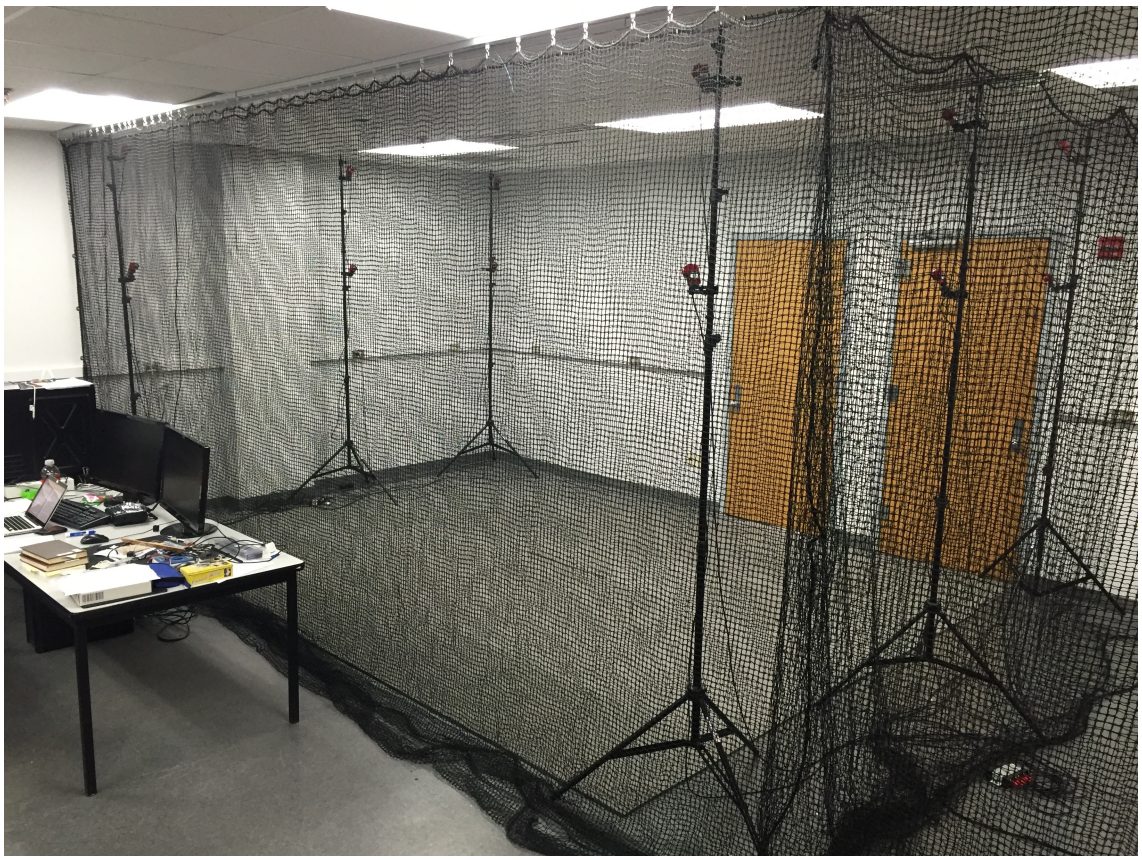

Figure 3.16: Flight Space picture taken from bottom left corner shows the PC used on the left and camera system setup on tripods in the lab.

Construction of the flight space, fig. 3.16, occurs first in a 15×25ft section of the Cooperative Autonomy Lab. Rubber mats measuring 12×16ft are placed on the floor in the far right corner of the lab. Two nets are mounted 6 inches apart and stretch along the left edge of the flight space. They are hooked along the back wall

to secure the area. A secondary net is placed on the bottom edge of the flight space, 7 feet from the wall, to enclose the area covered by rubber mats. Entry into the flight space can be made through the double nets by passing next to the near wall.

The Windows 7 PC provided includes two monitors, a keyboard, and mouse. These are plugged into lab power and connected to each other via appropriate cables. The computer is also connected to the University of Maryland network via ethernet cable. Installation of MATLAB 2014b, Motive, AscTec SDK 3.0, AscTec ACI Tool, AscTec Research Update Tool, and MinGW for C compilation capabilities occurs next. Motive requires a USB stick with licensing keys to be plugged into the PC in order to run.

Mounting of all OptiTrack Flex: V100 R2 cameras occurs next on three tripod stands along the far side and three tripod stands along the near side of the flight space. Two cameras are mounted on each stand; one near 6 ft, and one near 9ft. Comparisons are made between the setup shown and placing the cameras equidistantly along the perimeter, however the perimeter placement reduces the viewable width. The chosen setup gives a greater trackable area, shown in fig. 3.17, for quadcopter flights.

Calibration of the OptiTrack camera system occurs with Motive and use of the calibration wand and calibration square. Clear the flight space and surrounding lab area of any IR markers. Open Motive software on the PC and individual camera views can be seen to detect and remove any remaining markers from the space. Next, select 'Mask Visible' from the calibration pane in order to cover spots seen by cameras that would be disturbances, but are not IR markers in the space (such

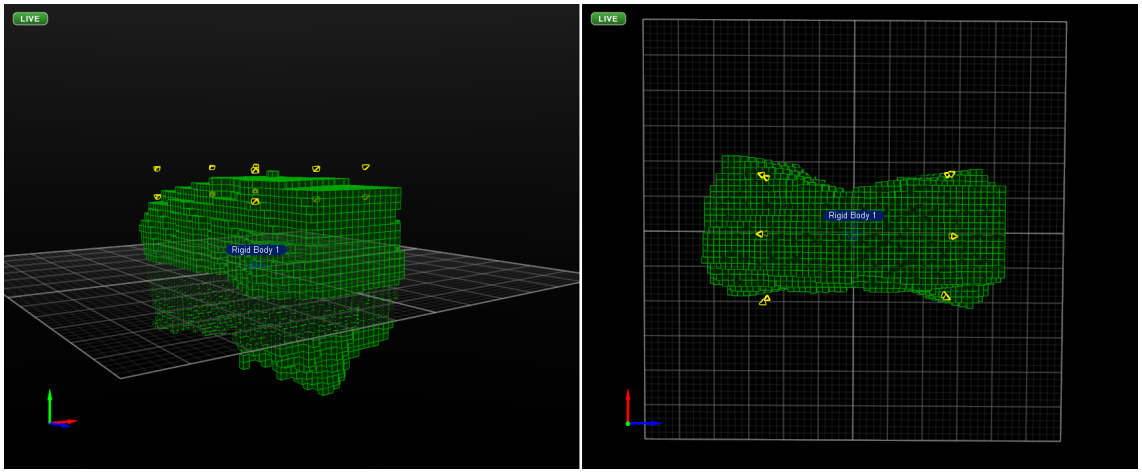

Figure 3.17: The volume in which at least three cameras have overlapping sight shown from the bottom left corner of the lab in a., and from above in b.

as light from other cameras). Clicking 'Start Wanding' and waving the wand in the flight space allows Motive to collect data points for creating a trackable area. Once enough data points are collected, Motive shows "Very High Quality" listed for the calibration. Next, Motive analyzes the data to generate the trackable area when 'Calculate' is selected. The calibration will be classified as exceptional if errors are minimized during the calculation. Lastly, ground plane setting occurs by placing the calibration square at the origin of the flight space with the Z side perpendicular to the rear wall and pointing toward the right wall. Select 'Set Ground Plane' and the flight space is set and the calibration can be saved for future use.

In order to track the quadcopter using the rigid body tracking capabilities of the OptiTrack system with the Motive software, infrared markers must be placed on the quadcopter. An asymmetric pattern in all axes is chosen to facilitate the angular tracking of the quadcopter as choosing any symmetrical pattern causes the software to sometimes flip the Euler angle orientation by 180 degrees along the axis

of symmetry. The central marker is screwed on the center of the quadcopter, one is placed on a carbon rod, and the remaining are glued to the edges as shown in fig. 3.18 a. Next we place the quadcopter in the trackable area of the flight space and select marker points in Motive to generate a rigid body as shown in fig. 3.18 b. Yellow rigid body center pivot point needs to be translated to the center point on the quadcopter for accurate rotation and position tracking. More information regarding the OptiTrack system can be found at [37].

