Modelling and Control of an Omni-directional UAV

MODELLING AND CONTROL OF AN OMNI-DIRECTIONAL UAV

BY

ERIC DYER, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2018) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Modelling and Control of an Omni-directional UAV
AUTHOR:	Eric Dyer B.Sc., (Electrical Engineering), McMaster University, Hamilton, Ontario, Canada
SUPERVISOR:	Prof. Shahin Sirouspour
NUMBER OF PAGES:	xv, 132

To my family

Abstract

This thesis presents the design, modeling, and control of a fully-actuated multi-rotor unmanned aerial vehicle (UAV). Unlike conventional multi-rotors, which suffer from two degrees of underactuation in their propeller plane, the choice of an unconventional propeller configuration in the new drone leads to an even distribution of actuation across the entire force-torque space. This allows the vehicle to produce any arbitrary combination of forces and torques within a bounded magnitude and hence execute motion trajectories unattainable with conventional multi-rotor designs.

This system, referred to as the omnicopter, decouples the position and attitude controllers, simplifying the motion control problem. Position control is achieved using a PID feedback loop with gravity compensation, while attitude control uses a cascade architecture where the inner loop follows an angular rate command set by the outer attitude control loop.

A novel model is developed to capture the disturbance effects among interacting actuator airflows of the omnicopter. Given a desired actuator thrust, the model computes the required motor command using the current battery voltage and thrusts of disturbing actuators. A system identification is performed to justify the use of a linear approximation for parameters in the model to reduce its computational footprint in real-time implementation.

The omnicopter benefits from two degrees of actuation redundancy resulting in a control allocation problem where feasible force-torques may be produced through an infinite number of actuator thrust combinations. A novel control allocation approach is formulated as a convex optimization to minimize the omnicopter's energy consumption subject to the propeller thrust limits. In addition to energy savings, this optimization provides fault tolerance in the scenario of a failed actuator.

A functioning prototype of the omnicopter is built and instrumented. Experiments carried out with this prototype demonstrate the capabilities of the new drone and its control system in following various translational and rotational trajectories, some of which would not be possible with conventional multi-rotors. The proposed optimization-based control allocation helps reduce power consumption by as much as 6%, while being able to operate the drone in the event of a propeller failure.

Acknowledgements

I would like to express my deepest gratitude towards my supervisor, Dr. Shahin Sirouspour, who has provided me with crucial guidance, support, and encouragement throughout my MASc studies. I would also like to acknowledge my fellow colleague Mohammad Jafarinasab who's extensive mechanical and controls knowledge provided valuable insight for design decisions. My appreciation extends to my other committee members Dr. T. Davidson and Dr. M. Narimani for their constructive feedback regarding my research. Finally, I would like to give a very special thanks to my family who have made it possible for me to excel in my studies with their endless love and support.

Notation and abbreviations

ADC	Analog to Digital Converter
COM	Center Of Mass
CSV	Comma-Separated Values
DOF	Degree Of Freedom
ESC	Electronic Speed Controller
EKF	Extended Kalman Filter
FC	Flight Controller
FIR	Finite Impulse Response
GPS	Global Positioning System
I/O	Input/Output
IMU	Inertial Measurement Unit
IR	Infrared
LiPo	Lithium Polymer
LQG	Linear Quadratic Gaussian

LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
MSP	MultiWii Serial Protocol
PC	Personal Computer
PID	Proportional, Integral, Derivative
PWM	Pulse Width Modulation
RTOS	Real Time Operating System
SLSQP	Sequential Least SQuares Programming
SMC	Sliding Mode Control
UAV	Unmanned Aerial Vehicle

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The use of small unmanned aerial vehicles (UAVs) has become common over the past decade as the costs to build such systems continue to be driven down while their utility in applications such as aerial imagery, search and rescue, and surveillance becomes more and more valuable. Multi-rotor drones comprise a subset of the UAV family and are of particular interest due to their superior agility compared to their older fixed-wing counterparts. Inexpensive multi-rotor drones have opened the door to explore spaces and capture area footage that was either expensive or near impossible as little as 15 years ago.

Most of these multi-rotor drones have all of their motor-propeller actuators lying in the same plane to compensate against gravity. This design leaves the vehicle with two degrees of underactuation as it cannot provide force along any vector within the plane of the propellers. This issue is largely circumvented by the fact that the vast majority of these vehicles are used in applications that do not require direct interaction with their environment (i.e., inspection, exploration, and surveillance) [1]. However, converting drones with strictly

sensing capabilities to ones that can interact with their environment promises greatly enhanced capability. Such scenarios include augmentation of a utility pole inspection drone to one that can perform repairs. Some research has already explored the feasibility of designs that include robotic arms for aerial manipulation and assembly [2, 3]; however, the inherent underactuation of these systems requires complex control strategies and impedes performance.

Trajectory planning is also limited by the underactuation of multi-rotors as the position and attitude of these systems cannot be controlled independently. This could limit the system from operating in confined spaces and also restrict the degree of movement of a mounted camera. These underactuated systems also suffer from reduced disturbance rejection along the horizontal world plane. If the vehicle would like to provide a lateral force on the environment, it must tilt.

The above shortcomings motivate the exploration of a fully actuated aerial vehicle. Such a vehicle can independently provide any arbitrary force and torque. This means that the vehicle would not need to change its attitude to interact with its environment. In fact, the vehicle could change its attitude to any arbitrary configuration while maintaining the same position. Such a system could reject disturbance forces and torques extremely quickly, limited only by the dynamics of the actuators providing thrust. This thesis presents the design, modeling, control, and testing of a fully-actuated, omni-directional vehicle hereafter referred to as the omnicopter .

1.2 Problem Statement

Although multi-rotor drones offer greater agility than conventional fixed wing and helicopter designs, they still suffer from an implicit under actuation in two degrees along with

minimal actuation in a third degree. This under-actuation results from conventional configurations having all propellers facing in the upwards direction. The design gives the drone significant actuation along the vector perpendicular to the plane of the propellers allowing it to effectively offset acceleration due to gravity. However, the major disadvantage is that the drone loses actuation along any vectors in the plane of the propellers. This means that the drone must tilt in order to move laterally. Another disadvantage is that the drone has very little torque actuation about the axis perpendicular to its propeller plane. This is due to the fact that torque about this axis only results from either pure torque coming from drag forces on the propeller or from change in the angular velocity of the propellers.

The inherent under-actuation of conventional multi-rotors reduces their ability to execute arbitrary maneuvers. For example, the vehicle cannot perform a roll or pitch maneuver without also translating. The drone is also more susceptible to external disturbances particularly in the directions of under-actuation. Drones are most often used in sensing applications that do not require them to directly interact with their environment. However, the future of this technology will likely explore practical solutions for connecting a robotic arm to the drone or installing some other kind of device that creates a disturbance force. One recent application is the installation of a fire hose onto a tethered heavy-lift drone to de-ice large wind turbines [4]. When a liquid is sprayed from the nozzle, the drone must tilt to counteract the reaction force and maintain its position. A typical proporitional, integral, derivative (PID) controller will experience some drift in position of the drone since it takes some finite amount of time for the drone to tilt and offset the disturbance. One solution to improve this performance involves modeling the disturbance in the control or by trying to predict its effect. Unfortunately, the development of such model-based controllers can be complex and require significant computational resources not available on board a small

UAV.

The proposed solution is to provide a degree of actuation in the directions that conventional multi-rotor drones have none. This fully actuated multi-rotor vehicle would be able to provide the three forces and three torques simultaneously. As a result, the vehicle would have the ability to reject any disturbance without needing to alter its orientation. Such a solution could reduce the delay to reject disturbance and allow a simple PID to replace complex control schemes required for conventional multirotors. Full actuation also implies that the UAV could follow any arbitrary trajectory opening the door for many interesting solutions in trajectory planning. The addition of lateral actuation opens the door for aerial vehicles that can interact with their environment. Multi-rotors no longer need to be largely confined to sensing applications, but rather could manipulate objects in their surroundings, greatly extending their usefulness in remote or dangerous situations.

1.3 Summary of Thesis Contributions

This thesis extends upon the work done in [5] to design, model, and control a fully-actuated multi-rotor UAV. This section outlines the major contributions presented in the work.

1.3.1 Formulation of Optimal Propeller Configuration

The propeller configuration is formulated as an optimization problem to find a solution that provides an even distribution of actuation over the entire force-torque space. This means that the transformation matrix, which maps the actuator thrusts to the net force/torque vector (hereafter referred to as the *Jacobian*), is well-conditioned such that it evenly spans over the entire force-torque space. The final solution involves eight actuators oriented such

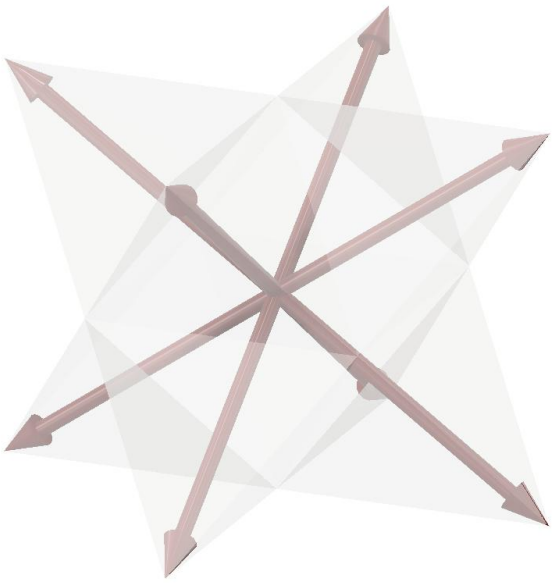

Figure 1.1: Stella octangula thrust configuration of the actuators

that their thrust vectors produce a stella octangula as shown in Figure 1.1. This shape can also be formulated by combining two tetrahedrons such that one tetrahedron is rotated 180 degrees with respect to the other [6]. This solution is intuitively satisfying due to the symmetricity of the design. Actuators with parallel thrust vectors are located on colinear arms of the omnicopter .

1.3.2 Propeller and Airflow Models

A propeller model is introduced based on work done in [7] to map the pulse width modulation (PWM) thrust commands to actual actuator forces. This model is identified and validated using a custom test bed apparatus. It is then extended to include the effects of interacting propeller airflows on overall propeller thrust. This novel model can determine the required PWM signals necessary to produce a desired set of actuator thrusts considering interactions by surrounding disturbing actuators. An experimental justification is provided

to allow for the simplification of this model to be linearly dependent on disturbing actuator forces. This reduces the computational cost of the model allowing it to be implemented in real time. Finally, the model's accuracy is evaluated experimentally.

1.3.3 Redundancy Resolution in Control Allocation

Two separate PID controllers are used to control the position and attitude of the omnicopter independently. The position controller includes a gravity feed-forward term and responds to position error by generating linear force. The attitude controller utilizes a cascaded architecture, where the outer-loop uses attitude error to generate a desired angular velocity, ω. The inner-loop controller follows this reference ω by producing a torque command.

The omnicopter's actuator configuration involves the use of eight propellers, providing over-actuation in two degrees. This redundancy means that a desired force-torque couple, if feasible, may potentially be produced by an infinite combinations of propeller thrusts. The over-actuation produces a 2D nullspace within the actuator input space, which may be traversed without affecting the output force-torque. This redundancy is initially resolved by minimizing the 2-norm of the vector of actuator thrusts. This approach is also taken in [5] and is used to verify the control architecture. A novel approach is then proposed, which involves the formulation of a convex constrained optimization to minimize the power consumption of the drone. The advantages of the new approach are that its cost function is more meaningful from an operational point of view, and that it explicitly incorporates the thrust limits in finding a solution to the problem. The optimization is then expanded so it can handle the case in which a feasible solution to the original problem does not exist, i.e., the combination of requested force/torque vectors cannot be produced. In such case, a

solution would be found that would minimize the difference between the actual requested and actual force/torque. The new formulation yields energy savings up to 6% compared to the 2-norm solution. The new formulation also provides a convenient way to deal with actuator failures by adjusting the limits on the actuator forces on the fly, and finding the best possible solution in such a scenario. An sequential least squares programming (SLSQP) solver is used to solve the problem in real time at 340Hz and experimental results validate the approach.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, a literature review is conducted to explore relevant contributions in the area of aerial robotics. Chapter 3 describes the steps taken to choose a propeller configuration that provides even distribution of actuation over the entire force-torque space. A model is then proposed to relate motor PWM commands to actual propeller forces. This model is identified and validated experimentally. Design of a conventional PID controller with gravity feed-forward compensation is presented in Chapter 4. The control system response is simulated in Chapter 5, using a model that captures the most significant aspects of the system dynamics. In Chapter 6, several control redundancy resolution strategies are presented. Experimental results demonstrating the capabilities of the omnicopter are presented in Chapter 7. Chapter 8 takes a practical look into the implementation of the test bed used to perform the experiments and collect the results. Finally, Chaper 9 concludes the thesis and suggests areas for future work.

Chapter 2

Literature Review

The literature review serves to present a concise background in the areas most relevant to this thesis. This foundational work provides a necessary starting point, while a critique of these works catalyzed the development of novel solutions that are presented in the following chapters.

2.1 Applications of Multi-rotor UAV Technology

The pace of technological advancement continues to accelerate at a startling rate. Computing devices that required a small building 50 years ago can now fit in the the palm of your hand. Due to the economy of scale, automation, and global competition, the cost associated with the production of these devices continues to be driven down providing incentive to add intelligence to even the simplest of devices [8]. Before the 2000s, computing platforms were simply too large and expensive to practically implement on multi-rotor platforms. As a result, research in this technology was stunted compared to its more mechanically-oriented counterpart, the helicopter. By the late 2000s, advances in computing technology

started to allow for the design of inexpensive, light flight computers capable of speeds necessary for stable flight [9] [10]. With the advent of credit-card sized single board computers, modern multi-rotor UAVs can perform computationally intensive onboard algorithms including video encoding, obstacle avoidance, and autonomous navigation [11] [12].

As processing power becomes more accessible and sensing technologies constantly improve, researchers continue to push the envelope to find new and innovative applications for multi-rotor technology. Compared to their fixed-wing counterparts, multi-rotors offer a new dimension of agility in the air at a fraction of historical cost. The evolution of the inexpensive drones has inspired a whole new class of applications for UAVs ranging from low-cost aerial photography, to agricultural and forest monitoring [13] to search and rescue operations [14] [15] and even to fog dissipation for city corridors during rush hour [16].

2.2 Operating Principles

Within the multi-rotor family, one of the most minimal designs is the quadrotor. Figure 2.1 depicts a common quadrotor configuration. Here, the frame R_b is assigned to the body of the quadrotor and R_i is the inertial (or world) frame. It may be noted that rotations about the x, y, and z axes will be referenced as roll, pitch and yaw respectively. The conventional quadrotor configuration boasts a simple design while elegantly balancing a number of considerations. First, all four propellers generally lie within a single plane, defined as the xy plane of the body frame. Aligning all propeller thrust vectors in this way provides the drone with a high degree of actuation in its positive z direction. Since the quadrotor's z axis is generally closely aligned with the inertial z axis, it can effectively cancel out gravity. Second, the propellers are normally found approximately equidistant from its center of mass

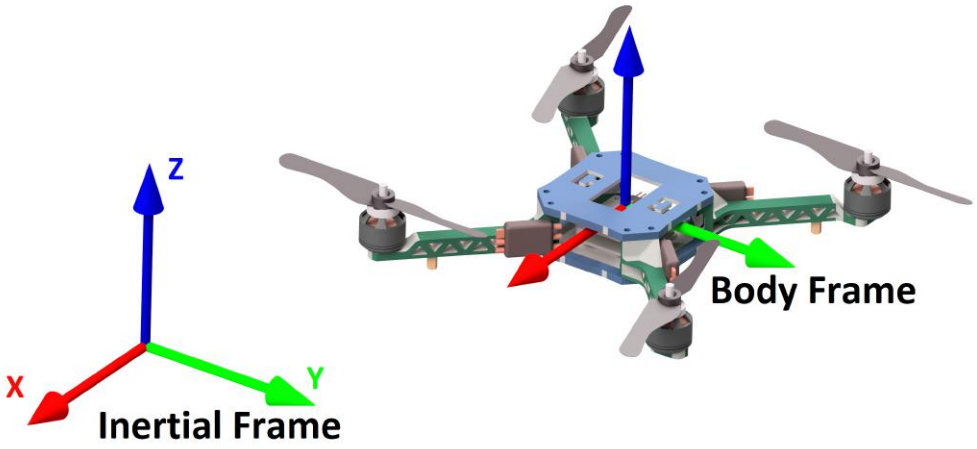

Figure 2.1: Frame assignments for the inertial (world) frame and body (quadcopter) frame in a conventional quadcopter

allowing for an even distribution of torque about the x and y axes. Third, propellers sharing a diagonal rotate in the same direction but opposite to those sharing the other diagonal. This provides two major advantages. A yaw actuation can be achieved by spinning two propellers on the diagonal faster; the torques created by each propeller are equal and opposite and therefore cancel. However, the pure torque produced by the propeller drag forces are greater for the faster spinning propellers creating a net torque around the z axis. Since all four propellers have similar angular velocity during normal operation, having half the propellers spinning in each direction also limits gyroscopic effects. Note that these design principles can be applied to other systems of the multirotor family such as conventional hexacopters and octocopters.

A multi-rotor usually has a central flight computer that serves as the brain of the vehicle. The flight computer can receive sensor data from devices like an inertial measurement unit (IMU), global positioning system (GPS), and cameras, among others. The IMU is among the most common sensors found on a multi-rotor as it provides critical information

about the vehicle's attitude and angular velocity. Brushed or brushless DC motors are the most common form of propdrive on a multi-rotor. These motors can be controlled by the flight computer using a motor driver circuit most commonly known as an electronic speed controller (ESC). Multi-rotors demand significant power draws for their size and therefore generally run on high performance lithium polymer (LiPo) batteries. The flight computer is responsible for taking sensor input along with a given command (either from a teleoperator or other trajectory planner) to come up with a desired force-torque vector denoted as $\boldsymbol{\zeta}$. In the case of a conventional multi-rotor with n propellers, $\boldsymbol{\zeta}$ is a 4×1 vector with the first element being thrust along the body z axis, f_z, and then next three elements representing the torques about the x, y, and z body axes, respectively. $\boldsymbol{\zeta}$ can be mapped to n individual propeller thrusts, f_i, $i = 1, \cdots, n$, through a Jacobian J such that

$$\boldsymbol{\zeta} = J\mathbf{f} \tag{2.1}$$

$$\begin{bmatrix} f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = J \times \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{2.2}$$

where f_i, $i = 1, \cdots, n$ are the propeller thrusts.

Before deriving the Jacobian, it must be noted that the pure torque, τ_{drag}, produced by a propeller's drag as it spins, is related to the propeller's thrust force f_{thrust} through the

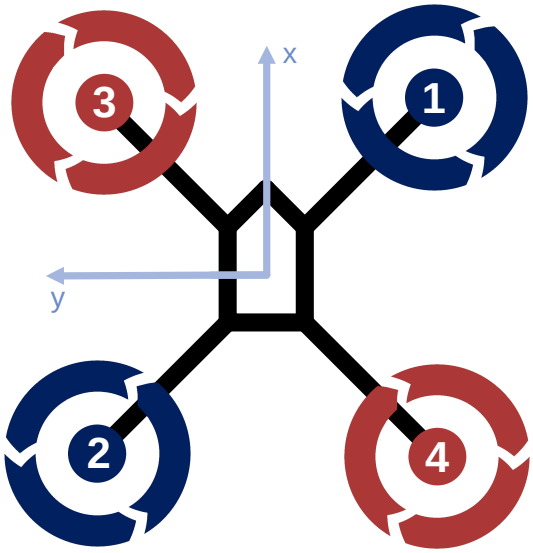

Figure 2.2: Motor number assignments for an X-frame quadrotor

proportionality constant k [17].

$$\tau_{drag} = k f_{thrust} \tag{2.3}$$

The derivation of the Jacobian now becomes a simple geometry problem and can be obtained from inspection of the particular drone configuration. An example Jacobian is presented for a quadrotor in a X configuration depicted in Figure 2.2 with arm length r between the center of each propeller and the vehicle's center of mass (COM).

$$J = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -r/\sqrt{2} & r/\sqrt{2} & r/\sqrt{2} & -r/\sqrt{2} \\ -r/\sqrt{2} & r/\sqrt{2} & -r/\sqrt{2} & r/\sqrt{2} \\ -k & -k & k & k \end{bmatrix} \tag{2.4}$$

2.3 Control Strategies

A diverse group of control algorithms have been applied to multi-rotor vehicles over the past couple decades. These algorithms fall into the categories of linear (or linearized) controllers and nonlinear model-based controllers. Since most conventional multi-rotors operate in their linear range, the first class of controllers have proven to be robust and simple to implement. Higher performance may be achieved through more complex model-based approaches; however, challenges can arise from increased sensitivity to noise and modeling errors.

2.3.1 Linear Control Strategies

The simplicity and robustness of PID controllers to uncertainty and disturbance make them by far one of the most common control algorithms used in industry today. A common cascaded position control architecture for a conventional multi-rotor is shown in Figure 2.3. A trajectory planner sends a desired position to the controller. The outer PID acts on the position error and maps this to an attitude and thrust command through a transform M. The transform M must at least know the yaw of the multi-rotor to produce appropriate roll and pitch commands to track the desired x and y position. The attitude loop tracks these commands by generating an angular velocity command proportional to the angular position error, i.e., the difference between the desired and actual attitudes. The innermost PID loop tracks this command using angular velocity feedback from the multi-rotor's gyroscope.

Each of the gains in the PID controller may be tuned either through a mathematically vigorous process, or more commonly, through experimentation given a good understanding of the system dynamics. Although simple, the PID sets an industry baseline for control algorithm performance due to its reliability and robustness to noise and modeling error

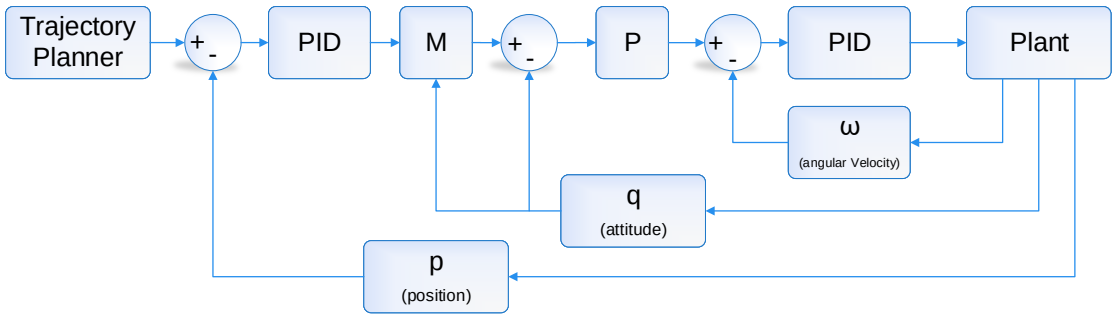

Figure 2.3: A common cascaded PID architecture for position control of a conventional multi-rotor vehicle

across a diverse range of applications.

Steps have been taken to enhance the PID algorithm while preserving computational simplicity. In [18], a disturbance observer is developed to try and eliminate external disturbances and allow treatment of the multi-copter using its nominal model. The research demonstrates that introduction of this new term can reduce some unmodeled errors, such as gyro drift, by as much as 75%. In [19], a robust compensator is added to a standard PD controller to minimize the influence of uncertainties on the controller performance .

Work has been done to implement linear quadratic regulator (LQR) and linear quadratic Gaussian (LQG) algorithms as an alternative to conventional PIDs. [20] [21] [22]. Although only suitable for linear systems, LQR and LQG algorithms have been developed to provide optimal solutions for multi-rotor operating in their linear approximate ranges in both in simulation and the real-time experiments.

2.3.2 Non-linear Model-based Control Strategies

To overcome some of the limitations of linear algorithms, non-linear model-based control strategies have been developed to take advantage of the full non-linear dynamic model of a

multi-rotor. Sliding mode control (SMC) offers an efficient and systematic way to design theoretically robust controllers for plants with high-order nonlinear dynamics [23]. Although this approach can stabilize a multi-rotor [24], the inherent chattering phenomenon produced by such controllers can be problematic for multi-rotor vehicles even with their high electro-mechanical bandwidth. Model predictive control (MPC) has the ability to directly predict the future behavior of the system and thereby choose control outputs that minimize tracking error. The work in [25] presents a switching MPC that can effectively reject wind disturbances. Other work done in [7] proposes a virtual decomposition technique to break the modeling problem into smaller simpler parts. This technique can prove useful when trying to control a high-dimensional robotic system [26] [27]; for example, a UAV with a robotic arm attached. Unfortunately, model-based approaches often suffer from a greater sensitivity to modeling errors and are often much more computationally complex than their linear counterparts.