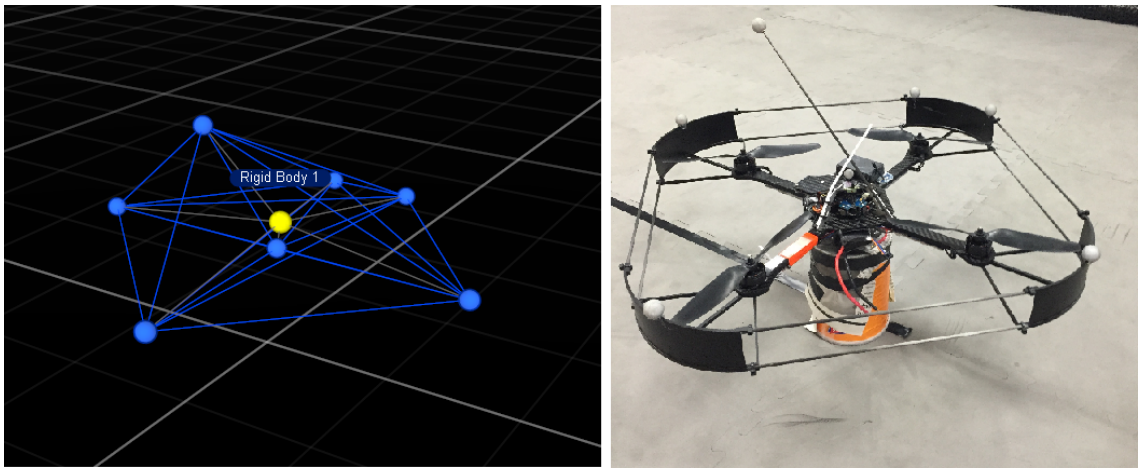

Figure 3.18: The rigid body selection in Motive resulting from the marker placement shown in a., and the physical placement on the quadcopter is shown in b.

The architecture description for the PC includes using Motive for collecting 6-DOF positional data for the quadcopter. This data is then sent using the NatNet protocol via TCP/IP connection to the MATLAB program by enabling frame data broadcasting using the OptiTrack Streaming Engine in Motive's data streaming panel. The MATLAB program makes use of the OptiTrack NatNet protocol to read the rigid body data including 6-DOF positional data and frame time data. Using this data, the system state is determined and provided to the controller. The controller

is an open code block which must be loaded with custom code for testing. The controller block provides outputs that should be set with individual motor speed commands during each loop run. Since the custom controllers will be designed to provide motor speeds, a conversion needs to be established associating motor rpm to appropriate motor command of 0-200. We secured the quadcopter to a heavy wooden frame, see figure 3.19, and used a digital tachometer to log motor speeds compared to motor commands and determined that the equation for conversion of rpm, r, to motor command, c, is approximately $c = (r - 1081.5)/37.267$.

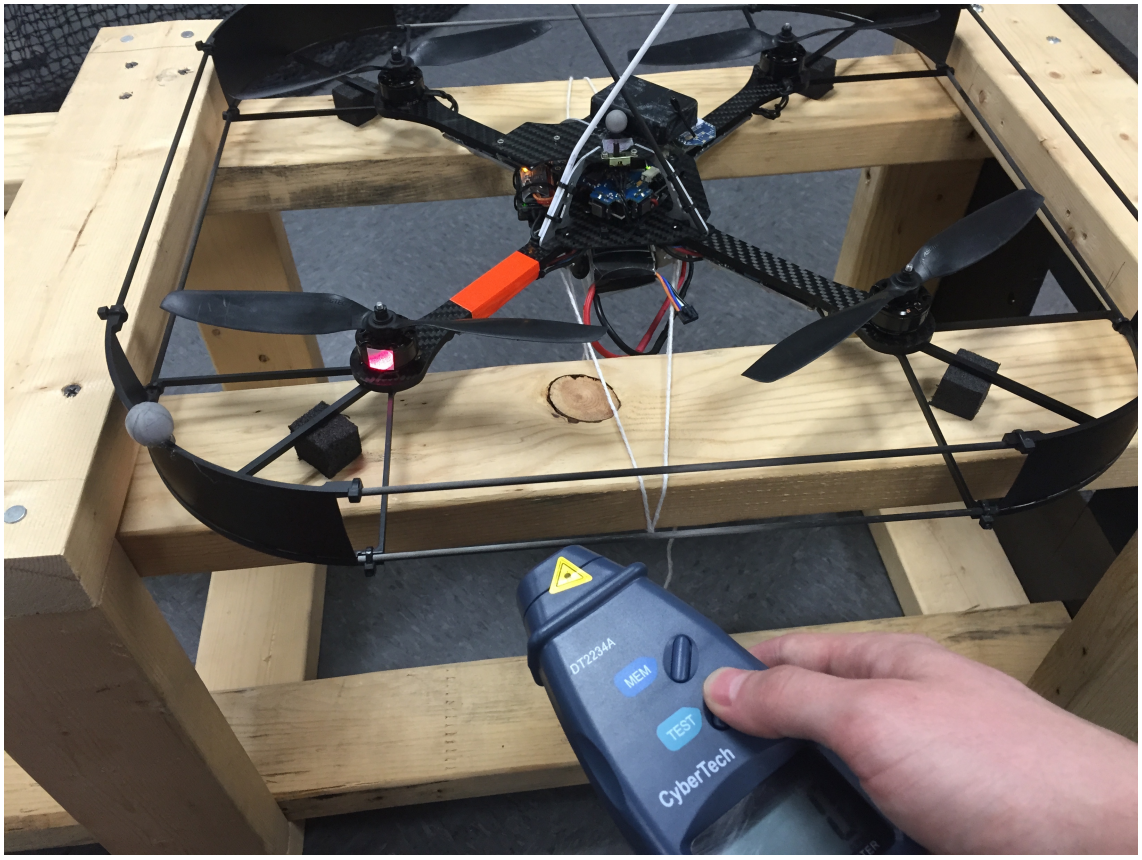

Figure 3.19: Quadcopter RPM Test Rig.

The MATLAB program then interfaces with the C program, providing the

individual motor commands that are to be sent to the quadcopter, by writing the four values to an output file along with a status which denotes if the flight controller is actively running or has finished. More information regarding NatNet and MATLAB can be found in [38] and [39], respectively. The MATLAB program code file list is included in Appendix B.

The C program is written based on the ACI command example which can be found at [34], but had to be migrated from the linux environment to the windows environment and is modified to stabilize runtime. In order to accomplish this, POSIX win32 libraries are installed for threading along with FTDI serial drivers and libraries for serial communication with XBEE. More details about Posix and FTDI can be found at [40] and [41], respectively. Additionally the C program uses ACI Remote protocol for wireless serial communication with the quadcopter, which AscTec denotes as the ACI Device. Running the program initiates a connection with the quadcopter and initializes command packets such that the quadcopter is set to accept direct motor commands of 0-200 via the wireless serial connection. The program loops and reads the output file of the MATLAB program that is updated at each frame read of the Motive software and includes the motor commands as well as a status which tells the C program whether to continue sending signals or to stop. If the status is active, then the C program sends the motor command packet to the quadcopter repetitively. The C program code file list is included in Appendix B.

For remote control of the quadcopter, the Futaba TC7 remote control (RC) interfacing provided by AscTec for the Hummingbird is used. First the RC and quadcopter receiver linking must occur. This is done by turning on the RC and then

the quadcopter while holding the link button on the receiver down for 3 seconds. The receiver led light should be a steady green if the connection is made. Calibration of the RC switches and joysticks can then occur with use of the AscTec Research Upgrade Tool. The PC serial port must be connected to the quadcopter LLP serial port via wired USB cable prior to running the Tool. More information on the TC7 remote and receiver can be found at [42] and [35].

In order to finalize implementation, the code on the quadcopter HLP must be loaded such that commands from the PC will be acknowledged and passed on to the LLP. The AscTec SDK on the PC provides HLP flashing capabilities using the JTAG connection with wired USB serial connection the the PC. The added lines of code are shown in Appendix B, and the successful flashing occurs based on methods in [34]. Instructions for flashing are also in the source code files of the AscTec SDK.

Note: A detailed parts list can be found in Appendix C.

3.6 Verification and Validation

System implementation will be finalized by completing verification and validation. Each component and subsystem must be tested against requirements for verification and then validated for use in the system. Once each part of the system has been verified and validated for use, the system can be tested to ensure a verified and validated design.

Flight Space Requirements:

1.1) The dimensions of the flight space shall be 12 feet x 18 feet. Verified by mea-

suring with measuring tape. Dimensions of the flight space are 12 feet x 18 feet.

1.2) The flight space shall protect the operator from direct impact with the quadcopter. Verified by testing the flight space double-netting. A wooden plank was thrown multiple times at the net and failed to protrude to the other side where the operator resides.

1.3) The flight space shall be open air. Verified by inspection that the flight space is open air.

1.4) Two nets shall be mounted dividing the flight space from the operator occupied space. Verified by inspection that two nets are mounted and divide the flight space from the operator occupied space.

The flight space requirements have been verified and as such the validation for inclusion in the system is complete. It is a space that protects the operator from the quadcopter, and provides a space for the quadcopter to fly in within the lab.

Quadcopter Requirements:

2.1) Quadcopter shall operate by commands via RC. Verified by linking the RC and successfully test flying the quadcopter with the Futaba TC7 remote control in manual mode.

2.2) Quadcopter shall have payload mount. Verified by inspection. The aluminum center frame of the quadcopter provides adequate payload mounting capabilities and was strong enough to support full RPM motor testing when attached to the wooden frame as in figure 3.19.