Simple linear controllers have proven to perform adequately for touchless applications where the drone does not interact with its environment. However, in environments that present significant disturbance forces, a model of the disturbance is often needed to achieve satisfactory disturbance rejection.

2.4 Drawbacks of Underactuated Aerial Vehicles

Simplifying the dynamics of a multi-rotor to a single rigid body allows the translational dynamics to be written as

$$m\ddot{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\boldsymbol{u} \tag{2.5}$$

where the contributing dynamics to position are gravity and the control input force $\boldsymbol{u}$. This force is produced along the body z axis of the multi-rotor. A rotation matrix R, derived from the attitude of the multi-rotor, expresses this thrust vector in the world frame. As shown in Figure 2.3, the outermost position control loop employs a PID to track a desired position command. The difference between the desired and measured position of the multi-rotor is taken to produce an error vector, $\boldsymbol{e}(t)$, which is used to derive the control input

$$u(t) = k_p\boldsymbol{e}(t) + k_i \int_0^t \boldsymbol{e}(x)dx + k_d\frac{d\boldsymbol{e}(t)}{dt} \tag{2.6}$$

Since a conventional multi-rotor can only produce a force vector that is perpendicular to the plane in which the propellers lie, $\boldsymbol{u}(t)$ is achieved by appropriately rotating the vehicle. This establishes a dependence between translational force and the multi-rotor's attitude [28] [29]. This dependence is an unfortunate characteristic of conventional multi-rotor systems, leading to an inherent underactuation. This means that the position and attitude of the drone cannot be controlled independently, which prevents the multi-rotor from executing arbitrary accelerations and trajectories. This underactuation is a direct result of the design choices mentioned in Section 2.2 and can cause challenges in control, especially when trying to reject disturbances that lie in the body xy plane.

It can be seen how a fully actuated vehicle, one that could provide any force-torque

vector, could greatly simplify the control problem by decoupling the position and attitude control loops. This offers improved disturbance rejection and opens the door to new possibilities in trajectory planning [30].

2.5 Unconventional Aerial Vehicles

This section explores some unconventional designs with full actuation and compares the benefits and shortcomings of such configurations.

Physical interaction between a multi-rotor and its environment presents a particularly complex set of challenges due to their inherent instability and underactuation. Full actuation of the vehicle allows it to produce an arbitrary linear and angular acceleration, giving it the capability to counteract any force-torque disturbance it encounters in its environment. Such fully-actuated designs entail unconventional actuator configurations that rotate the propellers off of the traditional vehicle xy plane depicted in Figure 2.1 [31], [32], [33]. These papers propose a variety of mechanical solutions that might accomplish such a tilting action and claim to have developed robust control strategies verified through extensive simulation.

True validation of a fully-actuated design is demonstrated in [34] with the realization of a working hexacopter prototype. The vehicle has six equidistant arms in its xy plane. Full actuation is achieved by tilting each actuator about the arm using rigid adapters. A rigid link is affixed to one end of the multi-rotor to demonstrate that it can successfully interact with its environment by executing various force commands on external objects with the link. A mechanical extension of the fully actuated hexacopter is introduced in [35], where each arm of the hexacopter may be tilted to control the propeller configuration in real time. This work demonstrates the ability to turn a conventional hexacopter into a fully actuated

system. However, the six additional motors that must be added to the system increases the vehicle weight and mechanical complexity, introducing more points of failure. It may also be noted that each actuator can only move through 720 degrees of rotation before the actuator cables are fully wound around the arms. This issue may be mitigated in the future using a slip ring, however, this adds further complexity and weight to the multi-rotor.

Another flavour of fully-actuated aerial vehicle is presented in [36]. Here, the actuator configuration involves two counter-rotating coaxial propellers that generate the main thrust vector. Mounted about the perimeter are three variable-angle ducted fans in an equilateral triangle configuration. The prototype demonstrates decoupling of position and attitude control by performing translations without needing to roll or pitch. However this prototype is not able to demonstrate tracking of arbitrary attitude trajectories. Since the main thrust component comes from the coaxial propellers, the vehicle would have difficulty maintaining orientations where these propeller thrusts are perpendicular to the gravity vector.

Yet another strategy uses only two coaxial propellers to emulate full actuation [37]. Precise modulation of the motor shaft torques excites a specific tilting response in a propeller, which is connected to the motor via a passive teetering hinge. As a result, the angle of each propeller can be independently and precisely controlled. This system benefits from the ability to tilt its actuators without the need for an extra motor or complex variable pitch mechanism. However, the achievable tilt of the propeller is limited, which prevents the vehicle from achieving large forces withing the body xy plane.

This thesis proposes the design and control allocation of a fully actuated aerial vehicle that can provide an even distribution of actuation across the entire force-torque space. The work done in [5] achieves such an actuator configuration using eight actuators distributed about a cube frame. A preliminary prototype demonstrates that the vehicle is capable of

flying through any arbitrary position and orientation trajectory. The work presented in this thesis seeks to extend upon this design by investigating further into the choice in propeller configuration as well as implementing a model to capture disturbance airflows between propeller streams. Furthermore, [5] resolves propeller redundancy through a simple 2-norm optimization which does not exploit the vehicle's full force-torque capabilities. This thesis formulates and implements a real-time energy optimal control allocation, which is robust to an actuator failure, and takes advantage of the omnicopter's full force-torque capability.

2.6 Sensing Strategies for Position and Attitude Determination

When controlling a multi-rotor UAV, the states that need to be fed back to the controller are position, linear velocity, attitude, and angular velocity. Some of these states can be measured directly by a sensor, while others require a fusion of multiple sensors to achieve the best results.

The position feedback for this research is obtained using an OptiTrack motion capture system as depicted in Figure 2.4. The system emits infrared (IR) light into the work space to illuminate passive IR reflective markers on the drone. Based on an understanding of the relative marker positions, the optitrack system can calculate the position and attitude of the drone with sub-millimeter accuracy at at a rate of 120 times per second. Linear velocity feedback is obtained through simple numerical differentiation of these position measurements. Unfortunately, this can introduce significant noise and thus requires low-pass filtering before it can be used [38]. Such filtering may be achieved using a simple moving

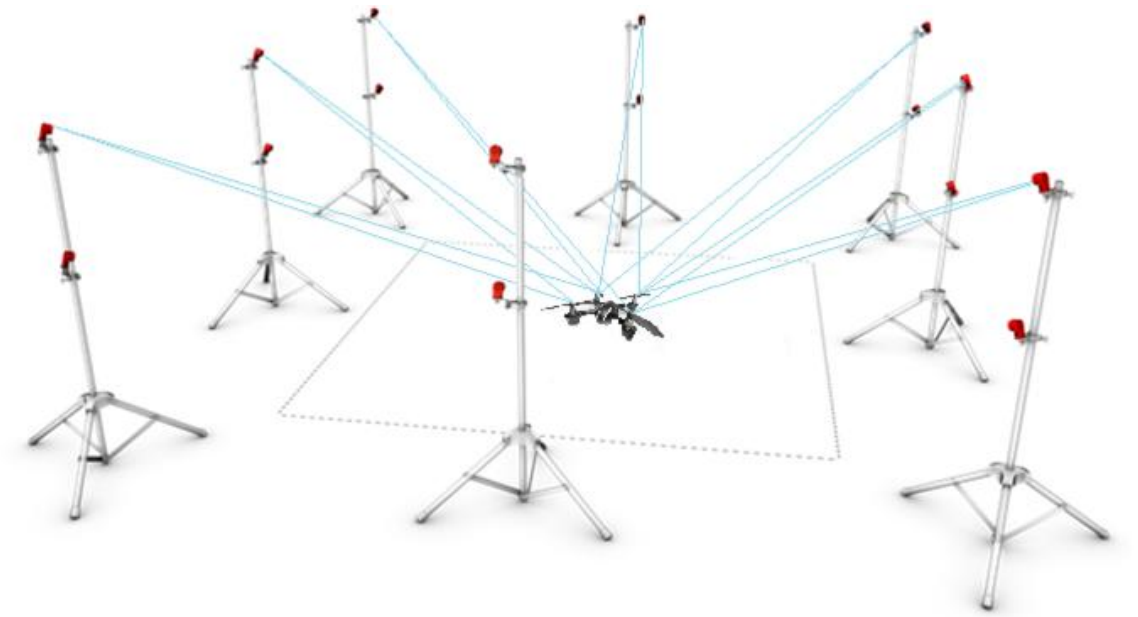

Figure 2.4: OptiTrack motion capture system

average filter or a tuned low-pass finite impulse response (FIR) filter. In practical applications where expensive motion capture systems are not available, rough position estimates can be obtained outdoors via GPS, while finer control can be achieved using optical flow technology [39] coupled with ultrasonic sensors for altitude control. Optical flow employs a ground-facing camera to track ground textures and visible features to determine vehicle velocity. As single-board computers continue to become more capable, there is increasing interest in single and stereo camera setups for intelligent autonomous UAV navigation and obstacle avoidance [40] [41].

Attitude determination plays a critical role in ensuring the drone's stability. Both attitude and angular velocity are derived from an onboard IMU that commonly consists of an accelerometer and a gyroscope. While angular velocity can be directly measured from the

gyroscope, the drone's attitude must be estimated by fusing together data from both sensors using an algorithm such as an extended Kalman filter (EKF) [42]. This is due to the fact that gyros tend to drift overtime and the orientation derived from the accelerometer becomes inaccurate when it experiences accelration other than gravity. Gyros are capable of high bandwidth in attitude estimation (approximately 1kHz feedback rate). The accelerometer provides a correction to these gyro measurements at a rate of 100Hz to prevent drift in the estimation [43]. Although the accelerometer can correct roll and pitch measurements, magnometer measurements should be included in the sensor fusion to correct gyro drift around the world z axis [44]. Unfortunately, magnometer measurements are unreliable for indoor applications as the magnetic field is distorted by the building. This feedback must be provided by other means for indoor applications. Such replacement sensing solutions include on-board vision based systems or off-board motion capture systems.

Chapter 3

System Design, Kinematics, and Dynamics

This chapter presents a rationale behind the design choices made for a fully actuated multi-rotor vehicle. First, a specification is laid out for how the vehicle should perform. This leads to the formulation of an optimization problem to determine a suitable propeller configuration. A mapping is then derived to relate individual propeller forces to the system's net force/torque. Finally, the simplified system kinematics and dynamics of the omnicopter are presented, laying a foundation for control design.

3.1 Propeller Configuration

The choice in propeller configuration for the omnicopter sets it apart from conventional multi-rotor drones, as the propellers are no longer constrained to the same approximate plane. The elimination of this constraint allows for exploration of more flexible, fully-actuated propeller configurations.

3.1.1 Design Requirements

The propeller configuration should be chosen to provide an even distribution of force and torque over the entire force-torque space. A symmetric force-torque distribution should allow the omnicopter to follow a position trajectory and cancel gravity regardless of its orientation.

Although the theoretical minimum for full actuation is six propellers, the use of eight propellers allows for a mechanically simplified cuboid vehicle frame. It yields two degrees of redundancy, which can be exploited for more robust and improved control. As shown in Figure 3.1, the omnicopter frame will be a reinforced cube consisting of a central hub upon which all electrical components can be mounted. Eight arms extend from the central hub out to the vertices of the cube. One motor-propeller actuator is located on each of these arms equidistant from the center.

In addition to being straightforward to assemble and mechanically robust, the symmetry of the proposed frame ensures that the inertial tensor of the omnicopter can be approximately represented by a scaled identity matrix, allowing the rotational dynamics to be similar around all three axes.

3.1.2 Optimization of Propeller Configuration

Based on the proposed frame design, the positions of all eight actuators are constrained since one actuator must lie on each arm and be equidistant from the center of geometry of the frame. Two of the rotational degrees of freedom are also constrained since the bottom face of each actuator must be parallel to a face on a rectangular arm. The only degree of freedom left is the rotation of the arm (and consequently the actuator) by an angle θ about its long axis.

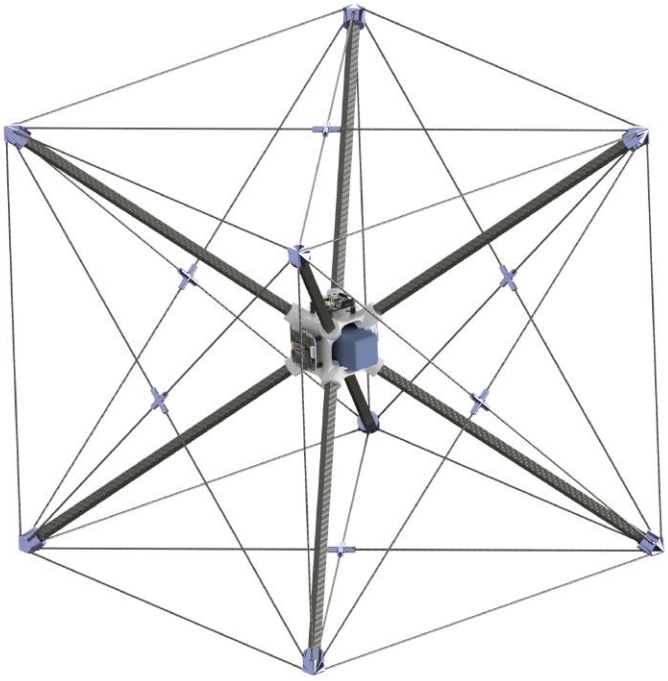

Figure 3.1: Proposed frame for the omnicopter

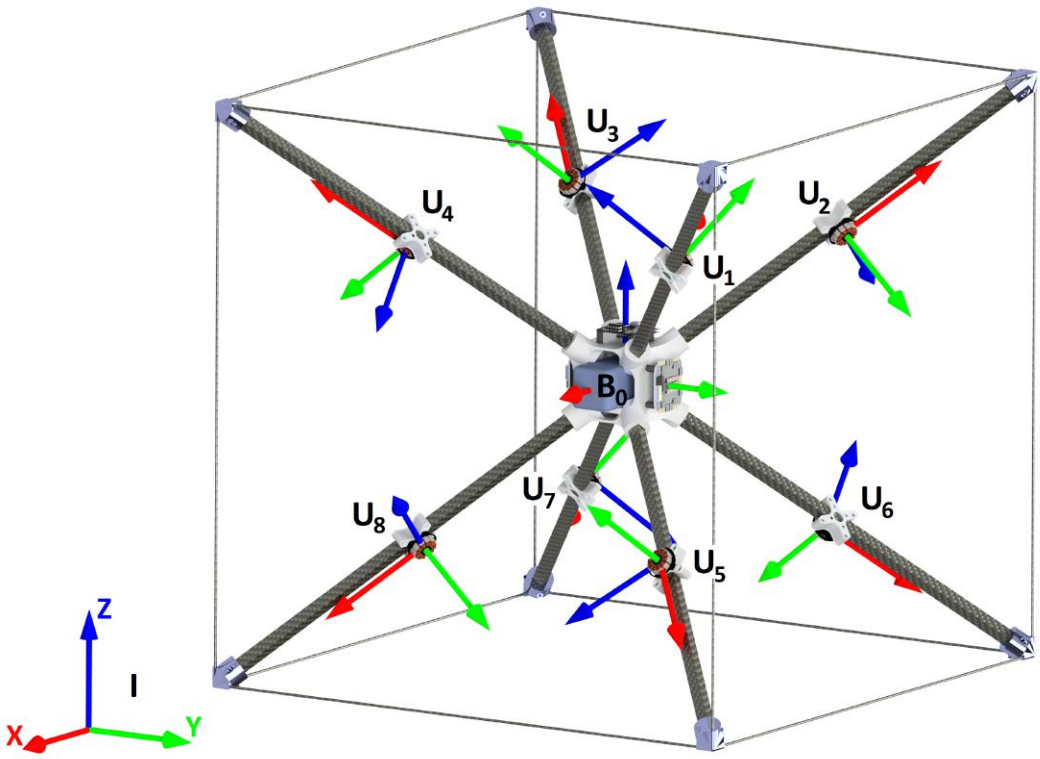

Figure 3.2: Frame definitions for the omnicopter

Figure 3.2 shows the frame assignments used to represent the omnicopter . The inertial, or world, frame I provides a reference for the omnicopter body frame B_0. A frame is also attached to each actuator U_i such that the y_i axis points in the direction of positive thrust and the x_i axis points towards an outer vertice of the omnicopter frame. Note that for the purposes of display, the actuator frame arrows are slightly offset so the x axes are not hidden within the carbon fiber arms.

To determine the angle θ_i by which each actuator U_i should be rotated, an optimization problem is formulated using $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_8]$ as the decision variables. Each frame U_i may be rotated about its respective x axis by θ_i. A rotation of zero degrees is defined when a

frame U_i is oriented such that its positive y axis sits in the plane formed by the z axis of B_0 and the corresponding arm.

The optimization should seek to find a set of $\boldsymbol{\theta}'s$ that produce a well-balanced distribution of force in all directions. This can be achieved by maximizing the achievable force along the weakest direction of actuation. This requires the formulation of a three level optimization. The outer level searches for a $\boldsymbol{\theta}$ that maximizes a cost function $C(\theta)$, the maximum force produced in the weakest direction. The outer level can be written as

$$\begin{aligned} &max\ C(\theta_1, \cdots, \theta_8) \\ &s.t.\ 0 \leq \theta_i < 2\pi \end{aligned} \tag{3.1}$$

For a given configuration, $\boldsymbol{\theta}$, the cost function, C, returns a scalar metric that proportionally reflects the magnitude of maximum achievable force along the direction of least actuation.

The derivation of this cost function now leads to the second level of optimization through an exhaustive search. Given a configuration, $[\theta_1, \cdots, \theta_8]$, this loop iterates through 625 direction vectors evenly distributed over the unit sphere to determine the direction that has the smallest maximum force capability.

The third level of optimization finds the maximum achievable force in the unit force direction $\boldsymbol{\zeta}$. The formulation starts by splitting the output force-torque term $\boldsymbol{\zeta}$ of Equation 2.1 into a unit vector and scaling term α.

$$J\mathbf{f} = \alpha\boldsymbol{\zeta}' \tag{3.2}$$

Here it may be noted that $\boldsymbol{\zeta}'$ is a unit force vector, where the three torque components of the vector are zero. J is the Jacobian transform, which is derived from the decision variables, $[\theta_1, \cdots, \theta_8]$, produced by the outermost optimization. The derivation of this Jacobian is presented in the Appendix. The scaling term α is the parameter that must be maximized to determine the maximum force output that may be achieved in the $\boldsymbol{\zeta}'$ direction.

This optimization takes advantage of the redundancy in actuation by searching through the two-dimensional nullspace of the actuator input space. Any vector in this null-space can be added to the actuator force vector $\boldsymbol{f}$ without changing the output force-torque $\boldsymbol{\zeta}$. Since the constraining factor of the omnicopter's force output is the limited thrust generation on each of its actuators, the minimum norm solution may not provide the maximum possible force output for a given direction. Rather, it is often necessary to exploit the nullspace to find this maximum possible output force.

The actuator force vector, $\boldsymbol{f}$, is therefore parameterized as

$$\boldsymbol{f} = \alpha J^{\dagger}\boldsymbol{\zeta}' + \gamma_1\boldsymbol{null}_1 + \gamma_2\boldsymbol{null}_2 \tag{3.3}$$

Here, two other decision variables, γ_1 and γ_2, give the input solution the freedom to traverse the nullspace along the orthogonal vectors $\boldsymbol{null}_1$ and $\boldsymbol{null}_2$ in search of a maximum α. The dagger symbol, † is used to perform the Moore-Penrose pseudoinverse on the Jacobian. Since these component vectors of the nullspace do not contribute to the output

force/torque vector, they can simply be added to the input vector.

An inequality constraint is used to limit the magnitude of each actuator within unity. It may be noted that selection of unity for the constraint is arbitrary and does not affect the optimization results. The innermost optimization can then be written as

$$\begin{aligned} & max \quad \alpha \\ & s.t. \begin{bmatrix} J^{\dagger}\boldsymbol{\zeta} & \boldsymbol{null}_1 & \boldsymbol{null}_2 \\ -J^{\dagger}\boldsymbol{\zeta} & -\boldsymbol{null}_1 & -\boldsymbol{null}_2 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \gamma_1 \\ \gamma_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \end{aligned} \tag{3.4}$$

This three-level optimization is both non-linear and non-convex, meaning that solving for the global optimum is not guaranteed. To ensure adequate exploration of the local minimas, the optimization is run multiple times with different starting points for $\boldsymbol{\theta}$ that evenly span the eight dimensional input configuration space. Since the choice of starting configuration is an eighth order problem, the resolution between different starting configurations is very coarse due the dimensionality of this space.

Although there are many local minima, the configuration selected for the omnicopter is

shown in Figure 3.2. The chosen solution for $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta} = \begin{bmatrix} -\frac{\pi}{4} \\ \frac{3\pi}{4} \\ -\frac{\pi}{4} \\ \frac{3\pi}{4} \\ -\frac{3\pi}{4} \\ \frac{\pi}{4} \\ -\frac{3\pi}{4} \\ \frac{\pi}{4} \end{bmatrix} \tag{3.5}$$

This particular configuration was selected due to its intuitively satisfying geometry. Actuators lying on collinear arms produce thrust in the same direction when spinning in opposite directions. If each pair of actuators produces the same amount of thrust, the resulting net torque, both from the moment arm and propeller drag torques, should be zero. Pairs of propellers spinning with similar velocities will also benefit from minimal gyroscopic effects since the angular velocities cancel. Often this is a reasonable assumption since required torques are small and pairs of propellers spin at the same angular velocity for pure force commands. It may also be noted that work done in [5] also supports the use of this actuator configuration.

Figure 3.3 provides a visual representation of the maximum available force capabilities of the omnicopter . A side-by-side comparison shows that in directions of maximal actuation, the nullspace optimization and minimum norm produce the same solution. In these directions, the omnicopter is able to produce a force that is about 4.6 times that of a single actuator's maximum thrust. However, the nullspace optimization excels in directions of

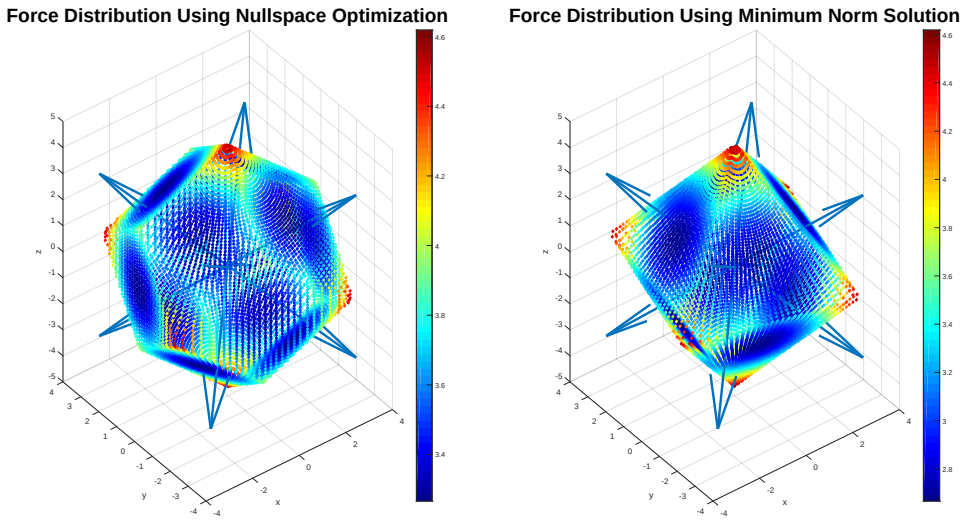

Figure 3.3: Force distribution for the propeller configuration

minimum actuation where it produces a force factor of 3.3 compared to the 2.7 from the minimum norm solution.

3.1.3 Analysis of the Jacobian Mapping

The Jacobian is used to verify that the selected configuration provides an even distribution of actuation over the entire force-torque space. The singular value decomposition can be performed on the Jacobian

$$J = U\Sigma V^T \tag{3.6}$$

where the right singular vectors in V are the principal directions with corresponding singular values in the diagonal Σ matrix. These singular values should be similar in magnitude to

ensure the Jacobian is well conditioned and there is ample actuation over the entire force-torque space. The Jacobian derived in the Appendix, and presented in Equation A.14, can be decomposed as

$$J = \begin{bmatrix} 0 & 1.00 & 0 & 0 & 0 & 0 \\ -0.02 & 0 & -1.00 & 0 & 0 & 0 \\ -1.00 & 0 & 0.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.16 & -0.99 \\ 0 & 0 & 0 & 0 & 0.99 & -0.16 \\ 0 & 0 & 0 & -1.00 & 0 & 0 \end{bmatrix} \times$$

$$\begin{bmatrix} 1.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.37 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.38 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.38 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.36 & 0.34 & -0.35 & 0.36 & -0.35 & 0.36 & -0.36 & 0.34 \\ -0.48 & 0.13 & 0.48 & -0.13 & 0.48 & -0.13 & -0.48 & 0.13 \\ -0.12 & -0.49 & 0.14 & 0.48 & 0.14 & 0.48 & -0.12 & -0.49 \\ -0.35 & 0.35 & -0.35 & 0.35 & 0.35 & -0.35 & 0.35 & -0.35 \\ -0.50 & -0.02 & 0.50 & 0.02 & -0.50 & -0.02 & 0.50 & 0.02 \\ -0.02 & 0.50 & 0.02 & -0.50 & -0.02 & 0.50 & 0.02 & -0.50 \\ -0.50 & -0.50 & -0.50 & -0.50 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.02 & 0.50 & 0.50 & 0.50 & 0.50 \end{bmatrix} \tag{3.7}$$

giving a condition number of 4.32. However, the block structure of the U matrix from the decomposition reveals that the first three singular values are associated with the output force and the last three are related to the output torque. Therefore, it can be concluded that the drone's force and torque outputs are evenly distributed in their respective spaces. It is worth noting that this analysis ignores the limits on the actuator thrusts, which in practice impact the actual force and torques distributions, e.g. see Figure 3.3.

3.2 System Kinematics

For the purposes of attitude estimation, control, trajectory planning, and description of the system dynamics, the *quaternion* representation of rotation is used instead of Euler angles to avoid a phenomenon known as gimbal lock. This phenomenon occurs when a rigid body

approaches an orientation where a change in the target space (rotation) is not realized in the source space (Euler angles). This happens when two axes of rotation become collinear yielding an infinite number of correct rotational combinations between these two axes. In contrast, the quaternion representation provides a compact, mathematically eloquent, and singularity free description of orientation [45].

The unit quaternion, $\mathbf{q}$, can be used to represent a rotation and is defined as

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{3.8}$$

where $\|\mathbf{q}\| = 1$ and $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are the fundamental quaternion units that obey the following rules.

$$\begin{aligned} \boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} &= -1 \\ \boldsymbol{ij} = \boldsymbol{k}, \quad \boldsymbol{ji} &= -\boldsymbol{k} \\ \boldsymbol{jk} = \boldsymbol{i}, \quad \boldsymbol{kj} &= -\boldsymbol{i} \\ \boldsymbol{ki} = \boldsymbol{j}, \quad \boldsymbol{ik} &= -\boldsymbol{j} \end{aligned} \tag{3.9}$$

If a quaternion is used to represent some orientation, it may be rotated about its current axes by post multiplying by a second quaternion. The multiplication of two quaternions is achieved using the *Hamilton product*, which is represented by the $\otimes$ operator and defined

as

$$\begin{aligned}
&(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \otimes (r_0 + r_1\mathbf{i} + r_2\mathbf{j} + r_3\mathbf{k}) \\
= \quad & q_0r_0 - q_1r_1 - q_2r_2 - q_3r_3 + (q_0r_1 + q_1r_0 + q_2r_3 - q_3r_2)\mathbf{i} + \\
& (q_0r_2 - q_1r_3 + q_2r_0 + q_3r_1)\mathbf{j} + (q_0r_3 + q_1r_2 - q_2r_1 + q_3r_0)\mathbf{k}
\end{aligned} \tag{3.10}$$

A quaternion, $\mathbf{q}$, may also be converted to a rotation matrix expressed as

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 + 2q_0q_2 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{3.11}$$

This form of the quaternion will be used for the remainder of the chapter.

3.3 System Dynamics

In this section, the dynamics of the omnicopter are presented. The system dynamics may be presented in either of two ways. The *forward dynamics* arrangement of the equations is used to predict the system motion given some actuation. Conversely, the *inverse dynamics* seek to determine the actuation(s) necessary to achieve a certain system motion. It is generally necessary to solve for the inverse dynamics of the system when designing a model-based controller. However, since the controller presented in Chapter 5 is not model-based, only the forward dynamics are presented in this section.

Since the trajectories considered are not aggressive, the propellers are assumed to maintain nearly constant thrust allowing their dynamics to be ignored and the omnicopter to be treated as a single rigid body [46]. Newton's second law can then be used to write the

translational dynamics in the world frame as

$$m\ddot{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\boldsymbol{F_b} \tag{3.12}$$

where

- m: mass of the omnicopter
- $\ddot{\boldsymbol{x}}$: linear acceleration vector of the omnicopter 's center of mass expressed in the world frame
- g: acceleration constant due to gravity expressed in the world frame
- R: rotation matrix that transforms a vector expressed in the body frame to one expressed in the world frame
- $\boldsymbol{F}_b$: force produced by the omnicopter expressed in the body frame

Substituting in Equation 3.11 for the rotation matrix and including the relationship between actuator thrusts and net force from Equation A.4 and the top three rows of A.14, Equation 3.12 can be rewritten in terms of the omnicopter 's attitude quaternion $\boldsymbol{q}$ and propeller thrusts $\boldsymbol{f}$.

$$m\ddot{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2+2q_0q_2 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & 1-2q_1^2-2q_2^2 \end{bmatrix} \times$$
$$\begin{bmatrix} -a & b & a & -b & a & -b & -a & b \\ b & a & -b & -a & -b & -a & b & a \\ c & -c & c & -c & c & -c & c & -c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} \tag{3.13}$$

The rotational dynamics can be written in an analogous form and expressed in the body frame as

$$I_b\dot{\boldsymbol{\omega}}_b + \boldsymbol{\omega}_b \times I_b\boldsymbol{\omega}_b = \boldsymbol{\tau_b} \tag{3.14}$$

- $\boldsymbol{\tau}_b$: torque produced by the omnicopter expressed in the body frame
- I_b: inertia tensor for the omnicopter. It has been designed such that the matrix is diagonally dominant. I_b is expressed in the body frame so that it does not change with orientation
- $\boldsymbol{\omega}_b$: angular velocity vector expressed in the body frame

Substituting in Equation 3.11 for the rotation matrix and including the Jacobian relationship between actuator thrusts and net torque from Equation A.5 and bottom three rows of A.14

$$I\dot{\boldsymbol{\omega}}_{\boldsymbol{b}} + \boldsymbol{\omega}_b \times I_b\boldsymbol{\omega}_b = \begin{bmatrix} -ak - \frac{\sqrt{3}r(b-c)}{3} & bk - \frac{\sqrt{3}r(a+c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ bk - \frac{\sqrt{3}r(a+c)}{3} & ak + \frac{\sqrt{3}r(b-c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ ak + \frac{\sqrt{3}r(b-c)}{3} & -bk + \frac{\sqrt{3}r(a+c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ -bk + \frac{\sqrt{3}r(a+c)}{3} & -ak - \frac{\sqrt{3}r(b-c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ -ak - \frac{\sqrt{3}r(b-c)}{3} & bk - \frac{\sqrt{3}r(a+c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ bk - \frac{\sqrt{3}r(a+c)}{3} & ak + \frac{\sqrt{3}r(b-c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ ak + \frac{\sqrt{3}r(b-c)}{3} & -bk + \frac{\sqrt{3}r(a+c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ -bk + \frac{\sqrt{3}r(a+c)}{3} & -ak - \frac{\sqrt{3}r(b-c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \end{bmatrix}^T \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} \tag{3.15}$$

Chapter 4

Modelling of Propellers and Airflows

This chapter seeks to develop models that can predict the thrusts produced by the propellers while maintaining a simplicity that allows for their use in a real-time control system. In the first section, a propeller model is proposed and tested to relate PWM commands to single propeller thrusts. The model is then expanded in the following section to identify the effects of disturbing air streams coupling with that of the propeller's in the multi-propeller omnicopter. The final section describes the design of the experimental test bed to identify and verify these models.

It may be noted that all models developed in this chapter are static, ignoring actuator and airflow dynamics, to avoid the complexities introduced by attempting to model dynamic properties. Ultimately, the implementation of these models must be realizable on a real-time system requiring computational efficiency. For the purposes of control in this thesis, the trajectories are slow enough that static model is sufficiently accurate.

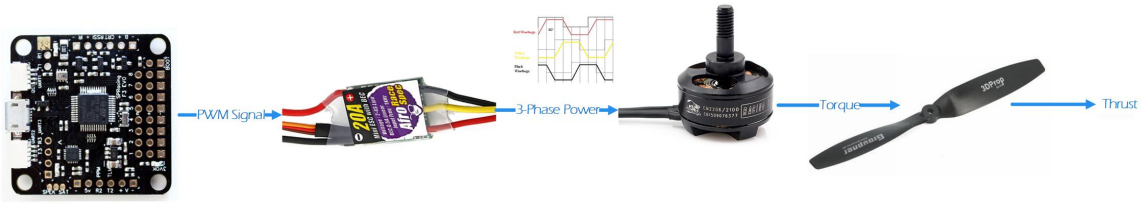

Figure 4.1: Block diagram of components making up an actuator

4.1 Propeller Model

Each actuator on the vehicle consists of an electronic speed controller (ESC), a brushless DC motor, and a bidirectional electric-flight propeller as shown in Figure 4.1. The flight controller depicted on the far left sends a pulse width modulation command to the ESC, which generates a discrete 3 phase power signal to produce torque in the brushless DC motor. This torque causes the propeller to accelerate and produce thrust. As the propeller continues to accelerate, the drag forces increase until they balance the torque output of the motor in steady state.

It is necessary to develop a mapping between the PWM command to the ESC and the final thrust produced. The work in [7] uses two sets of equations, namely, the equations of aerodynamic force/torque and those describing the DC motor to derive the relationship

$$V_{batt}|U_{pwm}| = \gamma_1|f_U| + \gamma_2\sqrt{|f_U|} + \gamma_3 \tag{4.1}$$

where V_{batt} is the battery voltage supplied to the motor, D_{pwm} is the scaled PWM signal sent to the ESCs. For the omnicopter's bi-directional actuators, the PWM signal can range between -500 (full negative throttle) to 500 (full positive throttle), where 0 is the neutral position where the actuator is stopped. The γ_{1-3} are 3 parameters to be identified through experimentation.

To identify γ_{1-3}, an experimental test bed is set up as described later in Section 4.3. The supplied voltage to the ESC is monitored while a program runs through a series of 10 PWM commands for both positive and negative thrust. For each test i, the motor ramps up to the desired PWM and then is held at the desired thrust f_i for six seconds, to allow for adequate settling time, while a force sensor collects data at a 1kHz sample rate. An average force is obtained over the 6 second period to reject a large portion of the sensor noise caused by motor vibration.

For n tests, the data could be organized in the standard $A\mathbf{x} = \mathbf{b}$ form

$$\begin{bmatrix} |f_1| & \sqrt{|f_1|} & 1 \\ \vdots & \vdots & \vdots \\ |f_n| & \sqrt{|f_n|} & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} V_{batt_1} * |U_{pwm_1}| \\ \vdots \\ V_{batt_n} * |U_{pwm_n}| \end{bmatrix} \tag{4.2}$$

The pseudo-inverse of A is taken to solve the least-squares solution for $\mathbf{x}$.

$$\boldsymbol{x} = A^{\dagger}\boldsymbol{b} \tag{4.3}$$

From the experimental data of all eight propellers, the average γ_{1-3} values are found to be

$$\gamma_1 = 417.57; \quad \gamma_2 = 1281.51; \quad \gamma_3 = -144.14; \tag{4.4}$$

In total 160 data points were collected to calculate the γ parameters.

Figure 4.2 overlays the experimental data of one experiment with the fitted model for a single bi-directional propeller operating in both directions. Each propeller was tested using 10 evenly spaced commands for both the positive and negative directions resulting

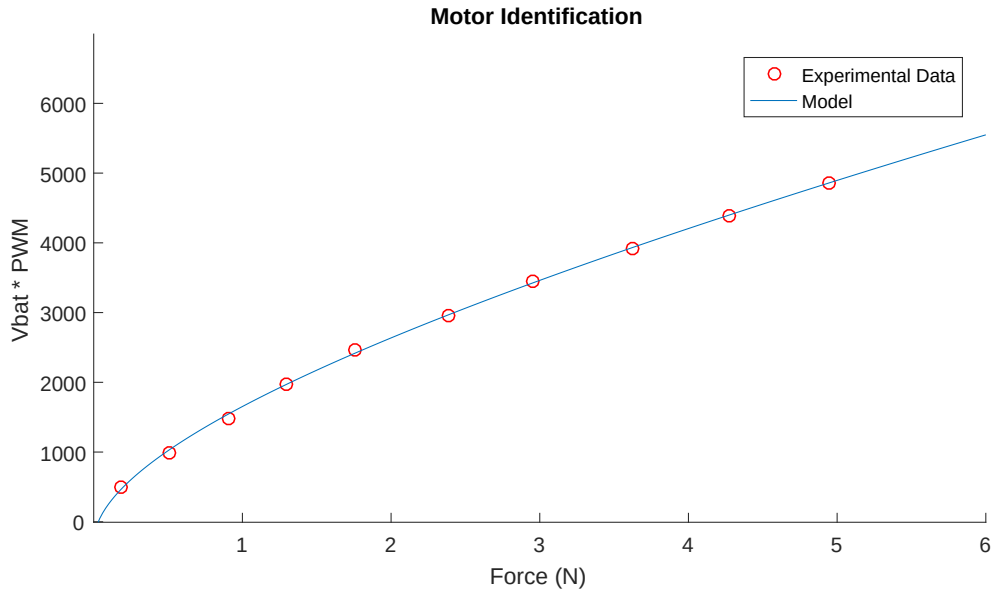

Figure 4.2: Comparison of experimental values vs. identified model of the the propeller

in the collection of 160 data points. Since the propellers are symmetric due to their bi-directionality, it is assumed that the γ parameters should be consistent for both directions. This is supported experimentally as the direction of thrust did not explain any significant variance in the γ_{1-3} values between experiments. Note that in producing this model, the sign of the PWM signal and the direction of thrust are ignored.

While collecting data to identify the propeller model, the parameter k from Equation 2.3 could also be identified. In a similar fashion to finding γ_{1-3}, the k was identified by collecting the propeller thrust along with the torque produced along the same direction of the thrust as a result of propeller drag torque. Data from multiple experiments using multiple propellers was regressed to find that

$$k = 0.01435 \tag{4.5}$$

4.2 Airflow Model

Unlike conventional multi-rotors, which have all propellers facing in the same direction, the propeller configuration chosen for the omnicopter does not constrain all the propellers to lie in the same plane. Although this provides the benefit of full actuation, it also brings with it the potential for propeller airflows to interact with each other. The model in Section 4.1 assumes that the propeller is operating in nominally *still* air. The propeller generates thrust by producing a pressure differential across its blades. Modelling such pressure differentials and airflows is extremely challenging but proves unnecessary since the simple model presented in Equation 4.1 can accurately predict steady state thrust generation.

Unfortunately, the situation becomes more complex with the introduction of disturbance airflows. Now the pressure differential is not always the same for a given PWM command and battery voltage as it depends on the thrust generation of other actuators. To explore these disturbance effects, a fluid dynamics simulation could be used to model the system. However, this path could become very involved and the desired model must be simple since it is to be used in a real-time control loop. For this reason, a simple model is proposed and justified. A system identification is then carried out to identify the model parameters and evaluate its performance.

4.2.1 Propeller Model Considering Interacting Airflows

Given that an actuator will be disturbed by the air flows of other actuators, the model for the required PWM signal from Equation 4.1 should not only be a function of V_{batt} and f_U, but should also include the thrusts from the other actuators. In still air, the identified γ_{1-3} are constant. The disturbing air streams can add complexity to the model such that the γ values are no longer constant but rather functions of the force produced by the disturbing

air streams.

For a given actuator U, there exists seven other actuators $D_i, i = 1, \cdots, 7$ that produce disturbance air flows to U. Equation 4.1 can now be expanded such that γ_{1-3} are functions of the disturbing actuators.

$$V_{batt}|U_{pwm}| = \gamma_1(f_{D1}, \ldots, f_{D7})|f_U| + \gamma_2(f_{D1}, \ldots, f_{D7})\sqrt{|f_U|} + \gamma_3(f_{D1}, \ldots, f_{D7}) \tag{4.6}$$

The proposed model may be unnecessarily complex due to the large number of function parameters for the calculation of each γ. By taking a closer look at the propeller configuration, it can be seen that for a given actuator U, there are only three other actuators $D_i, i = 1, \cdots, 3$ whose airflows directly intersect with that of U. An example of three such disturbing actuators are show in Figure 4.2. The three disturbing airflows can be differentiated from each other by analyzing angle and distance between the disturbing airflow vector and the U vector.

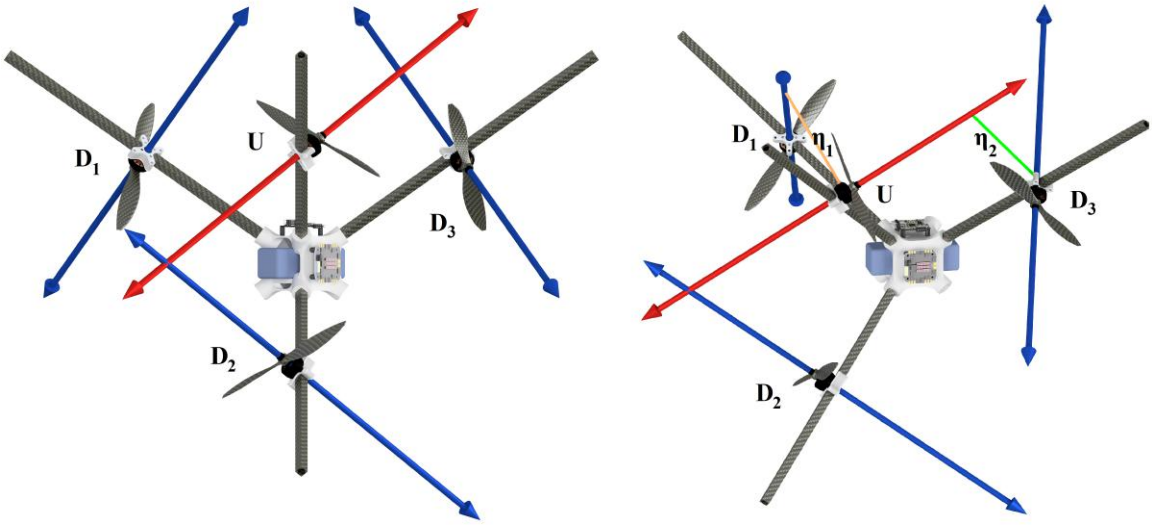

Figure 4.3: Diagram of the 3 actuator airflows (in blue) that disturb a given actuator (in red)

For each actuator, Table 4.1 shows the three disturbing motors and categorizes them as a type 1, 2, or 3 disturbance. The three types of disturbances are differentiated by examining how the airflows intersect. Drawing a vector through the middle of each airflow reveals that vector $\boldsymbol{D_2}$ directly intersects vector $\boldsymbol{U}$. Vectors $\boldsymbol{D_1}$ and $\boldsymbol{D_3}$ are skewed with respect to $\boldsymbol{U}$, and their angle with $\boldsymbol{U}$ is equal. To differentiate between the $\boldsymbol{D_1}/\boldsymbol{U}$ and $\boldsymbol{D_3}/\boldsymbol{U}$ skew pairs, a third vector $\boldsymbol{\eta}_{1/3}$ is defined for each pair. As shown in Figure 4.3, each $\boldsymbol{\eta}$ vector is perpendicular to both the actuator and respective disturbing motor vectors to mark the minimum distance between each skewed pair. Since this minimum distance is the same for both pairs, a distinction is made by observing how far the intersection is between the $\boldsymbol{\eta}_{1/3}$ vectors and the actuator vector $\boldsymbol{U}$. The $\boldsymbol{\eta}_1$ vector intersects 1.43 cm above the actuator meaning this disturbance airflow directly intersects the propeller, while the $\boldsymbol{\eta}_3$ intersection is much further above $\boldsymbol{U}$ at 19.9 cm.

U	D_1	D_2	D_3
1	4	5	2
2	1	6	3
3	2	7	4
4	3	8	1
5	8	1	6
6	5	2	7
7	6	3	8
8	7	4	5

Table 4.1: List of disturbance airflows for each actuator

Equation 4.6 can now be reduced to

$$V_{batt}|U_{pwm}| = \gamma_1(f_{D1}, f_{D2}, f_{D3})|f_U| + \gamma_2(f_{D1}, f_{D2}, f_{D3})\sqrt{|f_U|} + \gamma_3(f_{D1}, f_{D2}, f_{D3}) \tag{4.7}$$

where each γ_i parameter is now a function of the thrust produced by only three disturbing actuators. This simplification offers a model that is much more viable for use in real-time control.

A model should now be found to represent the hyper-surface for each of these γ_i functions. Since little is known about the shape for each γ, the model should be developed so that it has the flexibility to take on any arbitrary shape. To obtain an intuitive understanding of the shape of these surfaces, they will be is explored as a function of only two disturbance thrusts instead of three and the term that is not dependent on f_U is dropped to prevent over-fitting.

$$V_{batt}|U_{pwm}| = \gamma_1(f_{D1}, f_{D2})|f_U| + \gamma_2(f_{D1}, f_{D2})\sqrt{|f_U|} \tag{4.8}$$

In this way, it can be determined if a linear approximation would be suitable for describing the surfaces. Ideally, this could further imply that the true hyper-surface (a function of three disturbance actuator forces) can also be represented by a linear surface, which would greatly simplify the model.

This investigation is carried out by dividing each unknown surface into a series of gridded points of size $n \times n$. The grid lies on the x/y plane of Figure 4.4 where the two axes of this plane represent the disturbance forces produced by actuators D_1 and D_2. The grid limits are set large enough to span the full range of potential disturbance forces and given a suitable resolution that will reveal surface shape details without requiring superfluous amounts of data. Such a grid is shown in black in Figure 4.4. The z heights of each point in the grid will be calculated via a regression to produce the colourful γ surface shown above the black grid. Since there are two unknown surfaces, γ_1 and γ_2, the regression involves $2n^2$ unknowns.

An example experimental setup to produce Figure 4.4 involves choosing one of the actuators on the omnicopter as U. Then from Table 4.1, the disturbing actuators D_1 and D_2 can be found geometrically as explained in Section 4.2.1. PWM commands are then sent to these three motors and thrusts produced by each actuator are collected. This process is repeated to collect data for D_1 and D_2 disturbances forces that evenly span the pre-defined grid. It is necessary to collect multiple data points within each section of grid for the regression to work properly. Therefore, the actuator U is tested at three different command values, while actuators D_1 and D_2 span the entire grid for each of these set values. Further details of the experimental methods and setup is covered in Section 4.3.

The regression is formulated as a typical $A\mathbf{x} = \mathbf{b}$, where A should be a sufficiently

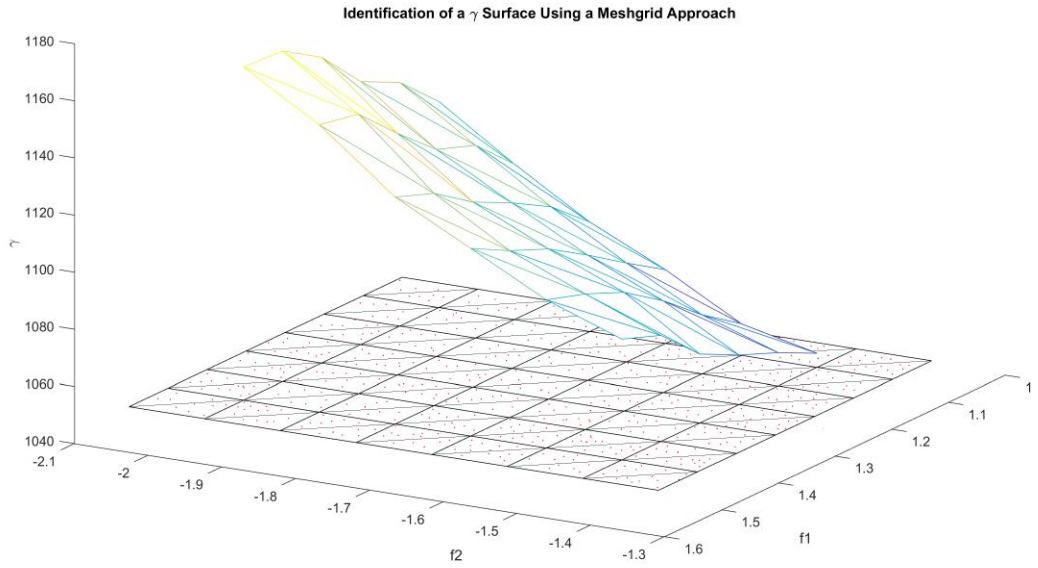

Figure 4.4: A grid is used to find the approximate shape of a γ surface

tall matrix to accomplish a meaningful regression. Each row of A represents a single experiment for which a PWM command is issued to each of U, D_1, D_2 and the resultant actuator forces are collected along with the supplied voltage to the actuators. Substitution of these measured values back into Equation 4.8 will produce an equation in terms of the two unknown γ surfaces. However, these γ surfaces are not a single variable but rather a collection of n^2 unknowns. Vector algebra is employed to rewrite Equation 4.8 in terms of the unknown grid point heights.