2.3) Quadcopter shall operate by commands via XBEE. Verified by test using the ACI Tool. Connection was made and motor commands were successfully sent to

the quadcopter. Success was determined by visual changes in motor speeds upon command sends.

5.7) IR Markers shall be placed on the quadcopter. Verified by inspection and can be seen in figure 3.18.

The quadcopter requirements have been verified and as such the validation of the quadcopter for inclusion in the system is complete. The quadcopter operates via RC, can have payload attached, and responds to wireless commands. It is also visible to the camera system.

Software Requirements:

2.4) Controller shell shall send heartbeat command at a minimum rate of 1Hz during controller run. Verified by logging number of times the loop was ran and the time of run to determine that the heartbeat command was sent at a rate close to 3Hz during any controller run.

4.2) Controller shell shall run on Windows 7 PC platform. Verified by inspection of successful MATLAB program runs on the Windows 7 PC.

4.3) Controller shell command output generation originating with state input shall take less than 0.01 seconds. Verified that time between frame processes within MATLAB is less than 0.01 seconds as long as custom control code processing is less than 5 milliseconds based on running timers during loop runs with MOTIVE capturing frames.

5.2) State information shall be provided at a minimum rate of 80Hz. Verified based on testing for loop counts over 10 second period to determine frequency that the frame update rate within MATLAB from Motive is 100Hz.

5.3) Motive Software shall be used on the Windows 7 PC to process camera data. Verified by inspection.

5.4) Processed Motive tracking data shall be used as quadcopter state information to be provided to the controller. Verified by inspection as Motive NatNet is used to transfer rigid body data to MATLAB where the quadcopter state is determined based on processed Motive tracking data.

5.5) The Motive API shall be used to feed tracking data from Motive to the controller. Verified by inspection as Motive NatNet is used and the API is used in the MATLAB program to handle the rigid body data.

6.2) Controller shell shall generate quadcopter motor commands. Verified by testing control block with rpm values and confirming motor command generation based on the relation determined during the RPM testing.

6.3) PC shall send quadcopter motor commands via XBEE. Verified by testing C program with various motor command inputs with visual confirmation from the moving rotors on the quadcopter due to received motor commands.

6.4) PC shall send commands at a minimum rate of 80Hz. Verified through testing of C program by running command send loop for 10 second runs and determining the frequency of the commands sent by the PC to be at 350Hz. This is within the capabilities of the XBEE communications which operate at 57.6kbps. The reason being is that the size of the command packet sent using the ACI protocol is 13 bytes and can theoretically be transmitted at 553Hz.

6.5) Ascending Technologies Communication Interface shall be used for sending commands from the PC to the quadcopter via XBEE. Verified by inspection in use of

ACI protocol in C program and successful information exchange as well as successful motor command packet sends to the quadcopter while using XBEEPRO for wireless communication.

All software capabilities needed for implementation of the system are verified and the software package is valid for system integration. The software captures the quadcopter visually and provides state data to a control block for processing. The processed motor speeds can be sent as motor commands wirelessly to the quadcopter and all updating occurs within the minimum desired times.

OptiTrack Hardware Requirements:

5.6) Camera Placement shall maximize the captured volume of the flight space. Verified by testing several orientations and using the volume visualization capability with minimum of three camera overlap to determine captured volume by the camera system. Maximized volumed can be seen in figure 3.17.

System Requiremtns:

1) System shall provide a flight space within the dimensions of the lab. Verified by inspection.

2) System shall use AscTec Hummingbird quadcopter for flights. Verified by inspection.

3) System shall accept operator quadcopter commands via RC. Verified through flight of quadcopter during system run.

3.1) System shall use Futaba TC7 Remote Control. Verified by inspection.

4) System shall run custom flight controller. Verified by test of motor ramping using system as well as LQR controller using system.

5) System shall use OptiTrack System for providing quadcopter state information to the controller. Verified by outputting state information received during system run to the MATLAB console.

6) System shall accept controller quadcopter motor commands. Verified by test of motor ramping using system as well as LQR controller using system.

6.1) System shall accept motor commands on Windows 7 PC. Verified by test of motor ramping using system as well as LQR controller using system.

7) System shall accept operator command to run controller. Verified by test of system start to run custom controller.

8) System shall accept operator command to stop controller. Verified by system test of remote control shut-off of wireless command using the RC switch.

9) System quadcopter shall fly in flight space. Verified by system test of LQR controlled flight and determination that the quadcopter could not leave flight space boundaries through prior tests of netting done with flight space requirements.

Based on the verification of the system and following demonstrations of the system operation in section 4.5's flight tests, we can say that the system is validated for use. The TERP system provides state information to the controller in a timely manner. The TERP system enables users with the space to run a custom controller with states as the feedback mechanism. It then uses motor commands from the controller to wirelessly command the quadcopter for flight in a protective flight space with options for the operator to take remote control flight of the quadcopter. The 100Hz frequency operation of the system loop is guaranteed only by the fact that the custom controller processing remains within the 5 millisecond time requirement

for the controller code block.

It is important to also note that all needs developed at the beginning of this system design have been fulfilled according to stakeholders and the future upkeep or modification of the system will be much simpler based on the modeling performed and information made available through this design process.

Chapter 4: Control System Development

4.1 Linear Quadratic Regulator Control

The linear quadratic regulator (LQR) controller is based on optimal control theory in which a control is chosen such that it will minimize a cost function associated with the current state and control of the system as in the following problem from [43].

$$\begin{aligned} \underset{u(.)}{\text{minimize}} \quad & J(x,u) = \int_0^T L(x,u)dt + V(x(T)) \\ \text{subject to} \quad & \dot{x} = f(x,u),\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m \end{aligned} \tag{4.1}$$

In our specific case, we are not just developing a controller to keep our system at an equilibrium point. We are developing this LQR controller in order to achieve trajectory tracking with our system. In order to achieve trajectory tracking, integral action is used to augment the system with additional states z based on the integral error of the output states to be tracked, $e = \int y - r$ [44]. The new states, $\dot{z} = y - r$, have a zero equilibrium point and thus result in $y = r$. Taking the continuous linear time-invariant system (2.5) from Chapter 2 and applying integral action leads to the following system.

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A_p & 0_{12\times4} \\ C_p & 0_{4\times4} \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} B_p \\ 0_{4\times4} \end{bmatrix} u + \begin{bmatrix} 0_{12\times4} \\ -\mathbb{I}_{4\times4} \end{bmatrix} r$$

The system can then be rewritten as a 16 state system where A, B, and B_c matrices correspond to the above system respectively.

$$\dot{x} = Ax + Bu + B_c r. \tag{4.2}$$

System (4.2) is controllable which enables use of LQR control to drive the states to zero thereby ensuring tracking of the trajectory. The LQR control is developed through modifying problem (4.1) in which we choose an infinite time horizon and eliminate the terminal cost. The resulting problem is the following as in [43].

$$\begin{aligned} \underset{u(.)}{\text{minimize}} \quad & J(x,u) = \int_0^\infty (x^T Q x + u^T R u) dt \\ \text{subject to} \quad & \dot{x} = Ax + Bu,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ x_o \text{ given} \end{aligned} \tag{4.3}$$

The solution to the optimization problem follows from the solution P of the matrix algebraic Riccati equation $PA + A^T P - PB^{-1}RB^T P + Q = 0$ where Q and R are symmetric positive definite matrices. Using the feedback law $u = -K(x - x_d)$, where K is the constant gain $R^{-1}B^T P$, minimizes the cost function J and guarantees $x(t) \to 0$ as $t \to \infty$.

In our case, x_d is the desired x reference state at time t which corresponds to our r in system (4.2). Also, as the simplification to system (2.5) is based on assumed equilibrium thrust of mg, the control law must be compensated with $u_d = [mg\ 0\ 0\ 0]^T$

such that

$$u = -K(x - x_d) + u_d. \tag{4.4}$$

In order to apply this control law, $Q \in \mathbb{R}^{16\times16}$ and $R \in \mathbb{R}^{4\times4}$ must be chosen as symmetric positive definite matrices. Suppose we want to minimize the error of outputs z that coincide with states of the system such that $z = Hx$. Being that (A, H) is observable when only the integral error states are selected as outputs to be minimized, choose $Q = H^T H$ and $R = \rho\mathbb{I}^{4\times4}$ where $\rho \in \mathbb{R}$ is used to modify state/input cost [43].

4.2 LQR Simulations

Directional flight simulations of the control law designed in section 4.1 for various choices of ρ will allow for making a better decision for the final baseline LQR controller. The simulink model of the AscTec Hummingbird developed in section 2.2 will be used for testing the LQR controller in simulink. Following the choice of the LQR gain matrix based on the directional response, various flight paths will be simulated to visualize the reference trajectory tracking capabilities of the LQR controller.