The formulation begins by noting that the square grid can be further divided into a series of triangles by connecting the top left and bottom right vertices of each square as shown in Figure 4.5. The f_{D1} and f_{D2} forces produced in a given experiment will place the data in one of the triangles of the grid. A given triangle can be found in one of two possible configuration as shown in Figure 4.4. From the figure, x_n and y_n represent the thrust forces of the disturbance actuators D_1 and D_2, while a_n should lie on the γ surface facet to be

determined.

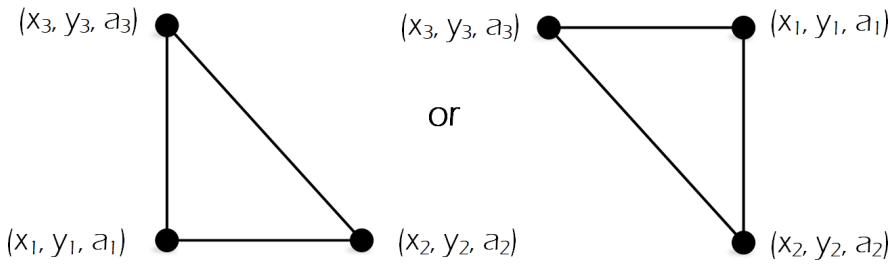

Figure 4.5: Triangular grids used to build up mesh

Each triangular surface facet can be described by the equation of a plane

$$Ax + By + Cz + D = 0 \tag{4.9}$$

The A, B, and C terms describe a vector perpendicular to this plane and can therefore be found by taking the cross product of two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ that lie on the plane.

Taking the cross product of $\mathbf{v}_1$ and $\mathbf{v}_2$ gives the vector

$$\mathbf{p}_\perp = \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} (a_1 - a_3)(y_1 - y_2) - (a_1 - a_2)(y_1 - y_3) \\ (a_1 - a_2)(x_1 - x_3) - (a_1 - a_3)(x_1 - x_2) \\ (x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2) \end{bmatrix} \tag{4.10}$$

D can be rewritten by rearranging Equation 4.9 and substituting one of the points from the grid as

$$D = -(Ax_1 + By_1 + Ca_1) \tag{4.11}$$

Equation 4.10 can be substituted back into Equation 4.9 along with a measured point

$[f_{D1}, f_{D2}, a_0]$ and rearranged to obtain

$$\begin{aligned}0 =& f_{D2}((a_1 - a_2)(x_1 - x_3) - (a_1 - a_3)(x_1 - x_2)) - f_{D1}((a_1 - a_2)(y_1 - y_3) - (a_1 - a_3)(y_1 - y_2)) \\ & - y_1((a_1 - a_2)(x_1 - x_3) - (a_1 - a_3)(x_1 - x_2)) + x_1((a_1 - a_2)(y_1 - y_3) - (a_1 - a_3)(y_1 - y_2)) \\ & + a_0((x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2)) - a_1((x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2))\end{aligned} \tag{4.12}$$

The measured point is the left-hand side of Equation 4.8. This can be expanded, refactored, and rearranged in terms of a_0, a_1, a_2, and a_3 to produce a result of the form

$$a_0 = \frac{c_1 a_1 + c_2 a_2 + c_3 a_3}{c_0} \tag{4.13}$$

where c_{0-3} represent the coefficients factored from a_{0-3}.

$$\begin{aligned}c_0 =& (x_1 - x_2) * (y_1 - y_3) - (x_1 - x_3) * (y_1 - y_2) \\ c_1 =& f_{D2}(x_2 - x_3) - f_{D1}(y_2 - y_3) - y_1 * (x_2 - x_3) + x_1(y_2 - y_3) \\ & - (x_1 - x_2)(y_1 - y_3) + (x_1 - x_3)(y_1 - y_2) \\ c_2 =& f_{D1}(y_1 - y_3) - f_{D2}(x_1 - x_3) + y_1 * (x_1 - x_3) - x_1(y_1 - y_3) \\ c_3 =& f_{D2}(x_1 - x_2) - f_{D1}(y_1 - y_2) - y_1 * (x_1 - x_2) + x_1(y_1 - y_2)\end{aligned} \tag{4.14}$$

This formulation must be expanded to reflect the proposed model represented by Equation 4.8. The equation details three surfaces where the first surface is scaled by the actuator U force, the second by the square root of the actuator U force and the third by a unity gain. The summation of a given point $(f_{D1}, f_{D2}, \gamma_{1-3})$ on each of these surfaces by their respective gains equals the measured a_0.

Each row of the A matrix represents a single experiment. Any given row will contain nine non-zero terms, three terms for each γ surface corresponding to the unknown grid elements in which the experimental D_1 and D_2 forces fall. An example row is given for the data point found in Figure 4.6. In this example, a 10x10 grid is constructed with data points labeled from top left to bottom right (first row 1-10, second from 11-20, etc). The disturbing forces produced in the experiment indicate that the red data point falls in the triangular facet surrounded by x_1, x_2, and x_{12}. Note that grid points are labeled traveling right and downwards. Equation 4.15 shows a single row in the A matrix.

$$\begin{bmatrix} f_U\frac{c_1}{c_0} & f_U\frac{c_2}{c_0} & 0 & \cdots & 0 & f_U\frac{c_3}{c_0} & 0 & \cdots & 0 & \sqrt{f_U}\frac{c_1}{c_0} & \sqrt{f_U}\frac{c_2}{c_0} & 0 & \cdots & 0 & \sqrt{f_U}\frac{c_3}{c_0} \end{bmatrix} \begin{bmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_{12}^1 \\ \vdots \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_{12}^2 \end{bmatrix} = V_{batt}U_{PWM} \tag{4.15}$$

The experiment is repeated to span all force combinations of D_1, D_2, and U to produce a tall A matrix upon which a regression is performed to solve for the grid points. The height of the A matrix is equal to the number of experiments conducted. Such a result is of the standard $Ax = b$ where x may be solved by taking the pseudo-inverse of A such that $x = A^{\dagger}b$.

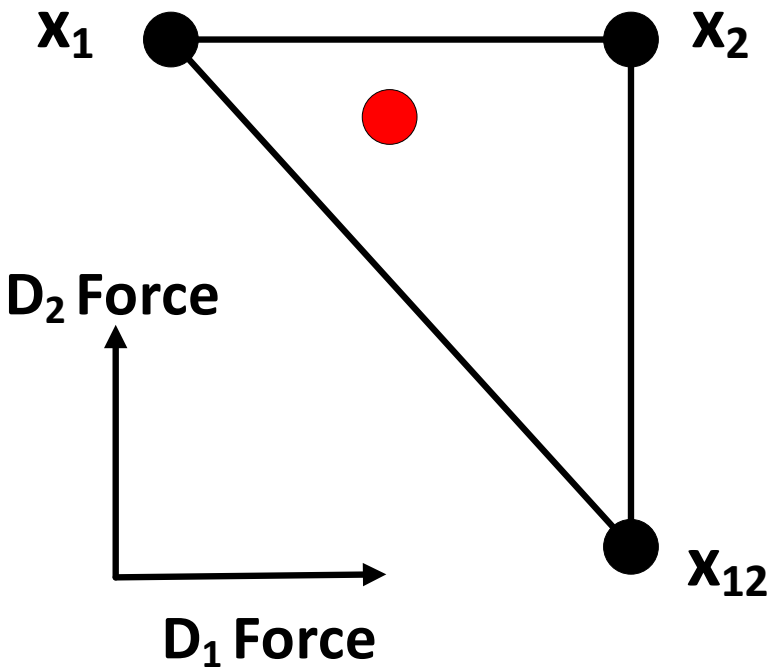

Figure 4.6: An example data point lying within one of the triangular facets of the grid

Each experiment carried out produces a relationship defined by Figure 4.6 corresponding to one row of coefficients in the A matrix. Enough tests are done to make A a sufficiently tall matrix. In this case, A is about four times taller than wide. A grid size of 8x8 is chosen for these experiments. This means there are 128 unknowns requiring a minimum 512 experimental data points.

When fitting data to the model, care must be taken not to over fit. It is found that it is better to not give degrees of freedom to $\gamma_3(f_{D1}, f_{D2}, f_{D3})$, which is why it is excluded in Equation 4.8. The A matrix must also be full column rank to ensure there is a unique solution. The regression leads to the generation of two γ surfaces as shown in Figure 4.7. This particular figure demonstrates the relationship between each of the γ's as functions of

the disturbance forces f_{D1} and f_{D2}. It may be noted that the relationship appears roughly linear.

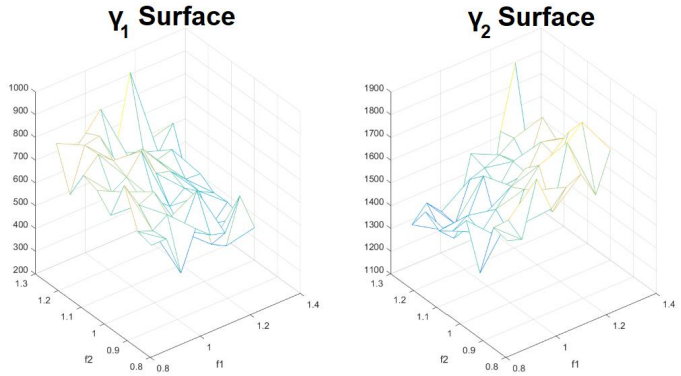

Figure 4.7: γ_1 and γ_2 surfaces

Using the mesh approach allows initial insight to reasonably approximate the relationship between disturbance thrusts and the γ parameters as linear. However, it can be seen visibly that the mesh approach may still suffer from some overfitting as seen by the noise or jagged edges in each plane. Fitting only a single gamma curve to the data as in Figure 4.4 provides a smoother surface suggesting less overfitting. Another disadvantage is revealed during the implementation of such a model: i.e., this grid could require a rather large look-up table that proves unwieldy on microcontrollers with limited resources. The assumption that the relationship can be approximated as linear will be used in the development of a simpler, more efficient model.

Revisiting Equation 4.7, the function for γ will now be modeled as a linear relationship between f_{1-3}

$$\gamma_i = a_i f_1 + b_i f_2 + c_i f_3 + d_i \tag{4.16}$$

The formulation of the regression problem now becomes simpler since there are only nine variables to solve for: four parameters for the γ_1 hyperplane, four parameters for the γ_2 hyperplane and a constant for γ_3, e_0. Equation 4.7 can now be written as

$$V_{batt}|U_{pwm}| = (a_1 f_{D1} + b_1 f_{D2} + c_1 f_{D3} + d_1)|f_U| + (a_2 f_{D1} + b_2 f_{D2} + c_2 f_{D3} + d_2)\sqrt{|f_U|} + e_0 \tag{4.17}$$

Since there are four bidirectional actuators in the tests (a primary actuator and three disturbing actuators), there are 16 possible combinations of positive and negative thrust. As a result, 16 sets of parameters must be calculated for each scenario. The ESCs have been set such that a command of zero provides no thrust and there is virtually no deadband for actuation. Therefore e_0 should effectively be zero. The calculated parameters are shown in Table 4.2. The direction of thrust for each of the actuators is encoded in the order [U D_1 D_2 D_3]. 'F' indicates a *forward*, or positive, thrust, while 'R' indicates a *reverse*, or negative, thrust. The four actuators iterate through all combinations of eight discrete PWM commands evenly distributed between positive and negative thrust. This leads to the collection of 4096 data points to calculate the 16 parameter sets.

4.2.2 Model Verification

The model must now be tested to see how accurately it predicts the effects of airflow interactions. Two verifications will be performed. The first method splits the data set into two halves. The first half is used to train the model, while the second half is used to verify how accurately the model can make predictions. The second method involves real experimentation where desired force\torque commands are fed to the omnicopter . The model is used to translate force-torque commands to PWM commands. A six degree of freedom

Actuator Directions	a_1	b_1	c_1	d_1	a_2	b_2	c_2	d_2
FFFF	61.3	-58.0	-102.8	622.1	-128.7	156.6	169.4	1121.2
FFFR	49.5	-80.3	-60.1	656.2	-127.8	166.6	161.3	1115.0
FFRF	69.2	-29.1	-95.2	602.5	-145.7	66.8	152.7	1163.9
FFRR	54.0	-13.3	-64.3	646.1	-142.5	62.8	176.3	1151.2
FRFF	109	-51.9	-99.7	617.2	-216.3	162.3	176.6	1099.7
FRFR	126	-74.3	-62.2	651.3	-222.2	171.8	153.1	1095.1
FRRF	101	-31.9	-90.4	593.8	-200.4	58.1	159.6	1142.5
FRRR	122	-16.1	-67.9	637.5	-209.4	54.9	168.4	1131.9
RFFF	89.5	-22.0	-38.5	-724.4	-214.9	79.6	110.6	-960.3
RFFR	74.8	-31.3	-107.4	-700.0	-213.4	74.2	205.6	-950.6
RFRF	39.1	28.5	-95.1	-599.3	-133.8	-13.0	202.1	-1162.8
RFRR	19.3	48.7	-56.3	-562.0	-123.6	-23.8	123.2	-1174.8
RRFF	53.9	-20.7	-37.1	-727.4	-149.6	77.4	108.3	-955.2
RRFR	57.7	-26.3	-111.6	-711.5	-137.7	65.2	213.1	-929.6
RRRF	104	32.4	-99.5	-589.6	-232.8	-19.9	209.7	-1179.7
RRRR	114	50.5	-54.7	-557.6	-231.2	-26.7	120.6	-1181.7

Table 4.2: Table of parameter values satisfying 4.17 for each combination of actuator directions

(DOF) force sensor is then used to determine how closely the system achieves the desired force/torque.

Verification Using the Euclidian Norm

The first performance metric used to verify the accuracy of the model is written as

$$error = \sqrt{\frac{(Y_{true} - Y_{predicted})^2}{((Y_{true})^2)}} \tag{4.18}$$

where

$$\begin{aligned} Y_{true} &= V_{batt}|U_{pwm}| \\ Y_{predicted} &= (a_1 f_{D1} + b_1 f_{D2} + c_1 f_{D3} + d_1)|f_U| + (a_2 f_{D1} + b_2 f_{D2} + c_2 f_{D3} + d_2)\sqrt{|f_U|} \end{aligned} \tag{4.19}$$

To test the model's performance, the data is randomly split into two groups. The first group is used as in Section 4.2.1 to calculate the model parameters a_i, b_i, c_i and d_i. For each test in this data group, Equation 4.18 is used to estimate unmodelled variance in the data set. Note that the model should not explain all the variance in the data set as some of this variance comes from sources of noise. As mentioned previously, if the model has too many degrees of freedom, it can become susceptible to *overfitting*, where noise starts to be included in the model.

To ensure that the model can make accurate predictions on new data, the model created with data set 1 is verified on a second independent data set. The resulting % error between the measured values and model prediction is shown in the third column of Table 4.4 and is similar to Column two. This supports the claim that the model does not over fit to noise.

It may be noted that the model deviates from measured values more significantly when actuator U is producing thrust in the reverse direction. These numbers can be misleading since percentage deviation is not directly related to absolute deviation. It is hypothesized these larger percentage deviations are due to the fact that the U airflow interacts less with disturbing airflows meaning that the disturbance effects are less pronounced resulting in a smaller signal to noise ratio in measurement.

Actuator Directions	Unmodeled Variance Data set 1	Unmodeled Variance Data set 2
FFFF	0.52%	0.53%
FFFR	0.55%	0.59%
FFRF	0.90%	0.90%
FFRR	0.94%	0.93%
FRFF	0.67%	0.67%
FRFR	0.58%	0.62%
FRRF	0.93%	0.93%
FRRR	1.42%	1.42%
RFFF	2.36%	2.35%
RFFR	2.61%	2.60%
RFRF	2.76%	2.76%
RFRR	2.79%	2.81%
RRFF	2.49%	2.48%
RRFR	2.55%	2.56%
RRRF	2.75%	2.75%
RRRR	2.64%	2.64%

Table 4.3: % error between model prediction and measured value

Experimental Verification

The system identification test bed described in Section 4.3 is used to send some desired force/torque commands to the omnicopter . The Jacabion from Equation A.14 is used to compute the desired thrusts for each actuator. The γ model is then used to calculate the necessary PWM value to achieve these thrusts. The system ramps up to the eight PWM commands and then holds these values for five seconds while the 6 DOF force sensor measures the force/torque produced by the omnicopter . The desired and measured forces/torques are then compared to evaluate the performance of the model.

Figure 4.8 shows the net x, y, and z forces produced by the omnicopter over a series of tests. Each test aims to produce 6N of force first in the positive and then the negative x, y, and z directions. It can be seen from the figure that overall the Jacobian and model for the airflows performs well to produce these desired forces while keeping the other linear

forces close to zero.

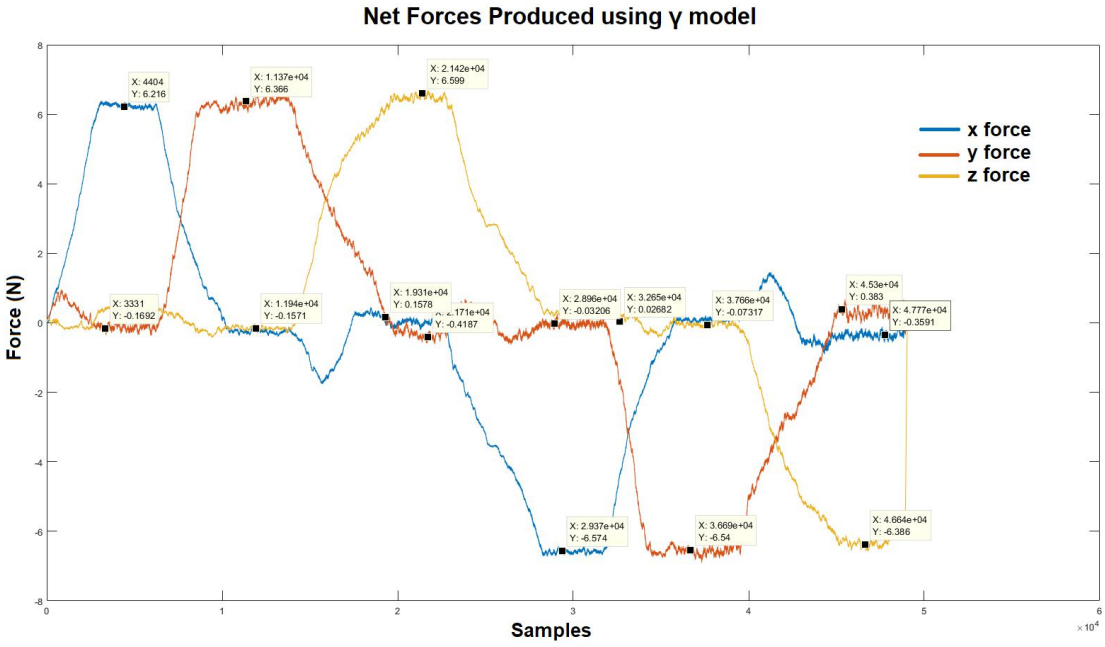

Figure 4.8: Experimental verification of the γ model for linear net forces

However, these tests produce actuator thrusts that are relatively symmetric, so it is valuable to investigate more arbitrary force/torque combinations. Table 4.4 shows the omnicopter 's performance for a variety of force/torque commands. Forces/torques with a d superscript are the desired actuations, while those with an m superscript are the forces and torques measured by the 6 DOF sensor. Some of these commands better expose some of the errors in the model as some forces and torque can be off by significant amounts. Investigation can be carried out in future work to upgrade the experimental test bed to remove sources of error (discussed in the next section) which may impact the accuracy of the model. Other more complex nonlinear models may be investigated. However, for the purposes of this thesis, the model performs adequately and as will be seen in later results, can be used in real-time to control the omnicopter.

Table 4.4: Unmodelled variance in the trained and prediction datasets

f_x^d	f_y^d	f_z^d	τ_x^d	τ_y^d	τ_z^d	f_x^m	f_y^m	f_z^m	τ_x^m	τ_y^m	τ_z^m
6	0	0	0	0	0	6.31	-0.29	0.38	0.03	-0.01	0.05
0	6	0	0	0	0	-0.20	6.39	-0.08	0.01	0.03	0.01
0	0	6	0	0	0	-0.13	-0.24	6.87	0.02	0.05	-0.01
1	0	6	0	0	0	0.90	-0.61	6.87	-0.00	0.03	0.00
1	1	6	0	0	0	1.11	0.66	6.92	0.02	0.04	-0.03
0	0	6	0	0.5	0	-0.44	-0.38	6.84	0.11	0.69	-0.01
0	0	5	0.5	0.3	0	-0.38	0.25	6.06	0.69	0.43	-0.05
1	3	1	-0.5	1	0.5	-0.22	2.31	1.14	-0.53	1.26	0.63
3	0	2	0	0	1	2.91	-1.44	2.15	-0.02	0.03	1.27

It may be noted that this chapter only investigates the steady state performance of the omnicopter as it is deemed sufficient for control of the omnicopter for conservative trajectories. Transient performance is investigated in Chapter 5 to model the omnicopter in simulation.

4.3 Experimental Test Bed

This section describes the experimental set up used to identify and verify the propeller and airflow models of Sections 4.1 and 4.2. Automation is an important consideration in the design of the test bed since many experiments must be run in order to obtain a meaningful regression. In the case of the airflow identification, one set of γ parameters (one row in Table 4.2) are determined by running the actuators in a given direction between 20% and 80% in 20% incremental steps. To run all permutations for four propellers requires 256 tests. This must be repeated for a total of the 16 directional combinations of the four propellers to reach a total of 4096 tests. Running this many tests manually would prove an immense task.

The first step is to move away from powering the test bed via LiPo batteries as these

require constant recharging and waste time. A 16 volt, 80 amp power supply is used to produce the required power for four propellers running continuously. The test bed can be broken down into two major systems: the control command and measurement systems. It is important that when a command is sent to the actuators, the response is measured with minimal latency. For this reason the test bed is run using a real-time Simulink application to ensure recorded commands are synchronized with measured data. Simulink relied on two expansion cards to make this possible.

The first is a Quanser Q8 board which has eight digital to analog converters (DACs) and multiple digital input/output (I/O) channels. The four PWM commands sent by Simulink are converted to an analog voltage by the Q8 board. These voltages are then read by an Arduino Due and encoded into the MultiWii Serial Protocol (MSP). This protocol allows for communication between the Arduinio and the flight controller. Finally, the flight controller produces the PWM commands necessary to communicate with the ESCs and drive the motors.

The measurement system starts with the ATI 6 DOF force/torque sensor mounted to the center of the omnicopter . This high performance sensor captures the net force/torque produce by the omnicopter by producing a set of analog signals which are amplified before being sampled at 1kHz by an NI DAQ card on the PC. These analog signals must be multiplied by a transformation matrix to obtain the net force/torque vector produced by the omnicopter . This matrix is uniquely identified in the factory at the time the force sensor is made and provided with the amplifier box. The voltage from the power supply is also monitored and fed back to Simulink.

For propeller identification in Section 4.1, the single motor-propeller assembly can be mounted directly on top of the force sensor. However, care must be taken to ensure that

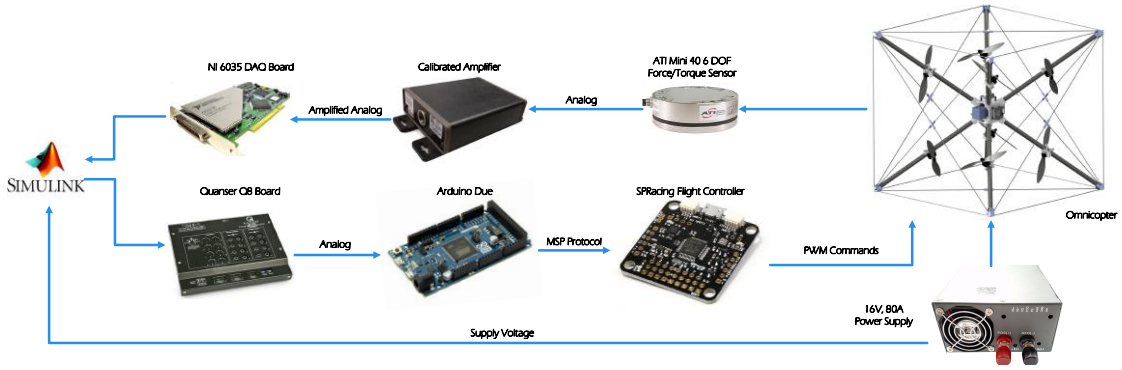

Figure 4.9: Block diagram of the experimental test bed for system identification and verification

heat from the motor is not transferred to the sensor as this will cause significant erroneous drift in measurement.