Figure 4.1 on the following page shows the LQR controller set up in Simulink. Output has been added to see if individual motor speeds stay within the bounds determined of 0-8000rpm while following the command of translation along x or z at 0.25m/s. Based on the directional responses in figures 4.2 and 4.3, we chose to use ρ of 0.1 for simulating flight paths.

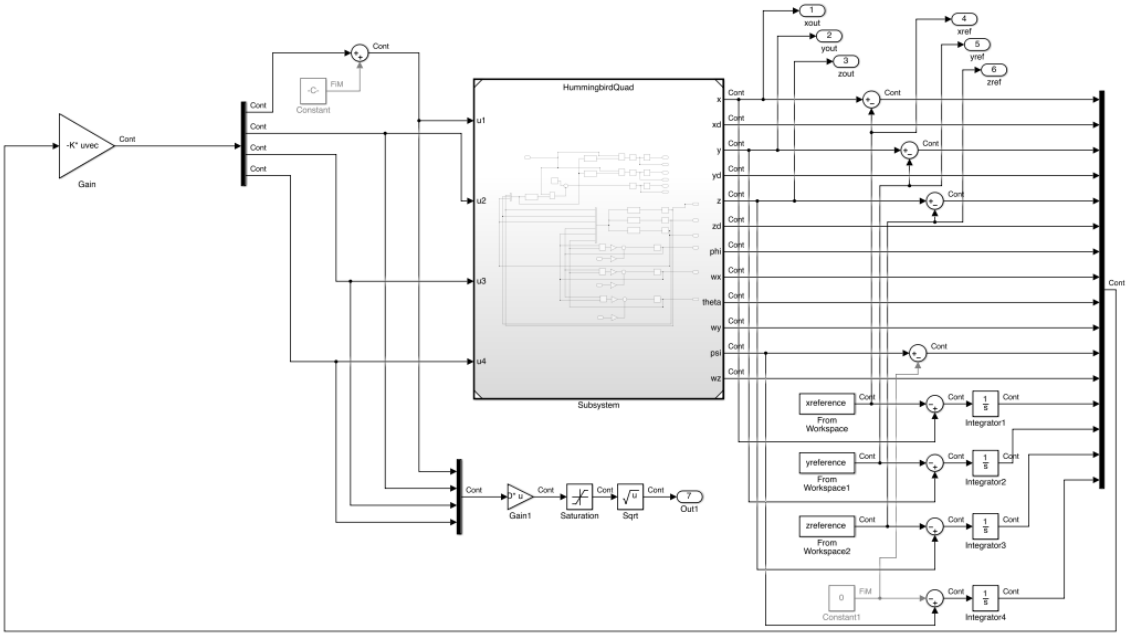

Figure 4.1: LQR Controller Simulink Model.

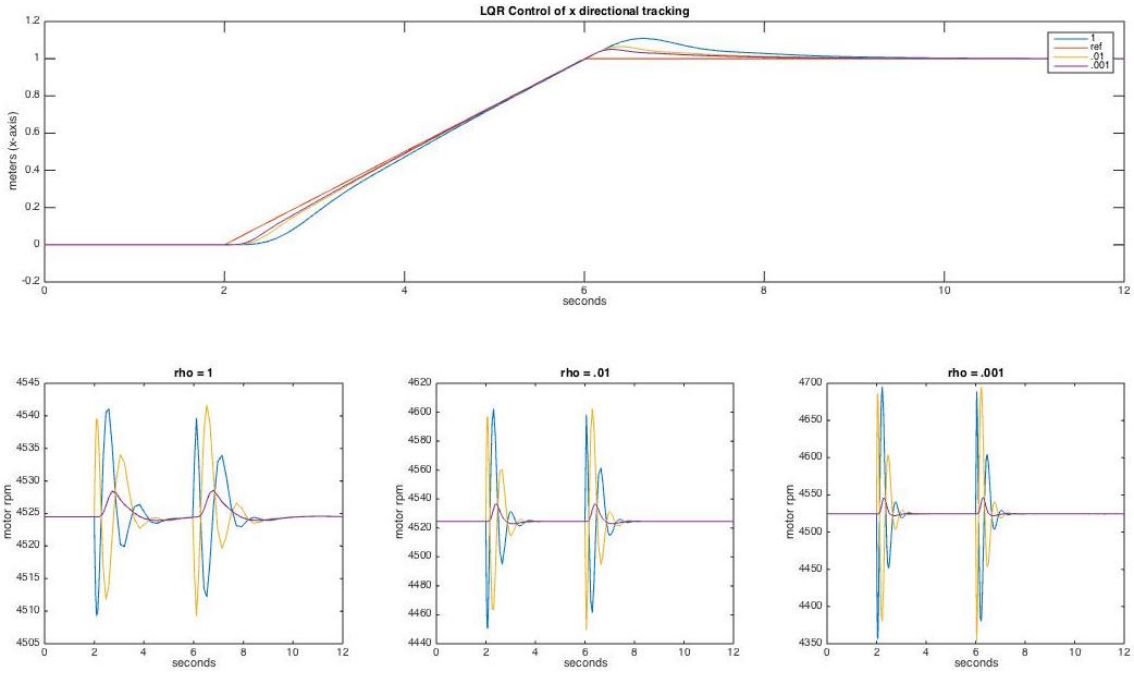

Figure 4.2: LQR Controller x directional tracking. Due to the assumed symmetry of the quadcopter, these results also hold for y directional tracking.

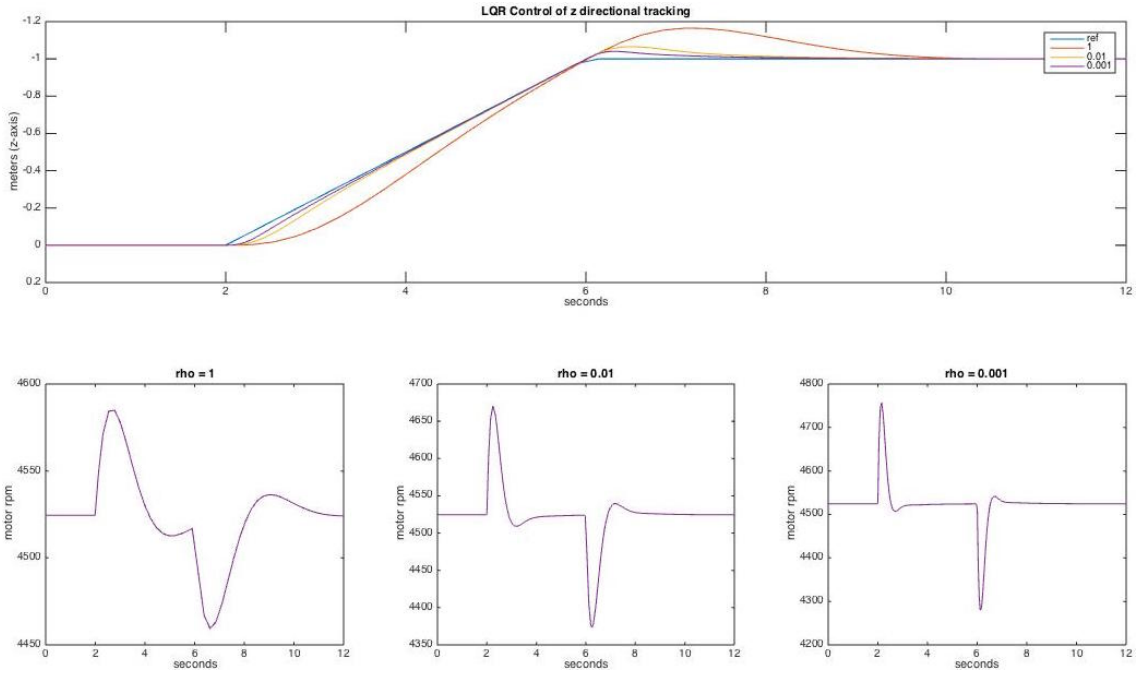

Figure 4.3: LQR Controller z directional tracking.