It should be noted that the model for actuator air flows requires knowledge of individual actuator thrusts. However, only one force sensor is employed to measure the net force-torque. Although the use of four sensors, each mounted to the base of a motor, would allow for direct force measurement, such a setup would be far more complex to implement as each sensor requires its own DAQ card. Fortunately, Equation 2.1 can modified as

$$\mathbf{f}_{4\times1} = (J')^{\dagger}_{4\times6}\boldsymbol{\zeta}_{6\times1} \tag{4.20}$$

where $(J')^{\dagger}$ is the pseudoinverse of the Jacobian. Here (J') is a subset of J that contains only the columns corresponding to active actuators. In this case, these are the three disturbing actuators $D_{1,2,3}$ and U. As long as six or fewer actuators are active, a unique, solution will exist for $\mathbf{f}$. Since this is an indirect way to measure the force produced by each actuator, it is susceptible to experimental error. Such errors include differences between the Jacobian derived in the Appendix and the real omnicopter . There may also be some error

when performing Equation 4.20 as the project of the force/torque vector on anything less than six actuator axes will have some residual error.

Figure 4.10 shows the omnicopter mounted on the force sensor (hidden, but in the area of the green plastic), which is secured between two tables.

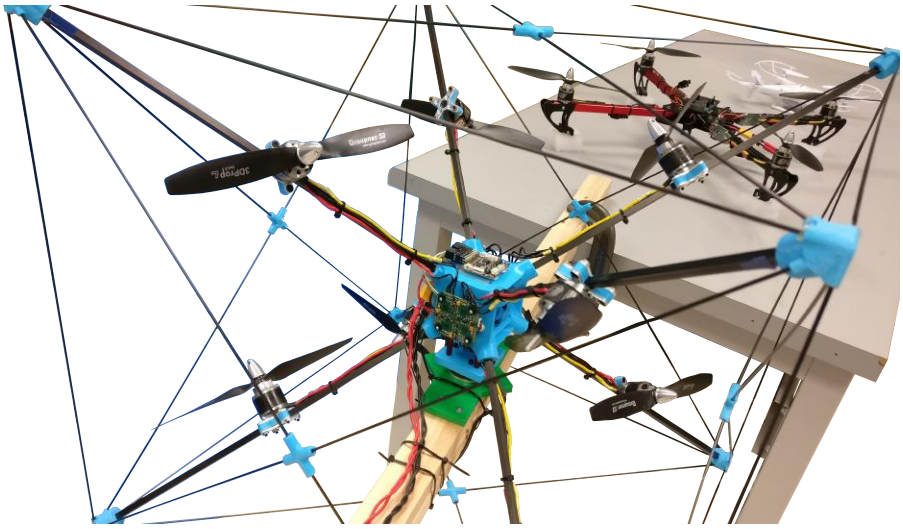

Figure 4.10: Experimental test bed for system identification and verification

Chapter 5

Control Strategy

In this chapter, a control strategy is presented for the omnicopter system. A standard PID controller with gravity compensation is selected for position control of the drone, while cascaded P and PID loops are used for attitude control. The control strategy is verified in the Matlab/Simulink simulation environment using a realistic model of the omnicopter.

5.1 Control Design

The position and attitude control problems of the fully-actuated omnicopter can be decoupled due to the full-actuation of the omnicopter . Note that the dynamics of the omnicopter are not fully decoupled when propeller dynamics are considered. However, for the trajectories performed in this thesis, these dynamics may be reasonably ignored. From

Equation 3.12, the translational dynamics can be written as

$$m\ddot{\boldsymbol{x}} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \boldsymbol{u_f} \tag{5.1}$$

where $\boldsymbol{u_f}$ is the control input force and $\ddot{\boldsymbol{x}}$ is the acceleration of the centre of mass of the omnicopter in the world frame. The control force is chosen as

$$\boldsymbol{u_f} = K_p^f(\boldsymbol{x_d} - \boldsymbol{x}) + K_d^f(\dot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{\boldsymbol{x}}) + K_i^f \int_0^t (\boldsymbol{x_d}(\zeta) - \boldsymbol{x}(\zeta))d\zeta + \boldsymbol{f_g} \tag{5.2}$$

where $\boldsymbol{x}_d$ is the desired position and K_p^f, K_d^f, and K_i^f represent the respective proportional, derivative, and integral gains for the position controller. The K gains are written as diagonal matrices to ensure that each axis can have an independent gain.

$$K = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} \tag{5.3}$$

The $\boldsymbol{f_g}$ vector is the gravity feed-forward term to cancel the gravity force and is given by

$$\boldsymbol{f_g} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{5.4}$$

Equation 5.1 and 5.2 are combined to write the closed-loop dynamics of the system as

$$m\ddot{\boldsymbol{x}} = K_p^f(\boldsymbol{x_d} - \boldsymbol{x}) + K_d^f(\dot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{\boldsymbol{x}}) + K_i^f \int_0^t (\boldsymbol{x_d}(\zeta) - \boldsymbol{x}(\zeta))d\zeta \tag{5.5}$$

A derivation of the attitudinal closed-loop dynamics starts in a similar way to the linear dynamics above. From Equation 3.14, the rotational dynamics are written as

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \boldsymbol{u_\tau} \tag{5.6}$$

where $\boldsymbol{u_\tau}$ is the control input torque and all elements are expressed in the body frame. Similarly to Equation 5.2, the control law for the angular velocity loop can be written as

$$\boldsymbol{u_\tau} = K_p^\tau(\boldsymbol{\omega_d} - \boldsymbol{\omega}) + K_d^\tau(\dot{\boldsymbol{\omega}}_{\boldsymbol{d}} - \dot{\boldsymbol{\omega}}) + K_i^\tau \int_0^t (\boldsymbol{\omega_d}(\zeta) - \boldsymbol{\omega}(\zeta))d\zeta \tag{5.7}$$

Note that the derivative term here would generate an algebraic loop with potential stability issues in discrete-time implementation since the dynamics of Equation 5.6 arefirst order in $\boldsymbol{\omega}$. However, the D term must be included here to address other dynamics (such as propeller dynamics) and delay that are not captured in Equation 5.6. It is common practice to include this term in the angular velocity controller as it improves performance by reducing oscillation.

The outer control loop produces the reference angular velocity for the inner-loop controller proportional to the angular position error. To this end, the measured and desired attitude quaternions, $\boldsymbol{q}$ and $\boldsymbol{q_d}$ respectively, can be related through an error quaternion $\boldsymbol{q_e}$

as

$$\boldsymbol{q} \otimes \boldsymbol{q_e} = \boldsymbol{q_d} \tag{5.8}$$

The inverse of a quaternion is given by

$$\boldsymbol{q^{-1}} = \frac{q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k}}{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{5.9}$$

where the denominator of Equation 5.9 is one for a unit quaternion. It is then easy to compute the error quaternion as

$$\boldsymbol{q_e} = \boldsymbol{q^{-1}} \otimes \boldsymbol{q_d} \tag{5.10}$$

Note that multiplication of quaternions via the Hamilton product, like other consecutive rotations, is not commutative.

Now $\boldsymbol{\omega_d}$ can be rewritten as a function of the error quaternion, which is expressed in axis-angle form. This form intuitively describes the axis around and angle by which the omnicopter should be rotated to achieve the desired attitude. This axis-angle is expressed in the body frame.

$$\boldsymbol{\omega_d} = K_p^{\omega} * \frac{2 * \arccos q_0^e}{\sqrt{1 - (q_0^e)^2}} \begin{bmatrix} q_1^e \\ q_2^e \\ q_3^e \end{bmatrix} \tag{5.11}$$

Care must be taken in implementation to deal with a singularity that occurs when the axis-angle representation presents an angle of zero. In this situation the denominator in Equation

5.11 becomes 0 and the solution infinite. For this case $\boldsymbol{\omega}_d$ is set to zero since there is zero error between the measured and desired attitudes.

Figure 5.1 shows the controller architectures for both the position and attitude controllers. The attitude controller employs a cascade-style architecture with two feedback loops each operating at a different rates. The slower outer loop operates on the error quaternion, $\boldsymbol{q}_e$ to generate a desired angular velocity as per Equation 5.11. This error quaternion is derived using Equation 5.10 by inverting the measured quaternion of the omnicopter and multipying it with the desired attitude quaternion. The inner loop follows the desired $\boldsymbol{\omega}_d$ and operates at a much faster rate using feedback from the onboard gyroscopes.

5.2 Considerations in Control Implementation

As discussed previously, the choice of control loop architecture is largely driven by the characteristics of the available sensor feedbacks. Since the position control loop only derives position feedback from a camera-based motion capture system, only a single loop is necessary. However, the attitude controller uses a cascade architecture since it has two feedbacks working at different rates. The outer attitude control loop uses the slower (120Hz) feedbacks from the motion capture system and accelerometer to correct attitude, while the inner angular velocity control loop takes advantage of the high bandwidth gyro feedbacks (1-8kHz).

Unfortunately, some of the sensor feedbacks suffer from significant noise. The issue is exacerbated when numerical differentiation needs to be performed (e.g., for deriving velocity from the motion-capture position measurements). A low-pass finite impulse response (FIR) filter is used to reduce high frequency noise. The trade-offs for an FIR design involved balancing filter order and delay. A high-order FIR is desirable for its ability to

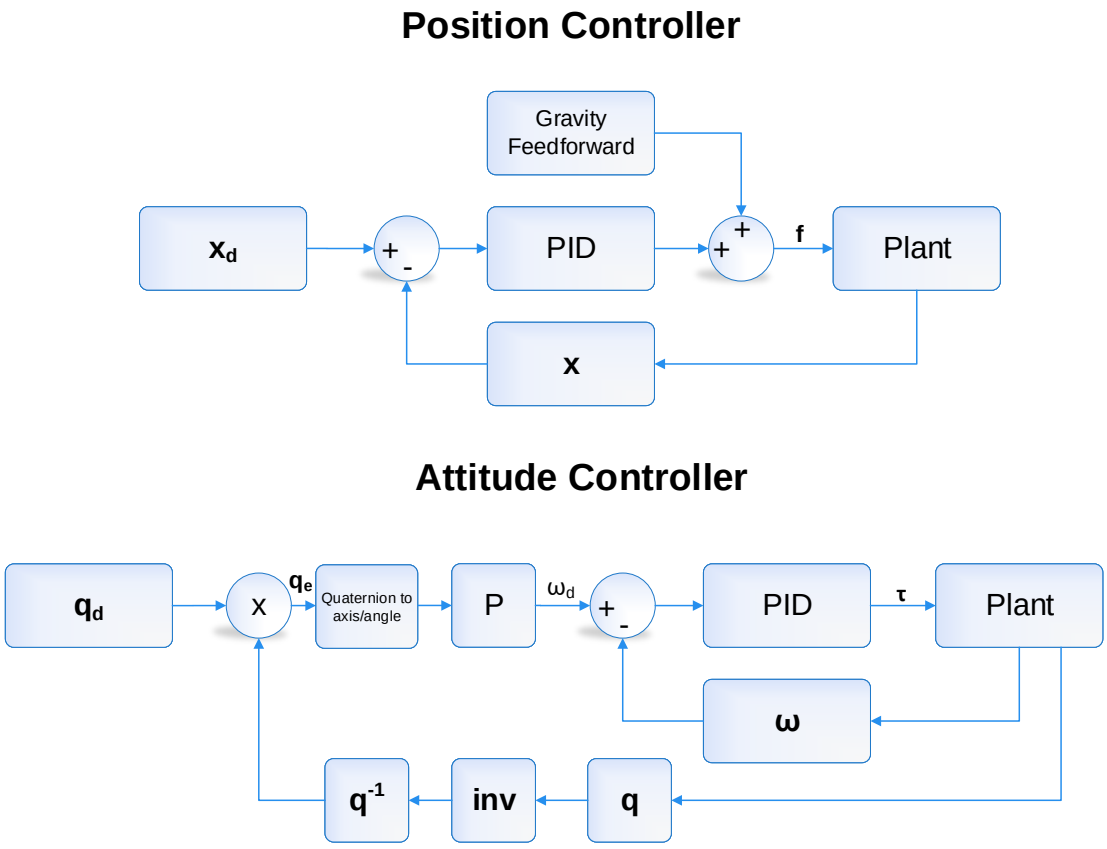

Figure 5.1: Architectural Block Diagrams for the Position and Attitude Controllers

reject high frequency noise; however, the unwanted side-effect of incurred delay should be minimized.

The amount of delay present in the system can limit the gains and subsequent overall performance of the controllers. This delay is minimized by using low-latency, real-time communication protocols and through the careful selection of filter orders. The group delay of a linear phase FIR filter is $(N-1)/2$ samples where N is the filter order [47]. The attitude controller is much more sensitive to system delay than the position controller requiring extra care in filter design. Delays in feedback are modeled in the following section.

5.3 Simulation of System Response

The proposed control architecture is verified in Simulink using the SimMechanics™ toolbox as seen in Figure 5.2. The trajectory planner generates a desired position and attitude trajectory. The plant block contains the SimMechanics model of the omnicopter and is responsible for simulating the system dynamics. It provides sensor feedback in the form of measured attitude, angular velocity, and position. The feedback is fed through a transport delay to mimic latencies produced by communication limitations and filtering. Velocity is derived numerically as in the true implementation. Finally, the desired force/torque is mapped to actuator thrusts via the ForceTorque_to_MotorCMDs block. The magnitudes of the actuator thrusts are bounded using a saturation block.

The motor dynamics must be considered to accurately model the omnicopter . In modeling the motor dynamics, the fast electrical transients are ignored to simplify the model and only slower mechanical dynamics are considered. Because these electrical transients are so much faster than the mechanical response, it is reasonably assumed that the torque

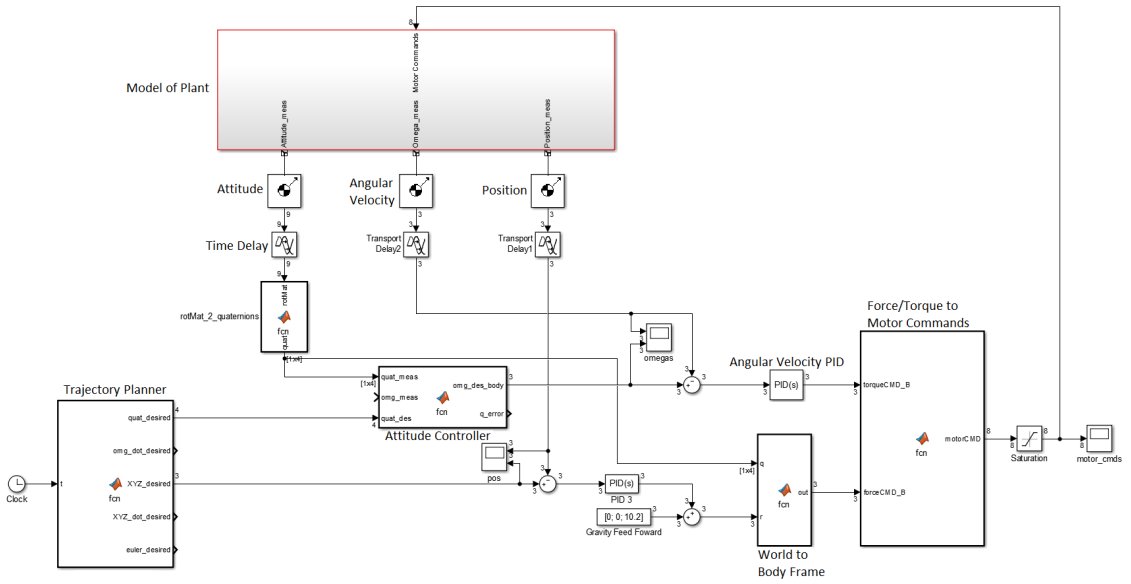

Figure 5.2: Simulation of controller using SimMechanics in the Simulink environment

generated by the motor is instantaneous such that the motion dynamics can be written as

$$J\dot{\omega} = \tau_M - \tau_L \tag{5.12}$$

where τ_M is the torque produced by the motor, τ_L is the drag torque from the propeller, and J is the inertia of the motor/propeller assembly [48]. The static thrust produced by a propeller f_p is related to it angular velocity by

$$f_p = \alpha\omega^2 \tag{5.13}$$

where α is a proportionality constant. Equations 2.3 and 5.13 can then be used to re-write Equation 5.12 as

$$J\dot{\omega} = \tau_M - k\alpha\omega^2 \tag{5.14}$$

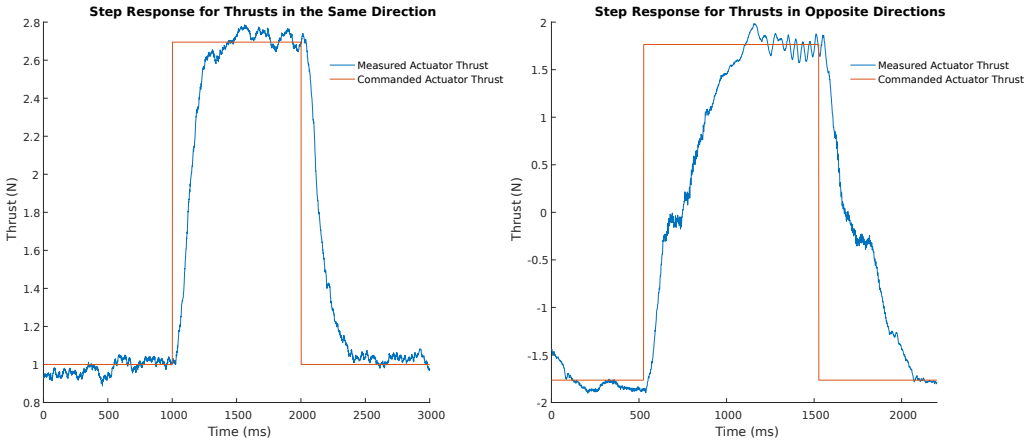

Figure 5.3: Test bed step responses of the actuators

Since the motor electrical dynamics are ignored, it can be assumed that the motor torque τ_m is instantaneously related to the motor command. The system identification test bed of Section 4.3 is used to roughly identify the step response of the motor. Since the purpose of the simulation is to perform a preliminary verification of the control design, it is not necessary to dedicate significant time to the identification of these motor parameters. Instead, the test bed is used to observe the step response of the motors between several different transitions and conservatively tune the Simulink model to match these results. Over the course of the tests, it was noted that the 90% rise time was always under 250ms when transitioning between thrusts of the same direction, while the transition between thrusts in opposite directions had a 90% rise time under 500ms. Two different step responses are shown in Figure 5.3.

Equation 5.14 is implemented in SimMechanics by providing an instanteous torque to the motor while simulating the resulting drag torque using an angular velocity feedback of the propeller. J is internally calculated by SimMechanics and used by the solver, k is set from the system of Section 4.1, and α is tuned from the transient response observations of

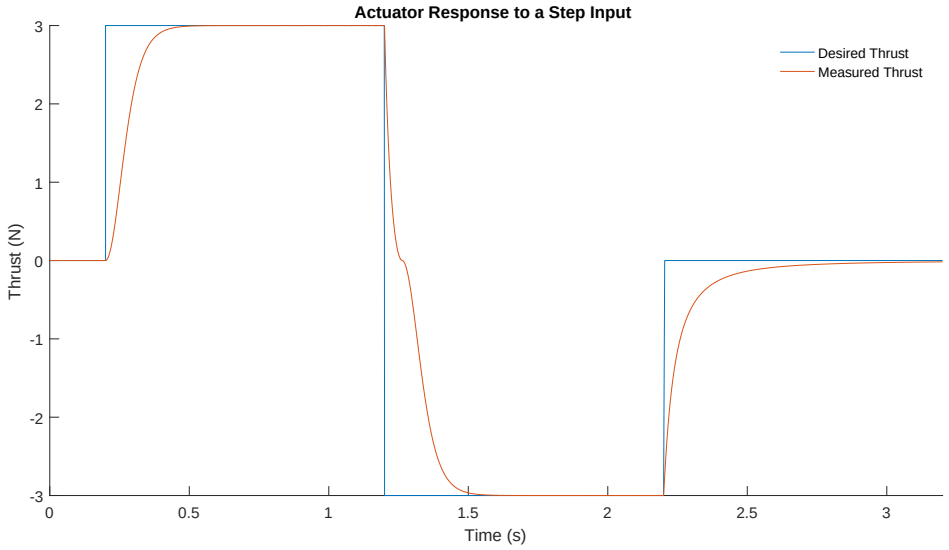

Figure 5.4: Simulated step responses of the actuators

Figure 5.3 to obtain an approximate model for the actuator dynamics. The response of the model to a step is presented in Figure 5.4. Although it does not include transients resulting from a transition between positive and negative thrusts, it reasonably approximates the test bed results for the purposes of this simulation.

The controller was successfully verified in Simulink. The results are shown in Figures 5.5 and 5.6. In the first trajectory, the omnicopter rises to a desired altitude of 1m while also performing a simultaneous 45 degree rotation in roll, pitch, and yaw. The small dip in the measured z position of Figure 5.5 is due to the fact that the gravity feedforward term is slightly less than the true force of gravity (just as it will be in real implementation). Since the omnicopter is not sitting on any surface at the beginning of the simulation, it starts by falling until the PID can properly compensate for error in altitude.

In Figure 5.6, the omnicopter maintains a constant altitude while performing a 1m and 2m translation in x and y respectively as well as performing an 85, 180, and 45 degree

rotation in roll, pitch, and yaw respectively. This test demonstrates the controller's ability to effectively follow a trajectory that is clearly in its non-linear mode of operation.

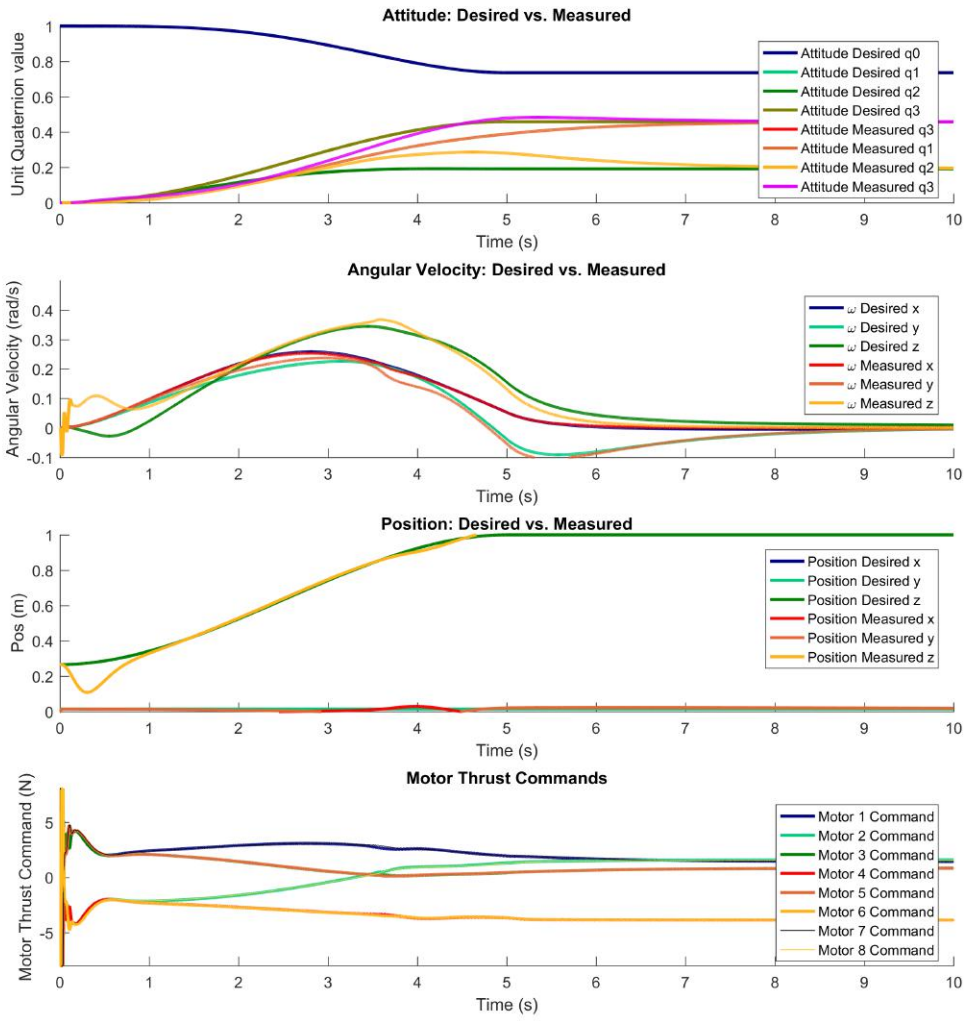

Figure 5.5: Simulation results for a 1m altitude hold while performing a 45 degree roll, pitch, and yaw

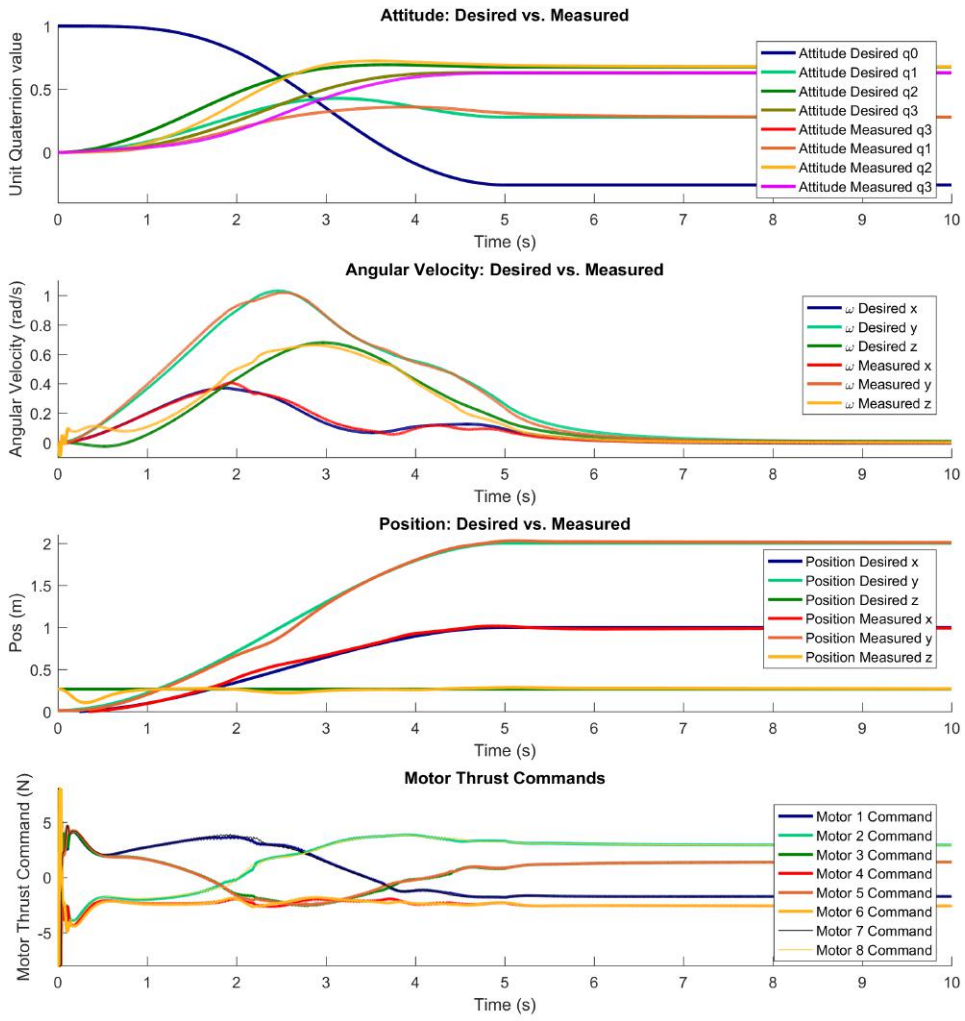

Figure 5.6: Simulation results for an arbitrary translational and rotational trajectory

Chapter 6

Energy Optimal Control Allocation

Since the omnicopter has six degrees of freedom with eight actuators, it is an over-actuated system with two degrees of redundancy. This means that, when feasible, there is a continuum of actuator thrust combinations that could produce a desired net force/torque. The combination of eight actuator forces can be thought of as an eight-dimensional actuator space. For a given desired force/torque, there exists a two dimensional null-space within the actuator space that may be traversed without impacting the desired net force/torque. This chapter discusses strategies for resolving this redundancy in actuation.

6.1 Control Allocation Using Minimum Norm

The work done in [5] employs the pseudoinverse of the Jacobian mapping to compute a suitable combination of actuator thrusts as shown in Equation 4.20. The use of the pseudoinverse effectively finds the solution vector $\boldsymbol{f}$ that has a minimal 2-norm.

$$\begin{aligned} min \quad & \sum_{i=1}^{8} f_i^2 \\ s.t. \quad & J\boldsymbol{f} = \boldsymbol{\zeta} \end{aligned} \tag{6.1}$$

This strategy provides a neat, closed-form solution to the problem.

$$\boldsymbol{f} = J^{\dagger}\boldsymbol{\zeta} \tag{6.2}$$

where the pseudoinverse is defined as $J^{\dagger} = (J^T J)^{-}1J^T$. It is also computationally efficient since the pseudoinverse of the Jacobian can be computed offline as this is a constant matrix when expressed in the omnicopter's body frame.

However, it can be argued that this method of control allocation does not fully take advantage of the flexibility offered by the null space. For one, the minimization performed in Equation 6.1 is blind to any constraints imposed by the actuators. This means that the pseudoinverse may generate an unrealizable solution, saturating the actuators, even if the desired force/torque command lies within the set of feasible force/torques. The control allocation can be re-formulated with the actuator constraints such that the optimization still tries to find a minimum norm solution, but also has the flexibility to explore the nullspace if the solution lies at the limit of the feasible set. An understanding of the actuator limits can also provide an even more obvious demonstration of redundancy. In the event that one

of the actuators fails, the limits of the actuator can be set to zero, finding a solution that allows the system to still fly, albeit with greatly reduced actuation in force and torque. The problem can formulated as the following convex quadratic optimization.

$$\begin{aligned} min \quad & \sum_{i=1}^{8} f_i^2 \\ s.t. \quad & J\boldsymbol{f} = \boldsymbol{\zeta} \\ & f_{min} \leq f_i \leq f_{max} \end{aligned} \tag{6.3}$$

6.2 Formulation of Energy Optimal Control Allocation

Although the above optimization can be used to successfully fly the omnicopter , the objective function does not have much of a physical meaning. In this section, the objective function is changed to minimize the energy consumed by the omnicopter. Energy optimal control allocation is critical for this kind of vehicle since its propeller configuration renders it highly inefficient, making flight times short.

The new objective function must express power in terms of thrust. The power, P_{motor}, consumed by a DC motor is function of the torque it produces, τ_m and its angular velocity ω,

$$P_{motor} = |\tau_m|\omega \tag{6.4}$$

For simplicity, it is assumed that the motor is in steady state, in which case the motor torque is equal to the drag torque caused by propeller drag and motor friction.

$$|\tau_{motor}| = |\tau_{drag}| = k\omega^2 \tag{6.5}$$

Equation 6.5 can be substituted back into Equation 6.4 to get

$$P_{motor} = k|\omega|^3 \tag{6.6}$$

The thrust f_i of an actuator can be related to the actuator's angular velocity [49, 50] as

$$|f_i| = \alpha\omega^2 \tag{6.7}$$

Finally, power is written in terms of actuator thrust by rearranging Equation 6.7 in terms of ω and substituting it into Equation 6.6

$$P_{motor} = \beta|f_i|^{(3/2)} \tag{6.8}$$

where β is a constant that consumes both the α and k constants. The formal optimization is now written as

$$\begin{aligned} min \quad & \sum_{i=1}^{8} |f_i|^{(3/2)} \\ s.t. \quad & J\boldsymbol{f} = \boldsymbol{\zeta} \\ & f_{min} \leq f_i \leq f_{max} \end{aligned} \tag{6.9}$$

Since the optimization must be solved in real time, it is important that the problem is convex. Convexity guarantees that a globally optimum solution can be found. From [51], second-order condition for convexity of the objective function holds if

$$\nabla^2 f(x) \geq 0 \tag{6.10}$$

The set of partial second derivatives of the objective function, also known as the Hessian, can be written as

$$\nabla^2 f(x) = \begin{bmatrix} \frac{3}{4}|f_1|^{-1/2} & 0 & \cdots & 0 \\ 0 & \frac{3}{4}|f_2|^{-1/2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{3}{4}|f_8|^{-1/2} \end{bmatrix} \tag{6.11}$$

It can be seen that the Hessian satisfies the condition from Equation 6.10 and therefore the objective function is convex. The feasible set defined by the linear equality and inequality constraints is also convex, hence the problem is a convex optimization problem.

6.3 Extension of Energy Optimization at Actuator Limits

The optimization formulation can be further extended to provide a degree of robustness when the requested force/torque is infeasible. In this case, the equality constraints can be relaxed by introducing a new variable u, which represents the 2-norm of the error between the the desired force/torque and the feasible force torque. When the solution lies with the feasible set of actuator commands, variable u should be zero such that there is no error between the desired and actual force/torque outputs. However, in the case where the requested force/torque is not feasible, u should be minimized.

This new optimization can then be written as

$$\begin{aligned} min \quad & \beta \sum_{i=1}^{8} |f_i|^{(3/2)} + u \\ s.t. \quad & \|J\boldsymbol{f} - \boldsymbol{\zeta}\| \leq u \\ & f_{min} \leq f_i \leq f_{max} \end{aligned} \tag{6.12}$$

where β defines some gain to weight the objective function between minimization of energy and error. Ultimately β should be small to ensure that error has a much higher weighting. This prevents a situation where error is made large in order to decrease power consumption. In the trivial case, this would result in a large error in order to allow all actuators to provide zero force and have zero power output. It can be noted that this extended formulation is still convex allowing it to be solved by the same routines as those used in the previous formulations.

6.4 Convex Solver Implementation

Since the mapping from the desired force/torque to motor thrusts occurs at the lowest level of control, it is critical that the optimization can be solved in a computationally efficient manner. Python provides a convenient libraries to describe the optimization to a solver.

These solvers can be written in a language other than Python (often C or C++), and wrapped for convenient use within the Python script. A Python solution was chosen because it allowed for fast prototyping by providing access to a variety of solvers. The objective function and Jacobian matrices along with an arbitrary starting point within the feasible set for the decision variables are provided to the algorithm. As will be detailed in Chapter 7, the solver is run on an off-board desktop computer to determine motor commands before sending them to the omnicopter. It must be efficient enough to provide solutions at a rate of approximately 340Hz. To ensure that amount of time required by the solver remains bounded, an accuracy threshold and a maximum number of allowed iterations are set. Whichever criteria is met first terminates the algorithm. In the future, investigation may be focused towards the implementation of other even more efficient solvers to allow the optimization routine to run on the omnicopter itself.

Chapter 7

System Implementation

This chapter details the hardware and software developments necessary to verify the models, control architectures, and optimizations established in Chapters 4, 5, and 6. The implementation can be broken into two major parts, namely the test bed itself and the omnicopter vehicle. The first section covers design considerations necessary for the creation of the test bed. This system provides a generic environment suitable for the verification of any small UAV. The next section describes the hardware and firmware components required to assemble the omnicopter . The final section discusses fail-safe mechanisms critical to guarantee safety in the operation of any UAVs flying on the test bed.

7.1 Test Bed Development

The test bed to be developed must overcome a variety of challenges to allow for stable flight. The first section lays out the over all system architecture and details the flow the data between hardware and software units. The next section explains strategies to establish a robust real-time data link between sensors, controllers, and actuators to ensure that the

closed-loop delay stayed within reasonable bounds. The final section explores the details of the software modules developed for the off-board PC. Such a system requires careful testing and development. Not only are more than 14 separate processing units involved, but the entire system runs on many asynchronous threads to meet both hard and soft real-time constrains, operating at a closed loop rate of 340Hz.

7.1.1 Test Bed Architecture

Data flows through the test bed along two major streams, namely the data acquisition and command streams. The data acquisition stream is responsible for collecting data from all necessary sensors to estimate the position and attitude of the omnicopter . The position is obtained using an Optitrack Motion capture system, which uses IR cameras to detect IR reflective markers mounted to the omnicopter . Eight of these IR cameras send data at a rate of 120Hz to a desktop PC where a tracking software, called Motive, uses the data to calculate the drone's position attitude. It is desirable to run the innermost angular velocity controller at rates greater than 120Hz, motivating the addition of an on-board IMU to the sensor network. The IMU supplies both an attitude estimation as well as the omnicopter 's angular velocity. All sensor data is sent to a Python application where sensor feedback is filtered and used in conjunction with a trajectory planner to make control decisions. The details of this application are discussed in Section 7.1.3.

Once the Python application has made a control decision, it sends the motor PWM commands over a serial line to the omnicopter. These commands are encoded between 1000-2000 with 1500 being the neutral command and the upper and lower bounds being maximum and minimum actuator thrust, respectively. The flight controller receives these commands and echos them in the form of a true PWM signal to each respective ESC to

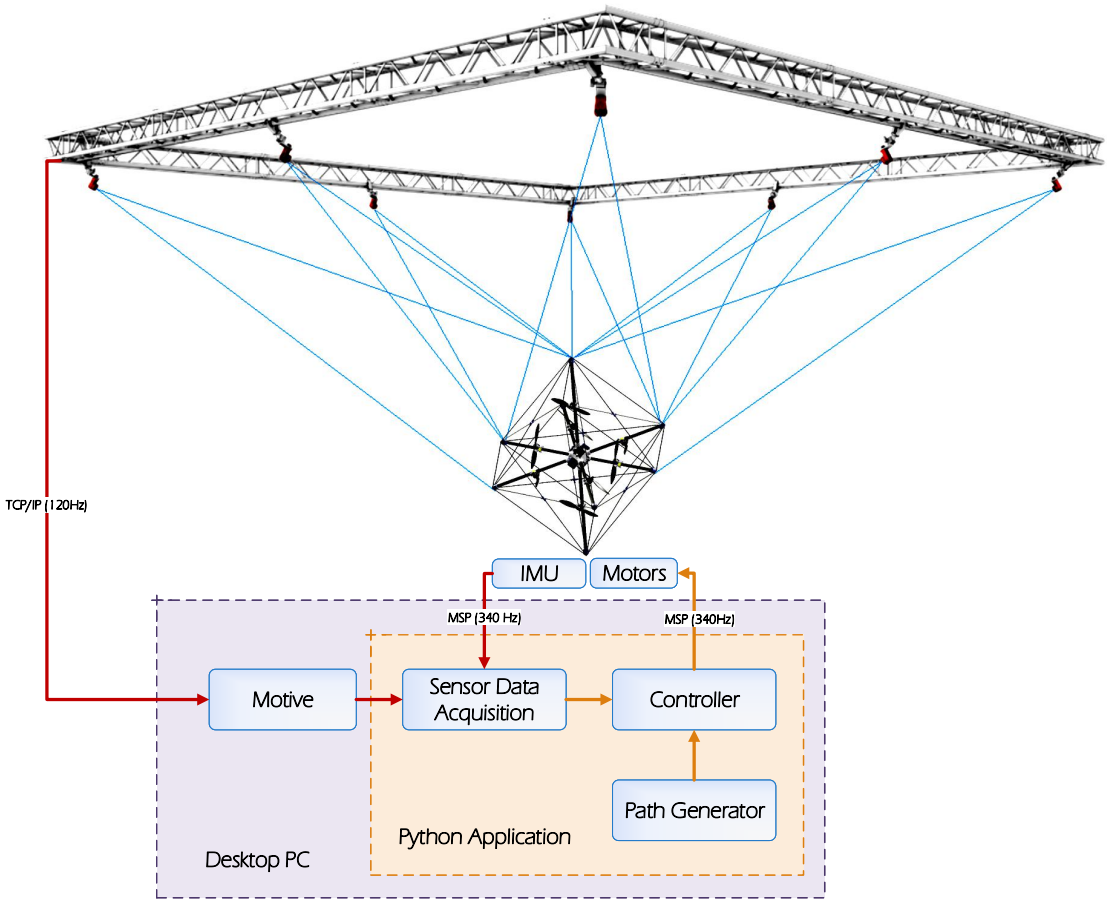

Figure 7.1: Block diagram of experimental test bed

drive the actuator. It is paramount that this entire process happen with minimal delay as this has a direct impact on the system performance.

The modular design of the test bed enables it to be adapted to a variety of future experiments as new sensing and control strategies are investigated. Each module can be easily swapped without significant changes to the system's framework.

7.1.2 Communications

Both hardware and software design decisions are predominantly driven by the need for hard real-time performance and robust communication links. Although data throughput on these links is low bandwidth, the low latency and real-time requirements become non-trivial challenges. Wireless links, although able to achieve high bandwidths, are notorious for their unreliability due to unpredictable latency spikes. As a result, conventional local area networking could not be used. Instead, attention was turned towards the Lairdtech RM024. This embedded wireless module can be tailored to the exact packet size of the transmission to minimize overhead and delay. The unit hops over 43 frequency bands to ensure a robust link. This module was used in the initial experiments performed on the test bed to allow either single or groups of drones to fly wirelessly. However, this still requires that the lowest control loops be located on the drone. To allow for rapid software development, the controller is kept on the desktop PC and communication is made via a serial USB cable from the computer. At the time of writing, development of an on-board controller is complete and is now in the testing/verification phase for wireless flight. The switch between wireless and cable solutions highlights the modularity of the framework as no code changes (except a COM port change) is required for the switch.

After deciding the hardware-level serial communication interface, a software-level protocol needed to be implemented to receive data packets from the flight controller. The Multiwii Serial Protocol (MSP) was selected due to its minimalist packet structure. The protocol comes native with the flight controller and has the ability to send and receive up to 256 unique commands containing up to 256 bytes of data. The MSP protocol also contains a single 1-byte XOR checksum at the end of the packet for simple yet effective error checking [52]. A library was written in Python to interface the flight controller with the

system PC using this protocol. The other Python communication module used socket programming to receive position and attitude data over a software link from motive and make it available to the Python application.

7.1.3 Off-board Software Development

The Python application mentioned in Section 7.1.1 is laid out in greater detail in Figure 7.2. Python was chosen over conventional C as it allows for rapid prototyping due to intuitive semantics and does not need the user to explicitly compile the software. The *main.py* module should be called to initialize all other modules. This module runs the main infinite while loop and is responsible for orchestrating function calls to other modules. This while loop starts by evaluating all fail-safe conditions to ensure the system is still operating safely. If any fail-safe is triggered, including a message from *EStop.py*, the system automatically shuts down and generates plots of the logged data. Details on the fail-safe mechanisms are reported in Section 7.3. At the beginning of an experiment, the actuators are ramped up to nearly cancel out the gravity vector. This ensures that when the controller is put in the loop, the gravity feed-forward term does not create a large discontinuity in motor commands.

The Main function then proceeds to collect data from system sensors and send to the signal processor. The signal processor is responsible for filtering and transforming sensor data. Transformations include rotating axes since the Optitrack and omnicopter coordinate frames use different conventions. The *Signal_Processor.py* also performs numerical differentiation on signals such as the omnicopter position to obtain velocity. Noisy signals such as the angular and translational velocity are passed through *Filter.py*. These signals are passed through 5th and 7th order low-pass FIR filters with cut-off frequencies of 150 Hz and 75 Hz respectively. As discussed in Section 5.2, care must be take to balance filter

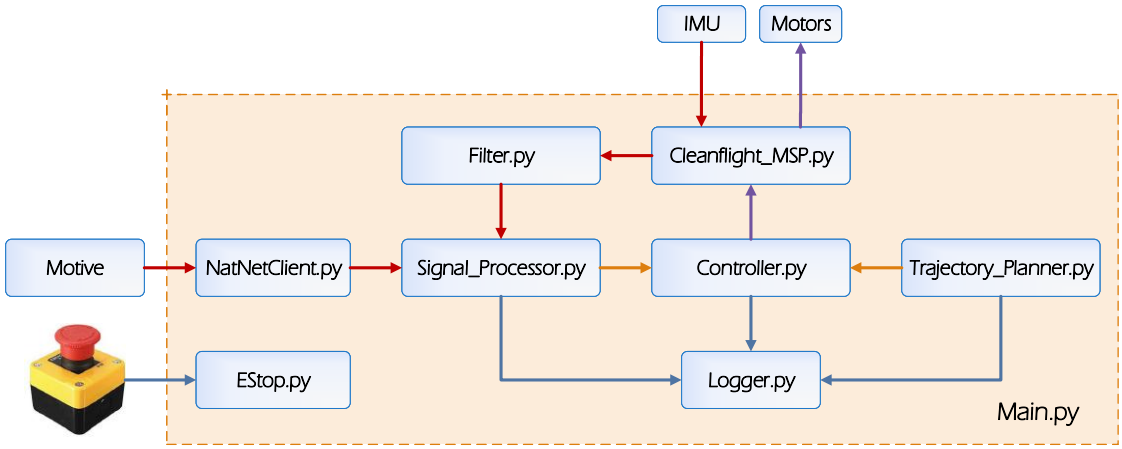

Figure 7.2: Block diagram of Python application

performance with subsequent delay. The *Trajectory_Planner.py* module also provides the signal processor with a smooth desired trajectory for each of the translational/attitudinal states plus their derivatives. The derivation of these trajectories is detailed in Section 8.1. The *Controller.py* module takes all conditioned signals from *Signal_Processor.py* and uses them in conjunction with the omnicopter model to generate desired actuator commands. The commands are then encoded into an MSP packet by the *Cleanflight_MSP.py* module and sent serially to the omnicopter. It may be noted that *Cleanflight_MSP.py* is also responsible for receiving sensor data packets from the omnicopter to be collected by the signal processor. The *Logger.py* module stores data of interest from the other module in a comma-separated values (CSV) file and generates plots following an experiment.

7.2 Omnicopter Development

This section discusses the mechanical and hardware design decisions made during the development of the omnicopter . The final design yields a viable solution to accommodate the

propeller configuration proposed from Section 3.1. In the final part, a justification is made for the selected flight controller and firmware support.

7.2.1 Mechanical Design of the Omnicopter

The mechanical design must be simple and robust to minimize the number of single point structural failures. The omnicopter was first designed in CAD software as shown in Figure 7.3. Components such as the central hub and connectors are 3D printed using PLA plastic. The high quality prints can replicate the CAD design with 0.1mm resolution and the part proves to be robust over numerous tests and crashes. Extruded carbon fiber rods are used for the arms of the omnicopter for their high strength to weight ratio. The selected rods have square profiles such that there are quantized orientations in which they can be inserted into the central hub. This ensures that each of the actuators face precisely at the correct angle within the omnicopter. All rods and connectors are anchored together with Original Gorilla Glue formula. The flight controller is separated from the central hub by several layers of 3M double sided tape to attenuate high-frequency mechanical vibrations from reaching the IMU. The final assembled omnicopter is presented in Figure 7.4.