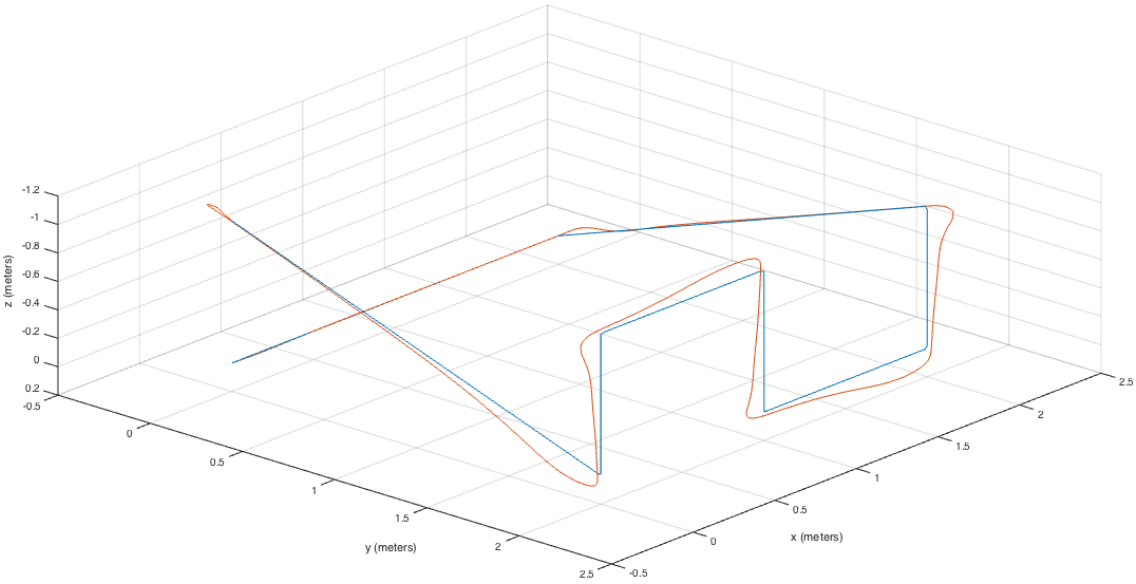

Figure 4.4: LQR Controller flying a square path with altitude changes. Reference trajectory is in blue, quadcopter flight trajectory is in orange.

The blue flight paths in figures 4.4, 4.5, and 4.6 are flown in a clockwise direction and do not exceed a velocity of 0.75m/s at any point in the trajectory.

The orange paths are the simulated flights of the Hummingbird quadcopter with the LQR controller. The flights stay within 15 cm of the desired trajectory, with the greatest of the overshoots occurring at sharp directional changes as the trajectories were not designed explicitly with the dynamic capabilities of the quadcopter in mind.

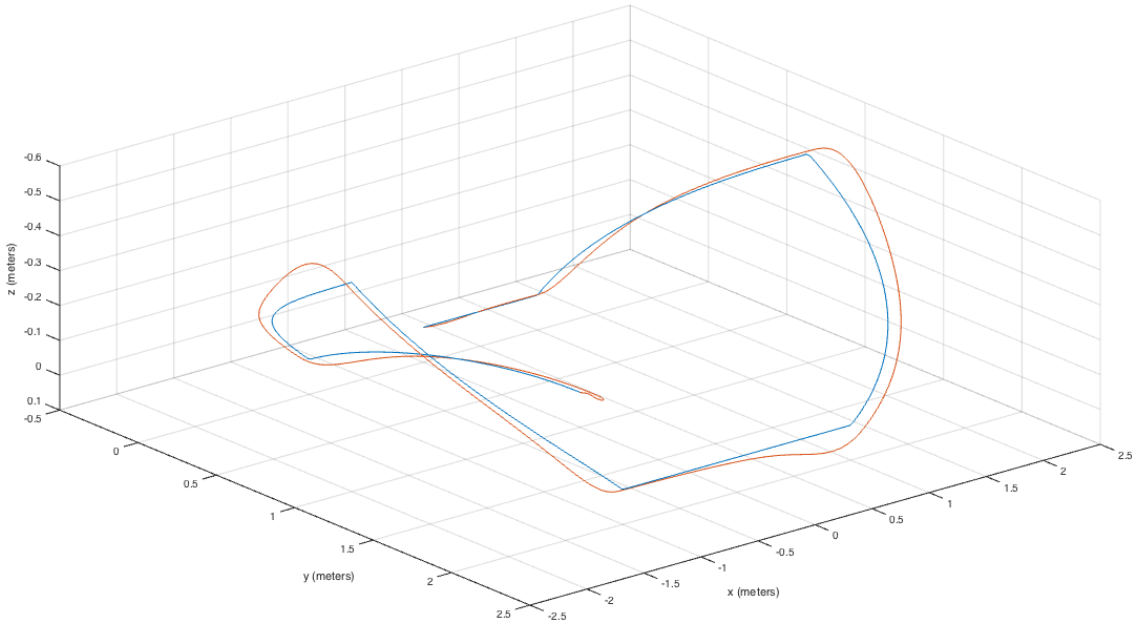

Figure 4.5: LQR Controller flying a circular path with altitude changes. Reference trajectory is in blue, quadcopter flight trajectory is in orange.

4.3 Model Reference Adaptive Control

In this section we will seek to match and improve upon the flight trajectory tracking capabilities of the LQR controller by using an adaptive controller developed by Dydek, Annaswamy, and Lavretsky in [30] to account for changes in the modeled quadcopter dynamics which will involve a variation in mass as well as inertial composition of the aircraft due to a carried payload.

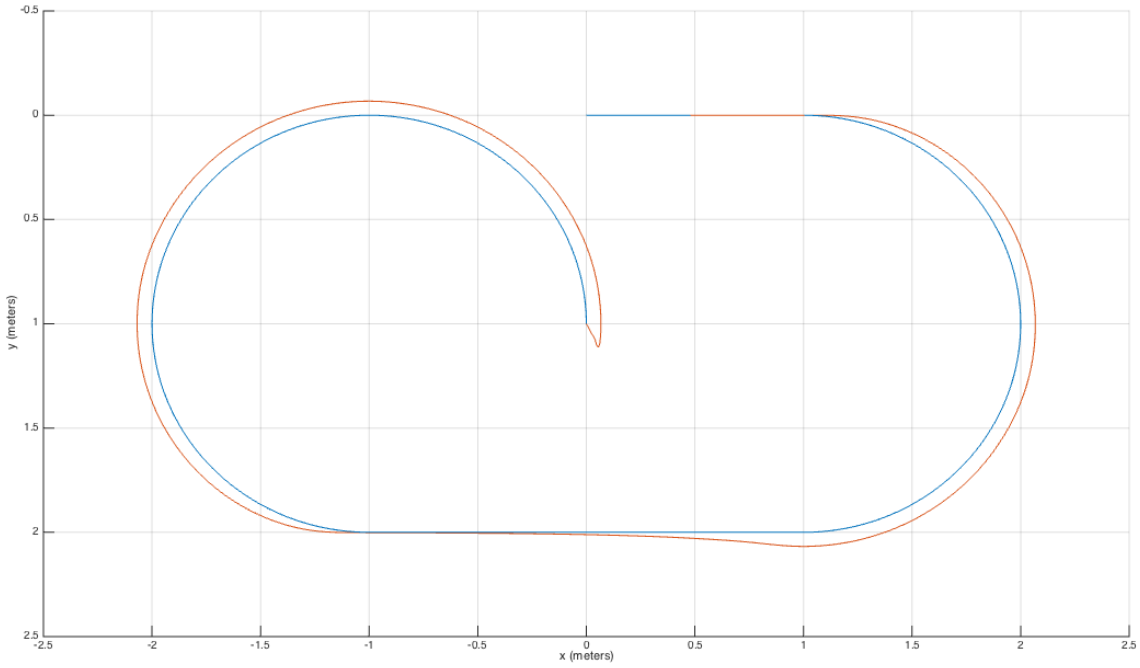

Figure 4.6: LQR Controller flying a circular path with altitude changes. Reference trajectory is in blue, quadcopter flight trajectory is in orange.

Consider again system (2.5) from chapter 2,

$$\begin{aligned} \ddot{x} &= -g\theta \\ \ddot{y} &= g\phi \\ \ddot{z} &= -\tfrac{1}{m}\triangle u_1 \\ \dot{\phi} &= \tfrac{l}{J_x}u_2 \\ \dot{\theta} &= \tfrac{l}{J_y}u_3 \\ \dot{\psi} &= \tfrac{1}{J_z}u_4 \end{aligned} \tag{4.5}$$

which is similar to equation 2 in [30]. We set up our problem in similar fashion to equation 4 in [30] by using the augmented system, (4.2), from earlier with the assumptions of uncertainties in the plant matrix A_P and possible thrust uncertainties

represented as Λ in

$$\dot{x} = \begin{bmatrix} A_p & 0_{12\times4} \\ C_p & 0_{4\times4} \end{bmatrix} x + \begin{bmatrix} B_p \\ 0_{4\times4} \end{bmatrix} \Lambda u + \begin{bmatrix} 0_{12\times4} \\ -\mathbb{I}_{4\times4} \end{bmatrix} r \tag{4.6}$$

We take the closed loop system developed in section 4.1 and, provided there exists a matrix K such that the actual system in (4.6) is stable, supplement the baseline control with an adaptive part. In MRAC, the control law is based on using time-varying adaptive parameters, Θ, along with a regressor vector, ω [46]. In our case we have $\Theta^T = [K_a^T \ \Theta_r^T \ \Theta_d^T]$ and $\omega^T = [x^T \ r^T \ 1]$, composed of states of the plant, reference states, and a constant for possible offsets.