7.2.2 Considerations in Component Selection

The component selection for the omnicopter must balance a number of design considerations. Most importantly, power to weight ratio should be designed such that the omnicopter can provide a net thrust acceleration at least twice that of gravity in any given direction to perform moderately aggressive trajectories. Section 3.1.2 shows that in the direction of minimum actuation, the chosen propeller configuration will provide a net thrust vector equal to at least 2.67 times a single propeller thrust. The balancing act now becomes

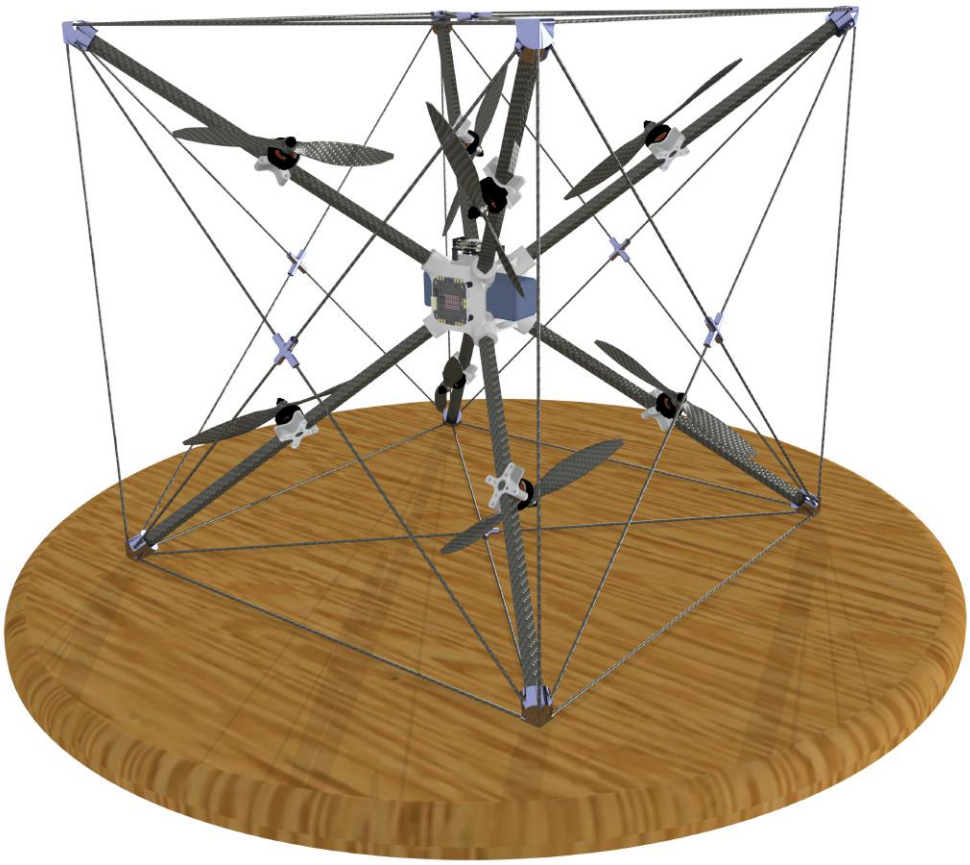

Figure 7.3: Preliminary CAD design of the Omnicopter

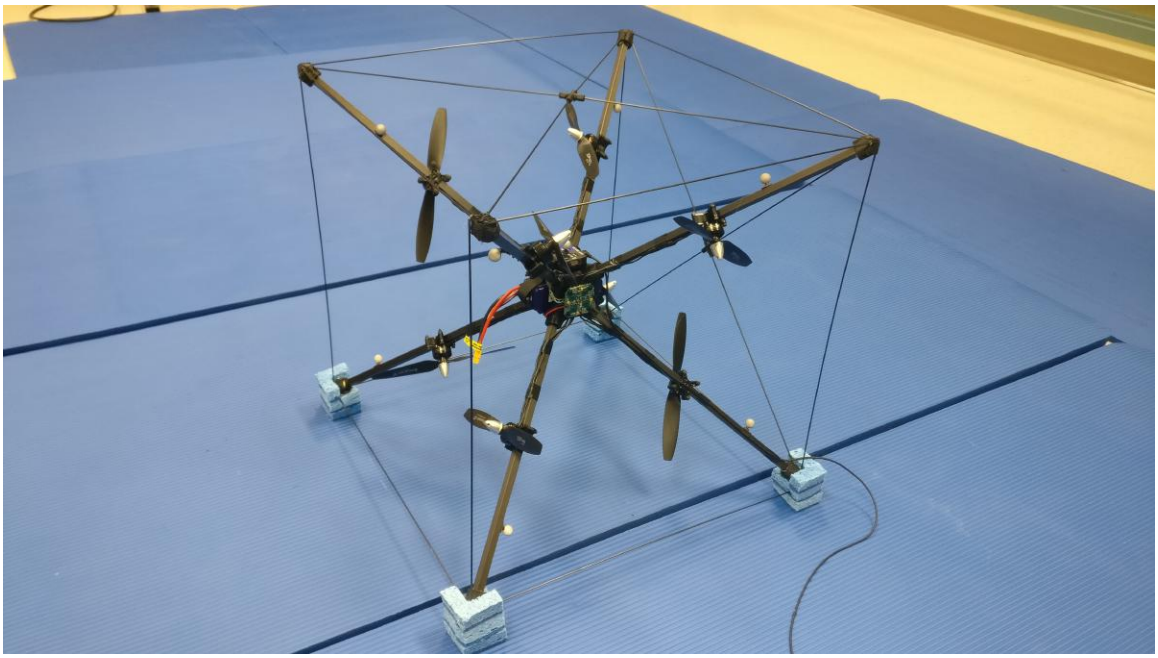

Figure 7.4: Assembled Omnicopter

choosing a motor that is light, yet can provide adequate thrust to ensure the omnicopter has at minimum $2g$ of actuation in any direction. The selected motor is the Cobra CM-2206/30-V2 1400Kv motor from Innovative Designs. From benchmark tests conducted in Section 4.1, it was found that this motor can safely provide a maximum thrust of 8N running on a 4S battery.

From Section 3.1.2, the null-space can be used to improve the distribution of actuation in the force-torque space. From this analysis, it was determined that the omnicopter's direction of minimum actuation could provide a force equal to 3.5 times the maximum thrust of a single actuator. This implies that at minimum the omnicopter has a thrust to weight ratio of $\frac{3.5\times 8N}{1.126\times 9.8} = 2.54$, which lies within the actuation requirements. Table 7.1 shows the breakdown, by mass, of each of the omnicopter 's hardware components. It is a balancing act to choose components that provide a good thrust to weight ratio.

Each actuator may pull up to 15A of current. It was therefore necessary to choose a battery capable of handling up to 120A of current in a worst-case scenario. For this reason, the 2200mAh battery has 45C-90C rating capable of handling up to 198A is chosen. Each ESC is similarly specified to handle the current demands.

Component	Unit Weight (g)	Amount	Total Weight (g)
Cobra 1400Kv Motor Assembly	41.5	8	332
Graupner 8x4.5 Bidirectional Propellers	7.0	8	56
Afro 4-in1 20A ESC	31.0	2	62
Turnigy 2200mAh 4S 45-90C Lipo	226	1	26
SPRacing F3 Flight Controller	20.0	1	20
Mechanical Frame	400	1	400
Wiring	30	N/A	30
Total			1126

Table 7.1: Mass breakdown of omnicopter hardware components

7.2.3 On-board Firmware Development

The open-source Cleanflight project is the firmware of choice as it supports 50+ hardware platforms and is specifically designed with the intention of having 'clean' and readable code. This firmware is actively maintained by the community and provides a fantastic framework for development. Although Cleanflight cannot be strictly considered a real-time operating system (RTOS), it does have a real-time scheduler capable of running tasks based on priority level and desired frequency. Such tasks include gathering IMU and serial port data, performing sensor fusion for attitude estimation, sending commands to the actuators, etc. The specific flight controller (FC) hardware used is the Seriously Pro SPRacing F3. Boasting a 72MHz STM32F303 MCU, this flight controller is more than capable of sampling the gyro at 32kHz and updating motor commands at rates beyond 2kHz.

A good understanding of the Cleanflight firmware was necessary before surgically modifying modules to suit the omnicopter's need. Careful planning allowed for minimal code changes to create custom MSP messages to be received from and transmitted by the FC. This allows the off-board controller to receive the required sensor feedbacks from the FC and send actuator commands back. Another area requiring modification were the algorithms used to estimate the omnicopter 's attitude. Since the test bed is indoors, an on-board magnometer is inaccurate. Integration of the on-board gyro tends to drift over time. Therefore, modifications are made in the algorithm to correct this drift using attitude data from the Optitrack system.

7.3 Fault Detection and Fail-Safe Mechanisms

An aerial robotics test bed comes with inherent dangers during experimentation. If any of the real-time systems fail, the UAV could behave unpredictably and cause damage to itself, the test bed, or those working with the system. To mitigate these risks, a number of fail-safes have been built into all levels of the test bed. At the lowest level, the on-board flight controller constantly monitors the motor commands it receives. If for some reason, the communication link between the flight controller and off-board controller was lost, the flight controller would time out after 100ms and cut power to the drone. The drone would immediately stop operating and fall out of the air. Although this dead stop fail-safe may cause damage to the drone, simplicity in fail-safe design was emphasized in order to ensure robustness. In addition, the experiment space is covered with half-inch thick mats to absorb some of the shock from a falling drone. Fortunately, this communication link has never crashed so the system has not yet failed in this mode during experimentation. An emergency stop button is also mounted beside the test bed PC. The E-stop button must

be pulled out to allow the drone to arm and fly. At any point in the experiment, one can press down onto the locking button to immediately deactivate the system. Other safety measures exist in the Python application itself. If the drone deviates too far from the desired trajectory, the experiment will terminate. Actuator commands also pass through saturation functions to ensure that unreasonably large commands are not sent to the drone. Over the hundreds of tests that have been performed, this system of fail-safes has proven as an effective means to keep the equipment and people involved safe. The drone has failed in all modes except the flight controller level mode and no damage to equipment or injury has ever resulted.

Chapter 8

Experimental Results and Discussion

In this chapter, the omnicopter's performance is evaluated by flying it in an aerial robotics test bed. The first section of the chapter briefly describes the methods used to generate smooth trajectories for the experiments. The experimental results are then presented in the following two sections and organized into two categories. First, the controller and actuator models detailed in Chapters 4 and 5 are verified by testing a number of trajectories including attitude holds and translations. It is then shown that the omnicopter can follow an arbitrary combination of attitude and position commands simultaneously. The second set of results demonstrates the advantages of formulating the problem of actuation redundancy resolution as an optimization problem. In this section, energy savings are achieved for certain trajectories and the system is shown to be robust in the event of an actuator failure. The last section contrasts the differences between filtered and unfiltered sensor inputs, highlighting the importance of appropriate signal conditioning.

8.1 Trajectory Planning

The trajectories generated for the experiments follow a simple set of rules to compute a smooth trajectory. Since the system is fully actuated, six of the omnicopter's states can be controlled independently. Three of these are in position and three in attitude. For each of these states, a setpoint is given at discrete times during the trajectory. The trajectory planner then computes all intermediate points necessary to connect the two setpoints.

Each state's trajectory is computed as a function of time, $f(t)$.To ensure that the trajectory is smooth, four constraints are placed on the trajectory [53]. Given a starting position of λ_0 at t_0 and a final position of λ_f at t_f, these constraints may be written as

$$\begin{aligned} f(t_0) &= \lambda_0 \\ f(t_f) &= \lambda_f \\ \dot{f}(t_0) &= \dot{\lambda}_0 \\ \dot{f}(t_f) &= \dot{\lambda}_f \end{aligned} \tag{8.1}$$

To satisfy these four constraints, a third order polynomial is selected of the form

$$f(t) = at^3 + bt^2 + ct + d \tag{8.2}$$

Given the constraints in Equation 8.1, the coefficients of Equation 8.2 can be written explicitly as

$$
\begin{aligned}
a &= -\frac{2\lambda_0 - 2\lambda_f - \dot{\lambda}_0 t_0 - \dot{\lambda}_f t_0 + \dot{\lambda}_0 t_f + \dot{\lambda}_f t_f}{(t_0 - t_f)^3} \\
b &= \frac{3\lambda_0 t_0 + 3\lambda_0 t_f - 3\lambda_f t_0 - 3\lambda_f t_f - \dot{\lambda}_0 t_0^2 - 2\dot{\lambda}_f t_0^2 + 2\dot{\lambda}_0 t_f^2 + \dot{\lambda}_f t_f^2 - \dot{\lambda}_0 t_0 t_f + \dot{\lambda}_f t_0 t_f}{(t_0 - t_f)^3} \\
c &= \frac{\dot{\lambda}_f t_0^3 - \dot{\lambda}_0 t_f^3 - 6\lambda_0 t_0 t_f + 6\lambda_f t_0 t_f - \dot{\lambda}_0 t_0 t_f^2 + 2\dot{\lambda}_0 t_0^2 t_f - 2\dot{\lambda}_f t_0 t_f^2 + \dot{\lambda}_f t_0^2 t_f}{t_0 - t_f)^3} \\
d &= -\frac{\lambda_0 t_f^3 - \lambda_f t_0^3 + \dot{\lambda}_0 t_0^2 t_f^2 - \dot{\lambda}_f t_0^2 t_f^2 - 3\lambda_0 t_0 t_f^2 + 3\lambda_f t_0^2 t_f - \dot{\lambda}_0 t_0 t_f^3 + \dot{\lambda}_f t_0^3 t_f}{(t_0 - t_f)(t_0^2 - 2t_0 t_f + t_f^2)}
\end{aligned}
\tag{8.3}
$$

This strategy can be extended beyond calculating trajectories between setpoints and applied to any parametric function. An example would be having the drone follow a circular trajectory in the xy plane. The parametric equations for such a trajectory are

$$
\begin{aligned}
x(t) &= sin(t) \\
y(t) &= cos(t)
\end{aligned}
\tag{8.4}
$$

However, the issue with entering into such a trajectory is that the velocity in the x direction initially starts as non-zero. This can create a discontinuity in the desired velocity leading

to issues in tracking. To avoid this, t is replaced by $f(t)$

$$x(t) = sin(f(t))$$
$$y(t) = cos(f(t)) \tag{8.5}$$

and the constraints from Equation 8.1 can be written as $\dot{f}(t_0) = 0$ such that $\dot{x}(t_0) = \dot{y}(t_0) = 0$. This allows for a smooth transition from a position hold into the circular trajectory.

8.2 Experimental Results from Propeller and Airflow Model

This section evaluates the performance of the propeller and airflow models proposed in Chapter 4 coupled with the position and attitude control architectures designed in Chapter 5. First, the omnicopter demonstrates its independence between rotational and translational states by following a trajectory in one while holding the other. The benefits of using a quaternion-based controller is also revealed in the 90 degree pitch trajectory. The omnicopter then follows an arbitrary trajectory in both translation and attitude to fully demonstrate its independent control over these six states.

8.2.1 90 Degree Pitch

The 90 degree pitch hold demonstrates the power of using a quaternion representation for rotations instead of the conventional Euler notation. Euler representation is suitable for conventional drones since they experience relatively small attitude commands from the horizontal. However, the omnicopter can undergo any amount of rotation and can therefore run into a gimble lock scenario as discussed in Section 3.2. In a preliminary implementation of the controller, Euler angles were used to represent the omnicopter's attitude, making

rotations around the current z, y, and x axes, respectively. Unfortunately, the singularity for this representation exists with a rotation of 90 degrees on the y axis as this rotates the new body x axis to be collinear with the old body z axis. This means that the Euler representation becomes inaccurate and then unsolvable as the drone approaches a 90 degree pitch. This is demonstrated in the second plot of Figure 8.1 where it can be seen that the measured roll and yaw significantly diverges from 0 degrees between 15-25 seconds. This is the same time that the pitch is approaching 90 degrees. The quaternion representation depicted in the third plot of Figure 8.1 has no such issue. Indeed the ability for the omnicopter to perform this trajectory demonstrates the necessity to use quaternions in place of Euler representation.

The first plot of Figure 8.2 shows the desired angular velocity, $\boldsymbol{\omega}_d$, sent to the inner angular velocity controller. The measured $\boldsymbol{\omega}$ from the onboard gyroscope oscillates around this control signal (due in part to the large vibrations present in the system), but ultimately follows it. The second plot shows the change in net force requested as the omnicopter undergoes a pitch. This net force is represented in the omnicopter's body frame and therefore the vector must move from being in the positive body z to negative body x to cancel the gravity vector. This change in net force is further reflected in the third plot where it can be seen that some motors, namely actuators 6 and 7, must reverse their direction to provide a positive thrust between 15 and 25 seconds.

8.2.2 360 Degree Roll

The 360 degree roll shown in Figure 8.3 demonstrates the full attitudinal range of the omnicopter while maintaining a position hold. Although positional error reaches up to 10cm at certain points in the trajectory, the omnicopter stably tracks the entire rotation

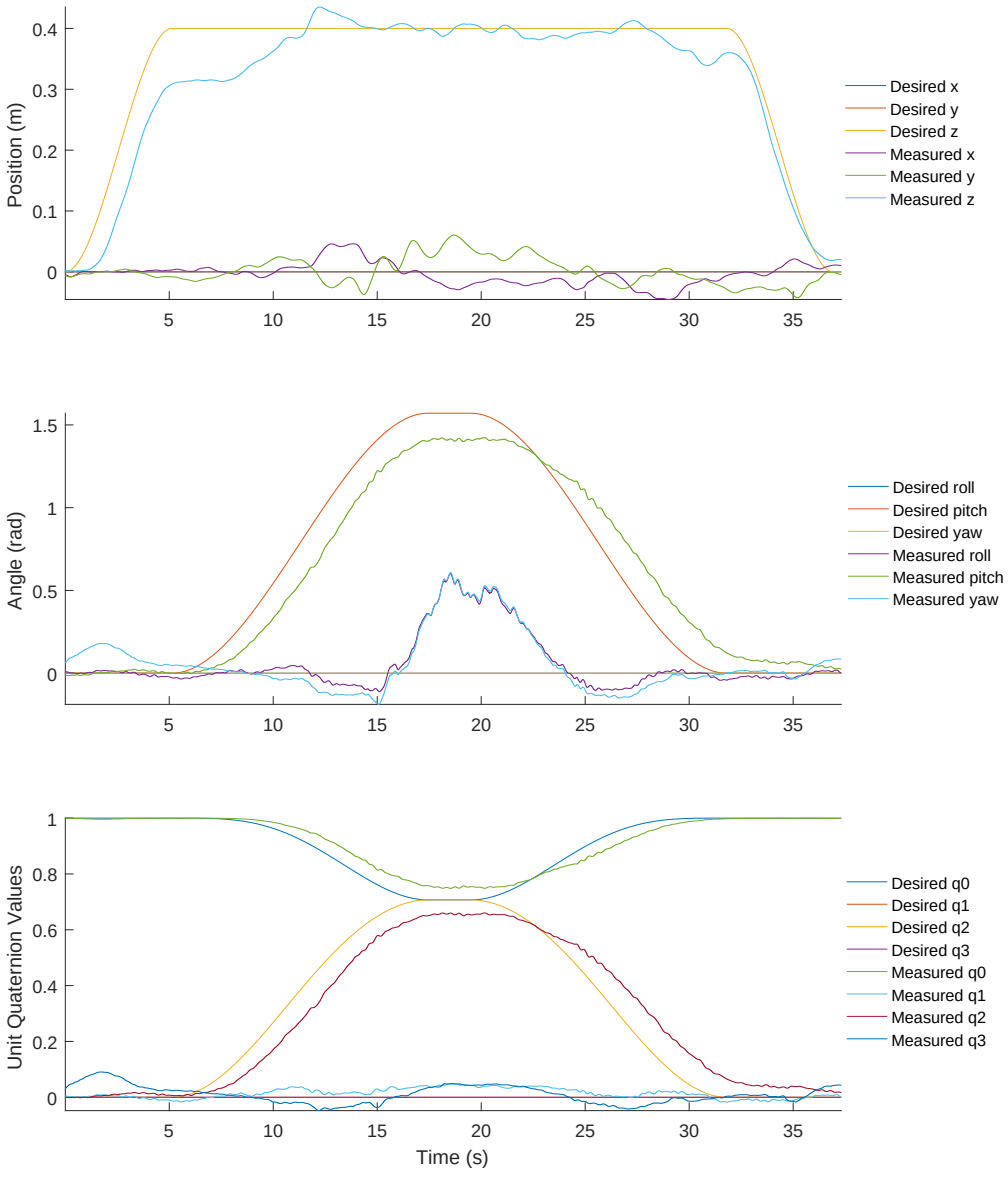

Figure 8.1: Position and attitude tracking for a 90 degree pitch

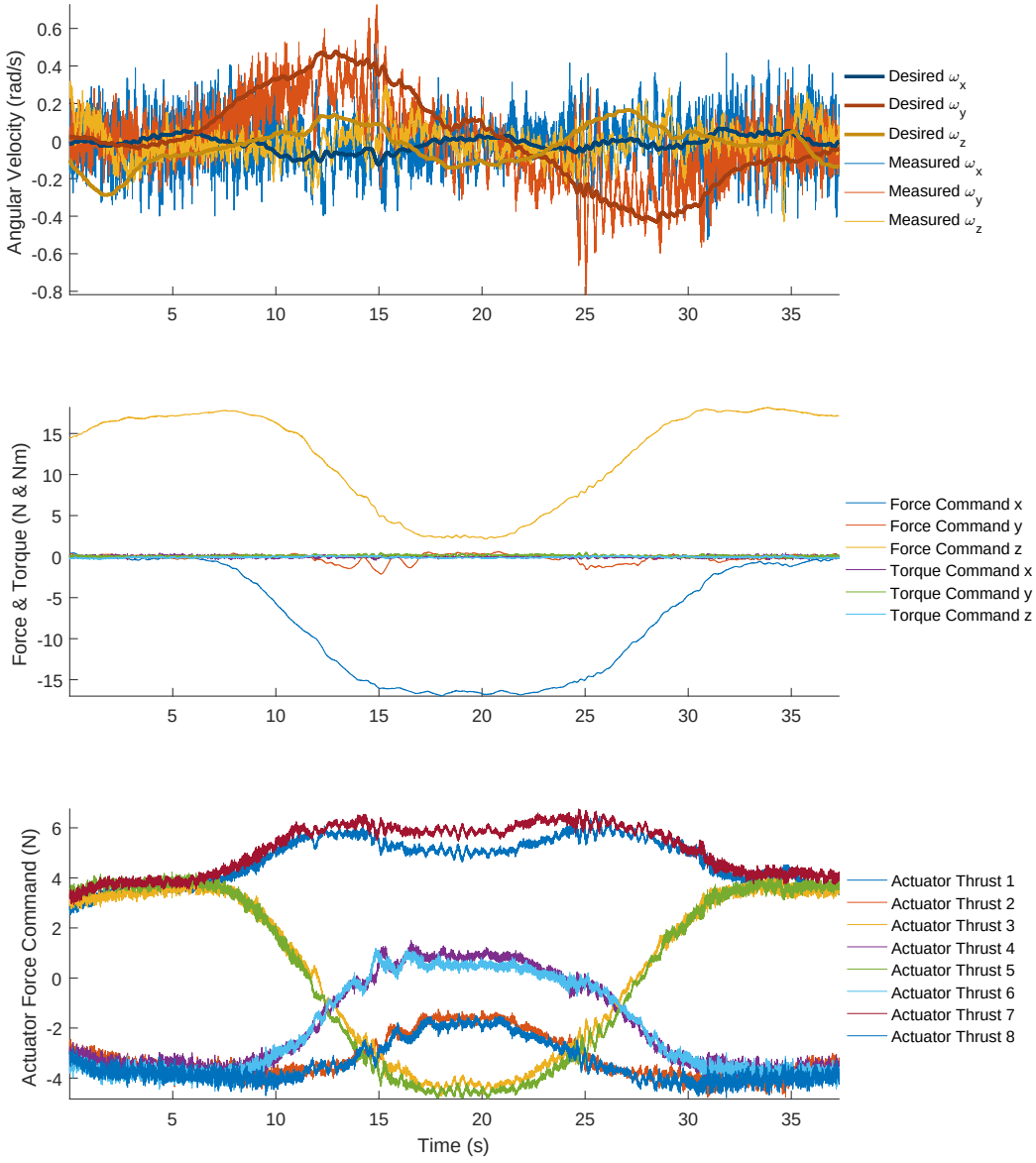

Figure 8.2: Angular velocity, net force/torque, and actuator thrust commands for a 90 degree pitch

within an acceptable degree of error.

8.2.3 Translational Trajectories

It has been shown that the omnicopter can follow attitude trajectories while holding a position set point. In this part, the omnicopter follows both a circular and square trajectory while holding an attitude set point. Figure 8.4 shows a plot of the x vs. y displacement of the drone as it follows a circular trajectory. The circle is 2m in diameter and is completed in 10 seconds. The first plot of Figure 8.5 shows that the xy error is reasonably small with the maximum reaching 7cm at 15s. The average error is under 5cm for the trajectory. As discussed in Section 8.1, a smooth polynomial is used as the parametric input ensures there are no discontinuities in velocity. This is verified in the second plot of Figure 8.5. The third plot demonstrates that the copter achieves a desired 0 degree rotation in roll, pitch, and yaw with a maximum 0.15 radian error and an average error magnitude beneath 0.6 radians. Furthermore there is no obvious correlation between attitude and position/velocity demonstrating that the omnicopter performs the translational trajectory independent of its attitude.

In a similar fashion, the omnicopter traverses around a square trajectory in Figures 8.6 and 8.7. For each side of the square, the omnicopter must translate 1m from a standstill. This somewhat aggressive maneuver causes a small, but acceptable overshoot. Control parameters were kept generic and not specifically tuned for any particular trajectory as such a customization is unrealistic in real application.