Typically in MRAC, the adaptive parameter update law results from using Lyapunov techniques for derivation based on a plant-model system error representation, as in [46], to guarantee asymptotic tracking of the model reference. The adaptive parameter update law, $\dot{\Theta} = -\Gamma\omega e^T PB$, given by [30] will be used for our simulations where P is the solution to the Lyapunov equation,$(A-BK)^T P+P(A-BK) = -Q$, where P and Q are symmetric, positive definite matrices, A and B are from the augmented system, (4.6), and K is the matrix developed through LQR methods. The control law, $u = -Kx+\Theta^T\omega$, based on [30] differs in composition to, $u = \Theta^T\omega$ developed in the work of [46]. It should however not affect the expected results as the latter would be started with an initial guess of the baseline controller gains for the adaptive parameters. It should also be noted that in working through the proof of model reference tracking error asymptotically going to zero, we were only able to

prove that uncertainties in the matrix Λ could be accounted for. The plant matrix A_p had to be known.

4.4 MRAC Simulations

We model the MRAC approach in Simulink as in fig. 4.7 and then simulate with the same square and circular reference trajectories as we did with the LQR controller. The following flight simulations in figures 4.8 and 4.9 involve the Hummingbird plant without modifications or payloads. In the model of fig. 4.7, the larger center block is the Hummingbird plant, the top block is the reference model, the lower block is the adaptive law update, and the left block is the controller.

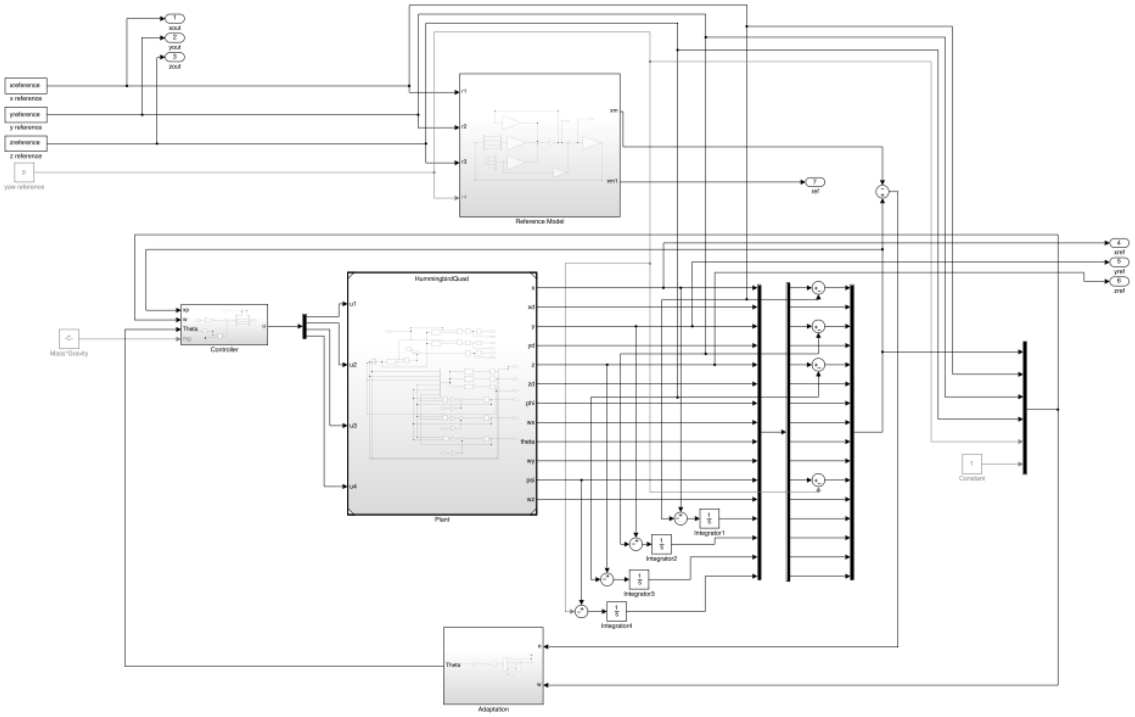

Figure 4.7: Full State MRAC Simulink Model.

The quadcopter begins to diverge from the square flight path around 10 seconds

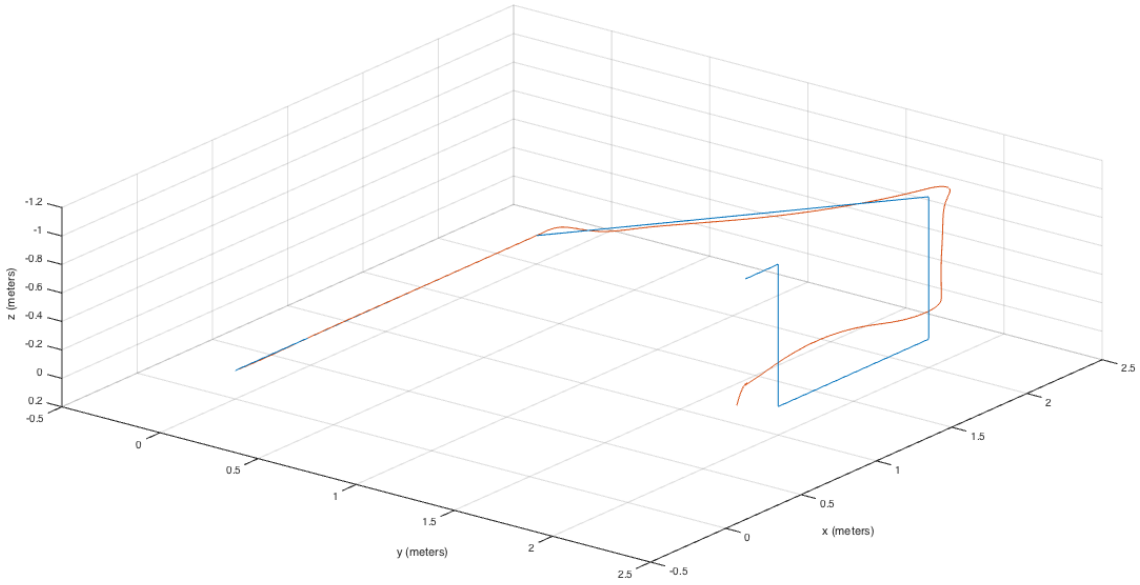

Figure 4.8: Full State MRAC Square Flight Simulation. Reference trajectory is in blue, quadcopter flight trajectory is in orange.

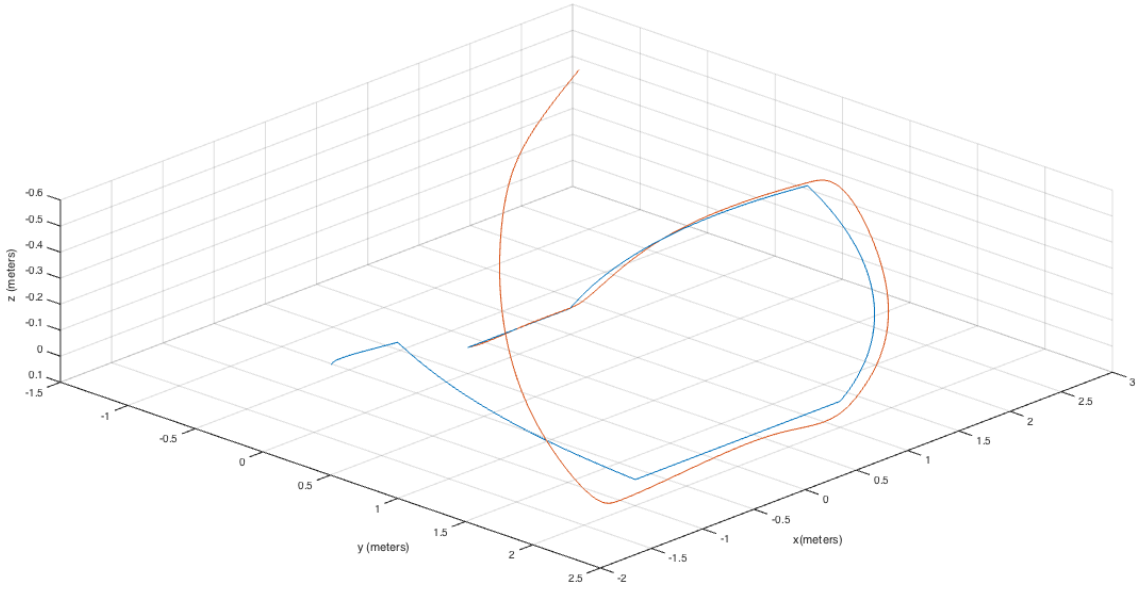

Figure 4.9: Full State MRAC Circle Flight Simulation. Reference trajectory is in blue, quadcopter flight trajectory is in orange.

into the flight, and the quadcopter begins to diverge from the circular flight path around 12 seconds into the flight. Several iterations were attempted with variations in adaptive gains and relying solely on the adaptive law without the baseline. None

of the attempts resulted in successful flight simulations. The failures may result from inaccuracies in precision or numerical errors due to the mathematical capabilities of MATLAB and Simulink.

4.5 Flight Tests

In order to validate the TERP system as well as test our LQR controller, we will conduct a series of system runs including flight tests. We first conduct a motor ramping test for basic validation. This shows proper integration of all parts of the system and is based on visual and auditory confirmation of the motor command pattern sent through ramping.

The next set of system runs involves step response in the z direction while using the LQR controller. We set up the quadcopter on a cylindrical base approximately 20cm above the ground and command a flight to 40cm above the ground for 8 seconds. Attempts were made with ρ set at 0.1, 1, 1.5, and 2. All flights became unstable at one point in the x and y directions and the controller was manually shut off. None of the flights left an area of one cubic meter showing that some relative sense of stability was achieved.

It was during these flights that we noticed the problem with our integral error accumulation overflowing the variable and causing MATLAB to crash. This occurs when the controller is not able to control the flight trajectory. These flights may also be developing instability due to the delays and simplicity of state estimation of the velocities which has been carried out simply based on two prior positional

points.

The TERP system has been validated through its use for running a set of custom controllers. It is easily modifiable for future controllers to be carried out by others.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

The development of the Quadcopter Test Environment and Research Platform through model-based systems engineering resulted in a successfully verified and validated product in a timely manner. This system can be used for testing custom quadcopter control strategies based on feedback provided by the camera system. These control strategies are then ran on the lab PC and wireless communication enables commands to be sent to the quadcopter for flight tests in a trackable flight space built in the Cooperative Autonomy Lab at the University of Maryland - College Park. Through the development process, the following system engineering artifacts were created to help model the system: structural block diagrams, use case diagram, use case narratives, activity diagram, system architecture internal block diagram, subsystem architecture internal block diagrams, requirements diagram, requirements table, implementation instructions, and operation instructions.

Integral error augmentation of the quadcopter model enabled successful development of a flight trajectory tracking controller through linear quadratic regulator methods. Valid cost function weights were chosen to minimize errors in the trajectory tracking and successful simulations in both square and circular patterns took

place. The model-reference adaptive control approach taken did not provide similar results in simulation. The failed adaptive based flight simulations may have been due to numerical errors in MATLAB or possible implementation errors.

Finally, flights using the TERP system along with the LQR controller to perform step responses were attempted. The Flights remained within a meter squared, however became unstable due to bad state estimates and associated noise due to estimates and delays. The accomplishment of successfully using the TERP system to test the flight with a custom controller successfully validated the system for use by future users.

5.2 Future Work

The primary avenue of future work to be carried out involves the development of a Kalman filter based on the linear quadcopter model to implement within the MATLAB custom code block. In doing so we hope to provide a far better state estimate. These estimates should enable successful flight tests based on the LQR control strategy by alleviating the instability due to state errors and delays affecting the system. Flight trajectory tracking tests with payloads can then be attempted.

In order to address the difficulties associated with the adaptive approach we took, we would like to explore a second approach based on [45]. Instead of attempting full reference state tracking, they focus on reference output tracking of x, y, z, and ψ which results in the relaxation of conditions for model choices that can be achieved. This approach is attractive due to the nature of the flights we are

attempting. The flights will be slow, small angled, and hope to be precise in moving payloads. To have positional and yaw tracking should then be sufficient, and roll, pitch, and acceleration tracking can be ignored.

Our suggestions for future work based on the development of this system include research in alternative control strategies, development of flight trajectories, and extension of the system for multi-vehicle flights and group dynamic research. The quadcopter can be used with an internal control loop using on-board sensor information, and different command packets can be used for developing alternative control methods. Additionally, the flight trajectories chosen were not designed with the exact capabilities of the quadcopter in mind. Better trajectory tracking shall be accomplished by creating more flyable trajectories using the dynamic capabilities of the quadcopter as reference.

Appendix A: Use Case Narratives

Use Case ID: U1
Use Case Name: Load Controller to System
Level: Actor-to-System
Iteration: 1
Description: Operator loads the controller to the PC in the System.
Assumptions:

- The operator is familiar with the system.
- Power to the system is stable and consistent.
- Communication between parts of the system is stable and consistent.
- The system is calibrated for use.
- The controller is a valid program to be loaded, i.e. it satisfies the controller guidelines.
- PC has enough memory to perform manipulation and loading operations.

Actors: Operator
Preconditions and Triggers:

- The operator desires to load a controller to the system.

Main Success Scenario:

1. The operator accesses the PC in the system and locates the port for the controller.
2. The PC in the system provides visual feedback that indicates the operator is in the correct location.
3. The operator loads the controller into the system.
4. The PC in the system provides visual feedback confirming that the loading of the controller has occurred.

Post-conditions and Guarantees:

- The operator has successfully loaded the controller to the system.
- System remains on and is calibrated.
- Electing to have the system run the controller would result in successfully running the most recently loaded controller.

Extensions/Exceptions Scenarios:
Specific Requirements
Functional Requirements:

1. System shall accept loading of custom controller.
2. PC shall run custom controller.
3. System shall provide quadcopter state data processed via Motive to controller.
4. System shall take quadcopter motor command from controller.

Performance Requirements:

1. Motor commands shall be updated at a minimum rate of 80Hz.

Use Case ID: U2
Use Case Name: Run Controller on System
Level: Actor-to-System
Iteration: 1
Description: Operator elects to run the controller on the system.
Assumptions:

- The operator is familiar with the system.
- Power to the system is stable and consistent.
- Communication between parts of the system is stable and consistent.
- The system is calibrated for use.
- A controller is loaded.
- PC has enough memory to perform manipulation and loading operations.
- Unless otherwise noted, RC sticks are in default positions.

Actors: Operator
Preconditions and Triggers:

- The operator desires run a controller on the system.

Main Success Scenario:

1. Operator activates quadcopter motors by sending full-yaw and zero-thrust command on the RC simultaneously for 3 seconds.
2. Operator activates quadcopter Xbee wireless acceptance of control by moving switch B on the RC backward.
3. System provides visual feedback of successful activation by rotating all propellers on the quadcopter at the lowest speed.
4. Operator starts controller and begins to monitor flight.
5. System PC starts run of controller.
6. Controller sends motor commands and heartbeat to quadcopter while controller runs.
7. Operator determines the flight is complete.
8. Operator deactivates quadcopter Xbee wireless acceptance of control by moving switch B on the RC forward.
9. System now accepts RC commands for control of quadcopter.
10. Operator deactivates quadcopter motors by sending full-yaw and zero-thrust command on the RC simultaneously for 3 seconds.
11. System provides visual feedback of successful activation by stopping all propeller rotations.
12. Operator stops controller.
13. System provides visual feedback that controller has stopped running.

Post-conditions and Guarantees:

- The system has successfully run the loaded controller.
- System remains on and is calibrated.

Extensions/Exceptions Scenarios:

6a. An early termination of flight desired where manually an emergency flight stop is not required. Operator turns off RC and the quadcopter system performs immediate balancing, stop of translation, and descent at a constant rate. Operator must then perform 8 followed by turning the RC back on. Continue at 9.

Specific Requirements

Functional Requirements:

1. Quadcopter shall operate by commands via RC.
2. System shall have a method to perform emergency stop.
3. Quadcopter shall operate by commands via XBee.
4. PC shall send quadcopter motor commands via XBee.
5. Controller shell shall send heartbeat command at a minimum rate of 1Hz during controller run.
6. System shall accept operator command to run controller.
7. System shall accept operator command to stop controller.
8. System quadcopter shall operate in flight space.
9. System shall provide flight space within the dimensions of the lab.
10. Flight space shall keep operator safe.
11. System shall make use of quadcopter system emergency landing protocol.

Performance Requirements:

1. System shall send command packet at a minimum rate of 80Hz.
2. System shall send heartbeat at a minimum rate of 1Hz.

Appendix B: Code Information

Please contact the author for copies of the code: deprins@ieee.org

B.1 C Program List

1. asctecCommIntf.c [34]
2. asctecCommIntf.h [34]
3. asctecDefines.h [34]
4. file.txt
5. ftd2xx.h [41]
6. main.c
7. makefile
8. pthread.h [40]

B.2 MATLAB Program List

1. MotorCmd.m
2. RunController.m
3. quaternion.m [38]
4. NatNetML.dll [38]

B.3 Hummingbird HLP Code

The following lines of code were added to the HLP block in the AscTec SDK file 'sdk.c':

```
aciSyncVar();

aciSyncCmd();

aciSyncPar();

aciEngine();
```

Appendix C: TERP System Parts List

1. Ascending Technologies Hummingbird Quadcopter
2. Ascending Technologies JTAG adapter
3. Ascending Technologies USB to AutoPilot Connection Cable
4. Digi International XBEEPRO USB module
5. Thunder Power RC 2100mAh LiPo Battery
6. X Peak 230 Bal Charger
7. Cell Balancer Adapter connection to charger
8. LiPo connection to charger cable
9. RC connection to charger cable
10. Futaba T7C Remote Control
11. Windows PC tower
12. Planar PX 2710MW Monitor ($\times 2$)
13. Dell USB keyboard

14. Logitech USB mouse

15. Giottos LC325 tripod stand (×6)

16. OptiTrack Flex:V100 R2 camera (×12)

17. OptiTrack OptiHub 2 (×2)

18. USB B-Male to A-Male cable (×2)

19. USB A-Male to B-Mini cable (×13)

20. USB A-Female to A-Male active extension cable

21. RCA Hub Sync Cable Hub

22. OptiTrack USB Hardware Key

23. OptiTrack calibration square

24. OptiTrack calibration wand

Bibliography

[1] DJI. (2015). *DJI - The World Leader in Camera Drones/Quadcopters for Aerial Photography* [Online]. Available: http://www.dji.com/

[2] Parrot. (2015). *Civil Drones* [Online]. Available: http://www.parrot.com/usa/drones/

[3] S. Bouabdallah, "Design and Control of Quadrotors with Application to Autonomous Flying" Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2007.

[4] Z. Dydek, "Adaptive Control of Unmanned Aerial Systems" Ph.D. dissertation, Dept. of Mech. Eng., MIT, Cambridge, MA, 2011.

[5] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M.W. Mueller, J.S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The FLight Assembled Architecture Installation: Cooperative construction with flying machines," *IEEE Control Syst. Mag.*, vol. 34, no. 4, Aug. 2014. pp. 46-64.

[6] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-air vehicle," in*Proc. IEEE Conf. Robot. Autom.*, 2012, pp.9-15.

[7] S. Lupashin, A. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadcopter multi-flips," in *Proc. IEEE Int. Conf. Robot. and Automat.*, Anchorage, AK, 2010, pp. 1642-1648.

[8] Z. Stone. (2013, August 7). *A Beautiful Film of New York City, Shot Entirely By Drone* [Online]. Available: http://www.fastcoexist.com/1682779/a-beautiful-film-of-new-york-city-shot-entirely-by-drone

[9] Lockheed Martin. (2015). *Indago VTOL Quad Rotor* [Online]. Available: http://www.lockheedmartin.com/us/products/procerus/quad-vtol.html

[10] B. Coxworth. (2013, May 13). *Canadian police save a man's life, using a drone.* [Online]. Available: http://www.gizmag.com/rcmp-quadcopter-locate-victim/27488/

[11] DHL. (2014, September 24). *DHL parcelcopter launches initial operations for research purposes* [Online]. Available: http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html

[12] Amazon. (2015). *Amazon Prime Air* [Online]. Available: http://www.amazon.com/b?node=8037720011

[13] Cornell Univ. (2015). *Systems Engineering @ Cornell* [Online]. Available: http://www.systemseng.cornell.edu/se/

[14] Johns Hopkins Univ. (2015). *Systems Engineering* [Online]. Available: http://ep.jhu.edu/graduate-programs/systems-engineering

[15] Georgia Inst. Tech. (2015). *Systems Engineering* [Online]. Available: https://pe.gatech.edu/subjects/engineering/systems-engineering

[16] Colorado State Univ. (2015). *Systems Engineering* [Online]. Available: http://www.online.colostate.edu/degrees/systems-engineering/

[17] Univ. of Maryland. (2015). *Institute for Systems Research* [Online]. Available: https://www.isr.umd.edu/home

[18] G. Hoffman, D. G. Rajnarayan, S.L. Waslander, D. Dostal, J.S. Jang, and C. Tomlin, "The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC)," in *Proc. 23rd Digital Avionics Systems Conf.*, Salt Lake City, UT, 2004, pp. 12.E.4-1-12.E.4-10.

[19] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-Time Autonomous Vehicle Test Environment," *IEEE Control Syst. Mag.*, vol. 28, no. 2, pp. 51-64, Apr. 2008.

[20] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP Multiple Micro-UAV Test Bed," *IEEE Robot. Automat. Mag.*, vol. 17, no. 3, pp. 56-65, Sept. 2010.

[21] S. Lupashin, M Hehn, M. W. Mueller, A. P. Shoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, vol. 24, no. 1, pp. 41-54. Jan. 2014.

[22] M. Austin, "Introduction" in *Systems Engineering Requirements, Design and Tradeoff Analysis*, Inst. Syst. Research, Univ. Maryland, College Park, Jan. 2014, ch. 1, pp. 3-44

[23] S. Friedenthal, "Preface" in *A Practical Guide to SysML: The Systems Modeling Language*, 2nd ed. Waltham, MA: Morgan Kaufmann, 2012, pp. xvii

[24] M. Austin, "Foundations of Model-Based Systems Engineering" in *Systems Engineering Requirements, Design and Tradeoff Analysis*, Inst. Syst. Research, Univ. Maryland, College Park, Jan. 2014, ch. 3, pp. 120-189

[25] M. Austin, "Requirements Engineering" in *Systems Engineering Requirements, Design and Tradeoff Analysis*, Inst. Syst. Research, Univ. Maryland, College Park, Jan. 2014, ch. 9, pp. 335-395

[26] S. Friedenthal, "Au Automobile Example Using the SysML Basic Feature Set" in *A Practical Guide to SysML: The Systems Modeling Language*, 2nd ed. Waltham, MA: Morgan Kaufmann, 2012, ch. 4, pp. 51-86

[27] M. Austin, "Foundations of Model-Based Systems Engineering" in *Systems Engineering Requirements, Design and Tradeoff Analysis*, Inst. Syst. Research, Univ. Maryland, College Park, Jan. 2014, ch. 4, pp. 335-395

[28] Austin. (2015). *V-Model-Systems-Development* [Online]. Available: http://eng.umd.edu/ austin/enes489p/images/V-Model-Systems-Development.png

[29] X. Zhang, X. Li, K. Wang, and Y. Lu, "A Survey of Modelling and Identification of Quadrotor Robot," *Abstract and Applied Analysis*, vol. 2014, art. i.d. 320526 pp. 1-16

[30] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluation," *IEEE Trans. Control Syst. Tech.*, vol. 21, no. 4, pp. 1400-1406, July 2013.

[31] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664-674, Jan. 2012.

[32] M. Schreier, "Modeling and Adaptive Control of a Quadrotor," in *Proc. 2012 IEEE Int. Conf. Mechatronics Automat.*, Chengdu, China, 2012, pp. 383-390.

[33] D. Mellinger, Q Lindsey, M Shomin, and V. Kumar, "Design, Modeling, Estimation and Control for Aerial Grasping and Manipulation," in *Proc. 2011*

IEEE/RSJ Int. Conf. Intelligent Robotics and Syst., San Francisco, CA, 2011, pp. 2668-2673.

[34] Ascending Technologies GmbH. (2014). *AscTec Research Home* [Online]. Available: http://wiki.asctec.de/display/AR/AscTec+Research+Home

[35] Futaba. (2012, Apr.). *Futaba R6303SB Manual* [Online]. Available: http://manuals.hobbico.com/fut/r6303sb-manual.pdf

[36] Natural Point. (2012). *OptiHub Quick Start Guide* [Online]. Available: http://www.optitrack.com/static/documents/OptiHub Quick Start Guide.pdf

[37] Natural Point. (2015). *Motion Capture Systems - OptiTrack* [Online]. Available: http://www.optitrack.com/

[38] Natural Point. (2013, September). *NatNet API User's Guide* (Version 2.5) [Online]. Available: http://www.naturalpoint.com/optitrack/static/documents/NatNet API User Guide.pdf

[39] MathWorks. (2015). *MATLAB* [Online]. Available: http://www.mathworks.com/products/matlab/

[40] R. Johnson. (2015). *POSIX Threads (pthreads) for Win32* [Online]. Available: https://www.sourceware.org/pthreads-win32/

[41] FTDI. (2012, February 23). *Software Application Development D2XX Programmer's Guide* (Version 1.3) [Online]. Available: http://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf

[42] Futaba. (2015). *Futaba 7C 7-Channel 2.4Ghz System* [Online]. Available: http://www.futaba-rc.com/systems/futk7000.html

[43] R. M. Murray. (2010, February). *Optimization-Based Control* (Version 2.1a) [Online]. Available: http://www.cds.caltech.edu/ murray/books/AM08/pdf/obc-complete_15Feb10.pdf

[44] K. J. Astrom and R. M. Murray. (2010, February). *Feedback Systems: An Introduction for Scientists and Engineers* (Version 2.1b) [Online]. Available: http://www.cds.caltech.edu/ murray/amwiki/index.php/Version_2.11b

[45] J. M. Selfridge, and G. Tao, "A multivariable adaptive controller for a quadrotor with guaranteed matching conditions," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, pp. 24-33, Dec. 2013.

[46] K. S. Narenda, and A. M. Annaswamy, *Stable Adaptive Systems*1st ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.