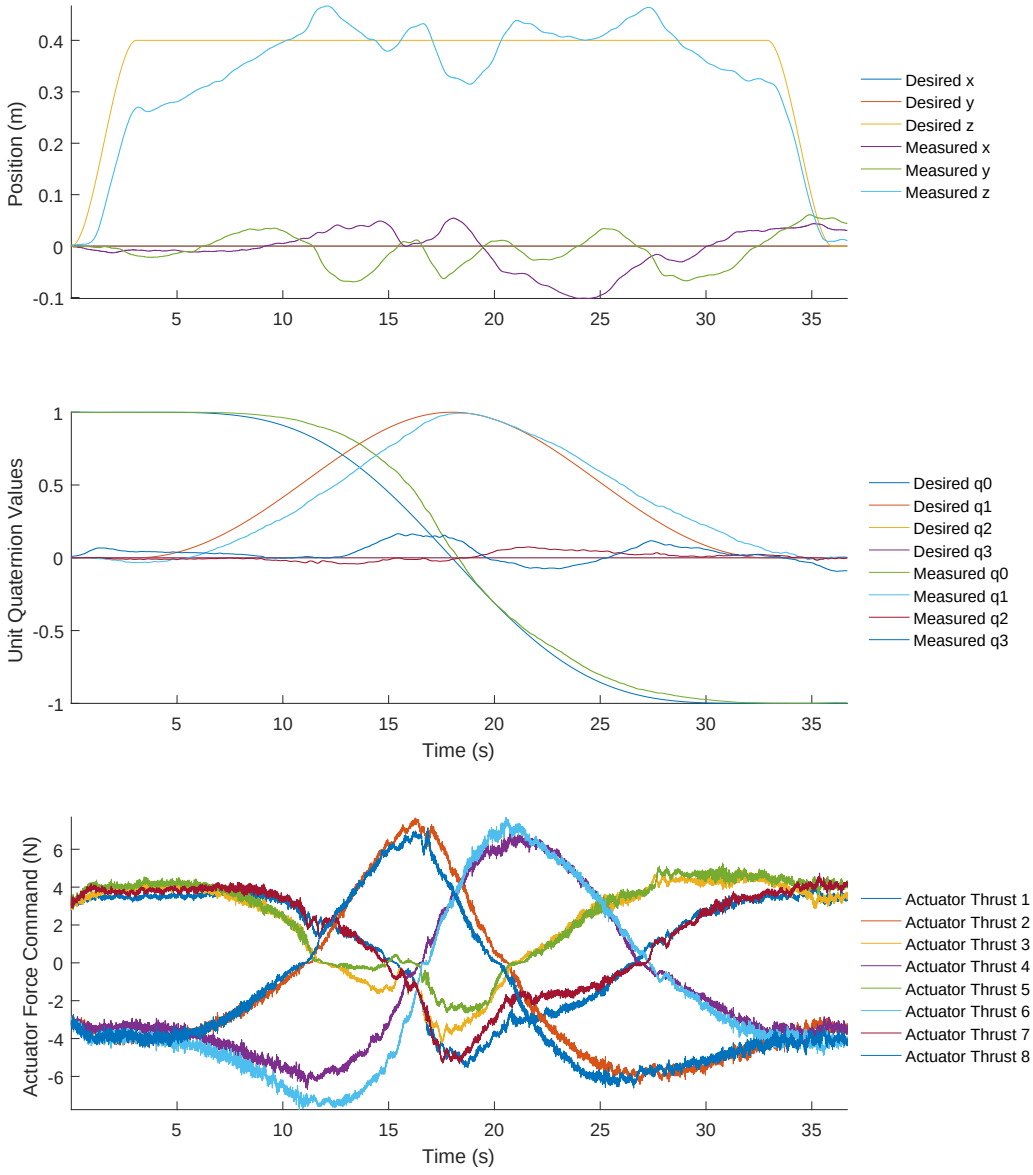

Figure 8.3: Position and attitude tracking and actuator force commands for a 360 degree roll

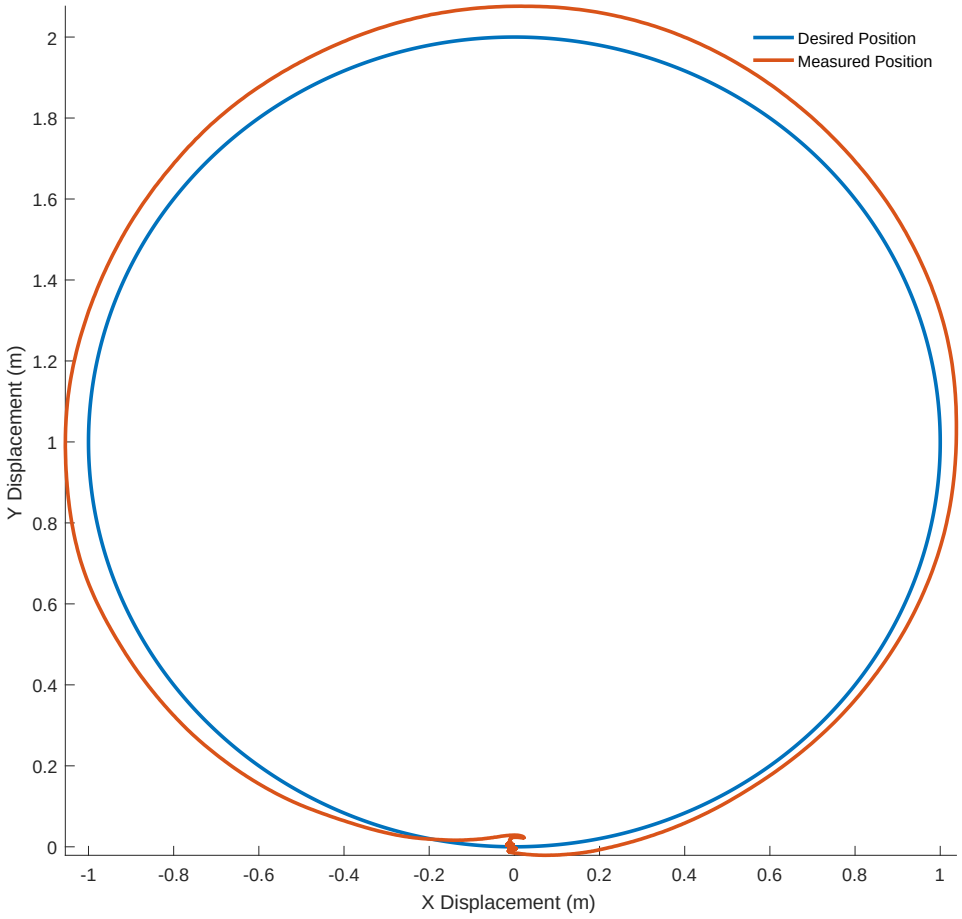

Figure 8.4: Position tracking for a circle with a 2m diameter

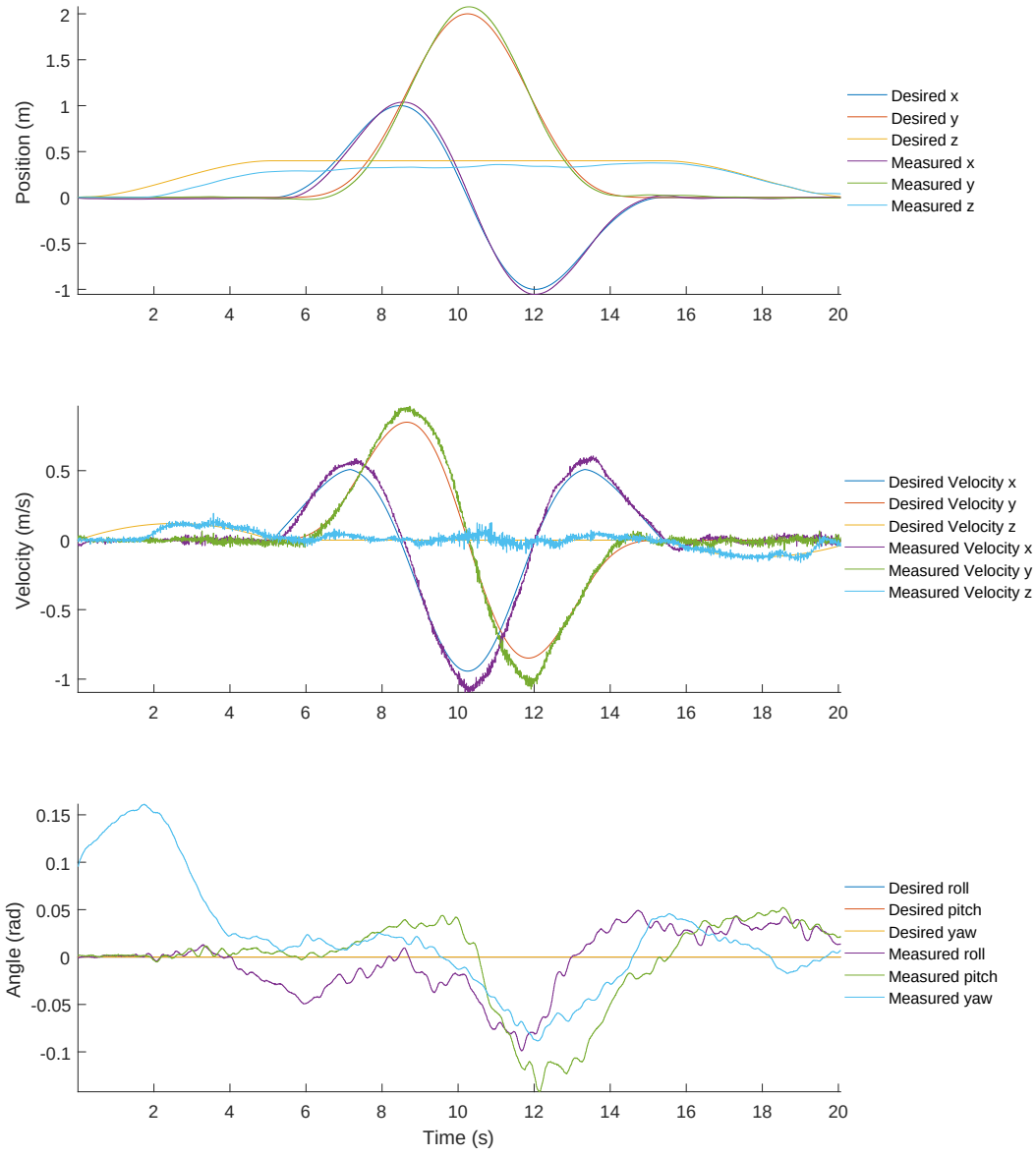

Figure 8.5: Position, linear velocity, and attitude tracking for a 2m diameter circle

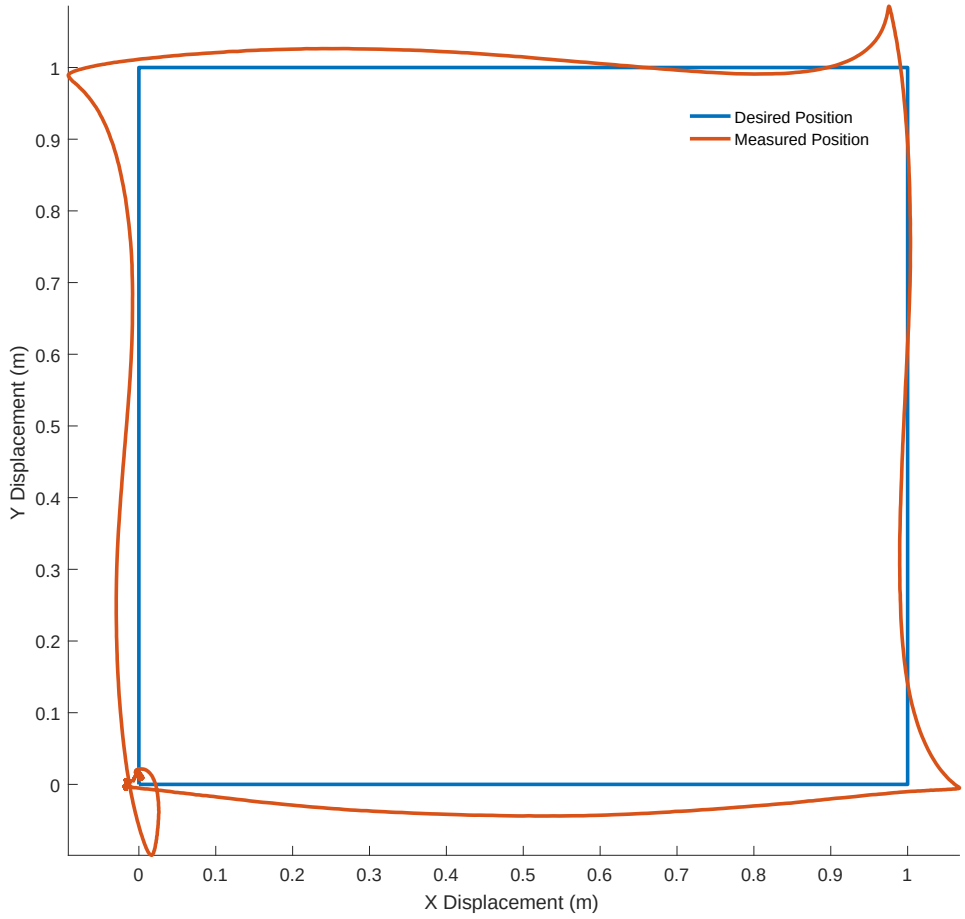

Figure 8.6: Position tracking for a square with 1m side length

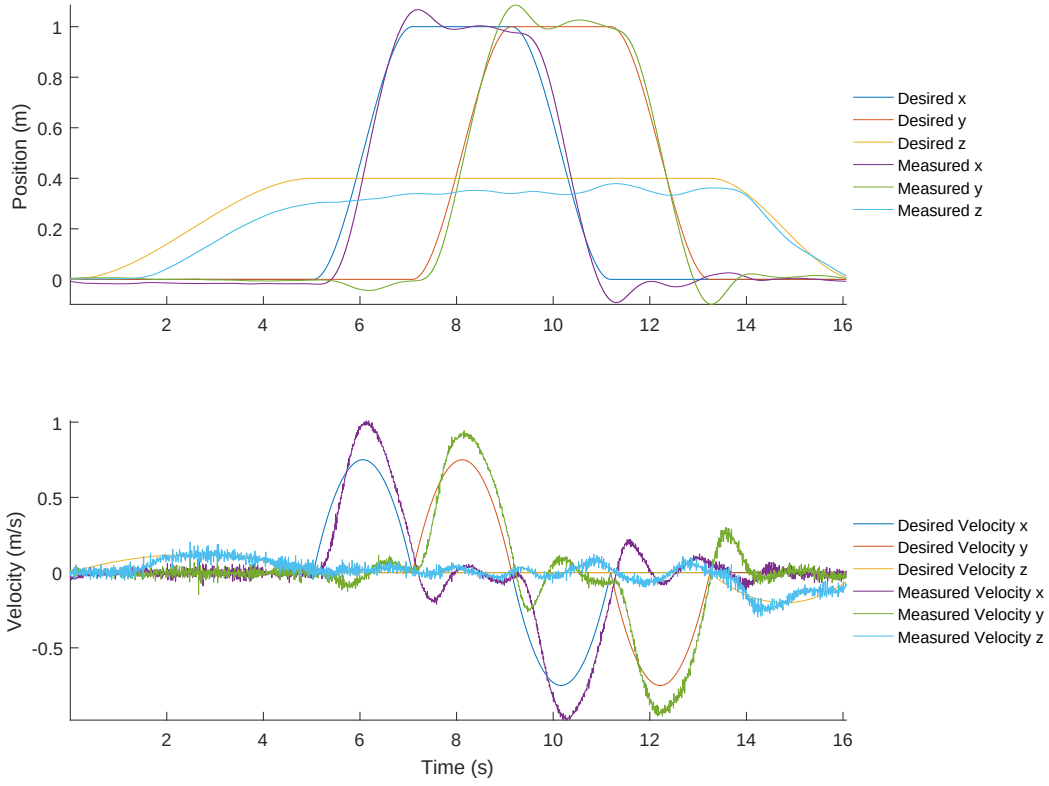

Figure 8.7: Position, linear velocity, and attitude tracking for a square with 1m side length

8.2.4 Arbitrary Trajectory

Figure 8.8 demonstrates the omnicopter is also capable of following an arbitrary trajectory in both position and attitude.

8.3 Experimental Results from Energy Optimization

Chapter 6 explored control allocation strategies used to resolve the redundancy property of the omnicopter in an energy optimal way. Up to this point in the results, the pseudoinverse of the Jacobian was used to resolve the redundancy. This section demonstrates some of the energy savings that may be achieved using the optimizations formulated in Chapter 6 implemented in a real-time solver. Another significant advantage of having an over-actuated system is its inherent robustness to an actuator failure. Even if an actuator stops operating, the omnicopter is still fully actuated with only seven actuators. Although the available force-torque space is reduced, the Jacobian still remains well conditioned with a condition number of 6.9124 if any one of the eight actuators fail. Two actuators may fail and the system may remain fully actuated; however, this greatly depends on which two failed as certain combinations can lead to an under-actuated configuration. The available force/torque space is even further reduced and keeping the omnicopter stable could become a serious challenge. For these results the failure of one actuator is investigated.

8.3.1 Demonstration of Power Savings

Figure 8.9 shows the energy savings for a 54, -13, and 7 degree rotation in roll, pitch, and yaw, respectively. The energy savings are calculated in real time by comparing the power required to achieve the minimum norm solution discussed in Section 6.1 with the

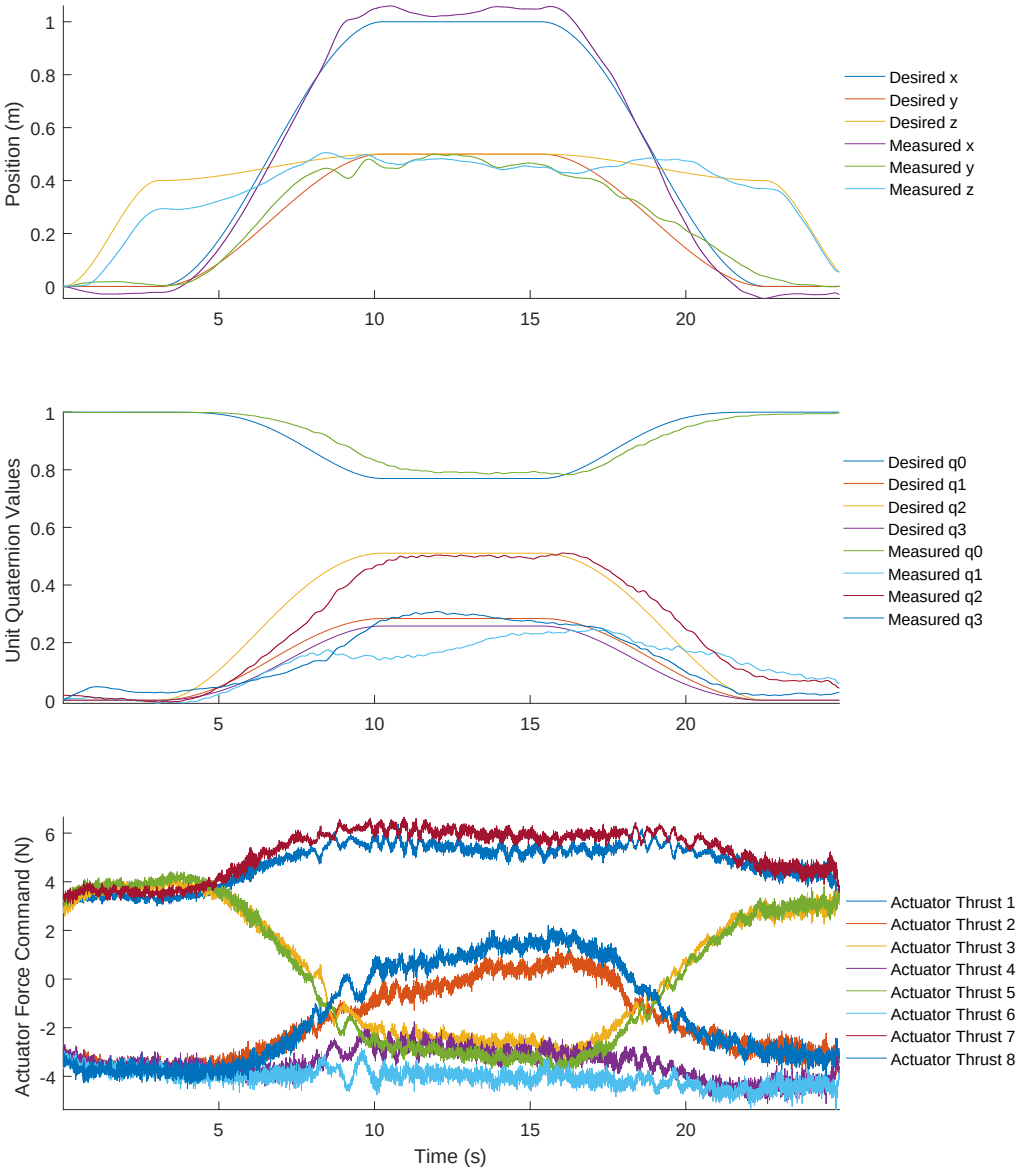

Figure 8.8: Arbitrary translation and attiude commands demonstrating full actuation of the vehicle

power required to produce the energy optimal solution. The power output is displayed as a function of time in the first graph of Figure 8.9.

It may be noted that as the omnicopter takes off with zero roll/pitch/yaw, there is an insignificant difference between the minimum norm and energy optimal solutions. The power outputs between the two solutions begin to diverge as the omnicopter follows its attitude trajectory, reaching a maximum power saving of about 6% before converging again as the omnicopter resumes a landing orientation.

This power output can be integrated over time to calculate the energy consumed. The percentage energy saved is shown in the second graph of Figure 8.9 where savings increase as the omnicopter assumes the attitude hold orientation.

8.3.2 Circular Trajectory with a Disabled Actuator

The circular trajectory tested in Section 8.2.3 is now performed again, except this time running the optimization routine. Furthermore, actuator one is turned off to simulate a failure. This is accomplished by setting both sides of this actuator's limits to zero in the formulation of the optimization problem. It can be seen from the xy displacement plot of Figure 8.10 that the system performs comparable to the original circular trajectory, even with a disabled actuator. The commanded motor thrusts for the trajectory are compared between the two plots of Figure 8.11. The first figure shows the commands requested in the event of a disabled actuator one, while the second shows the original motor commands using the pseudoinverse of the Jacobian as discussed in Section 6.1.

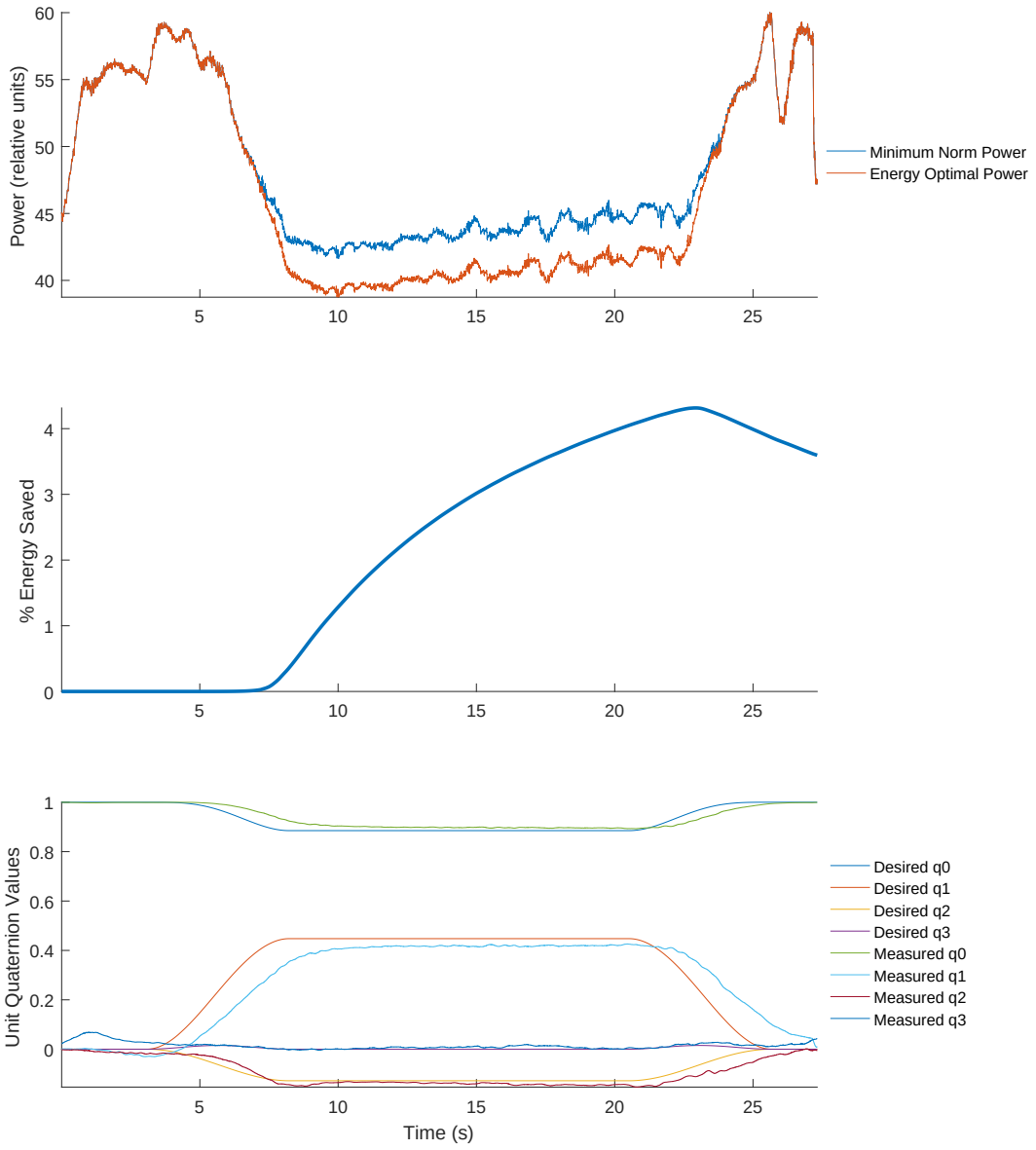

Figure 8.9: Power comparison between minimum norm and energy optimal algorithms to perform an attitude trajectory

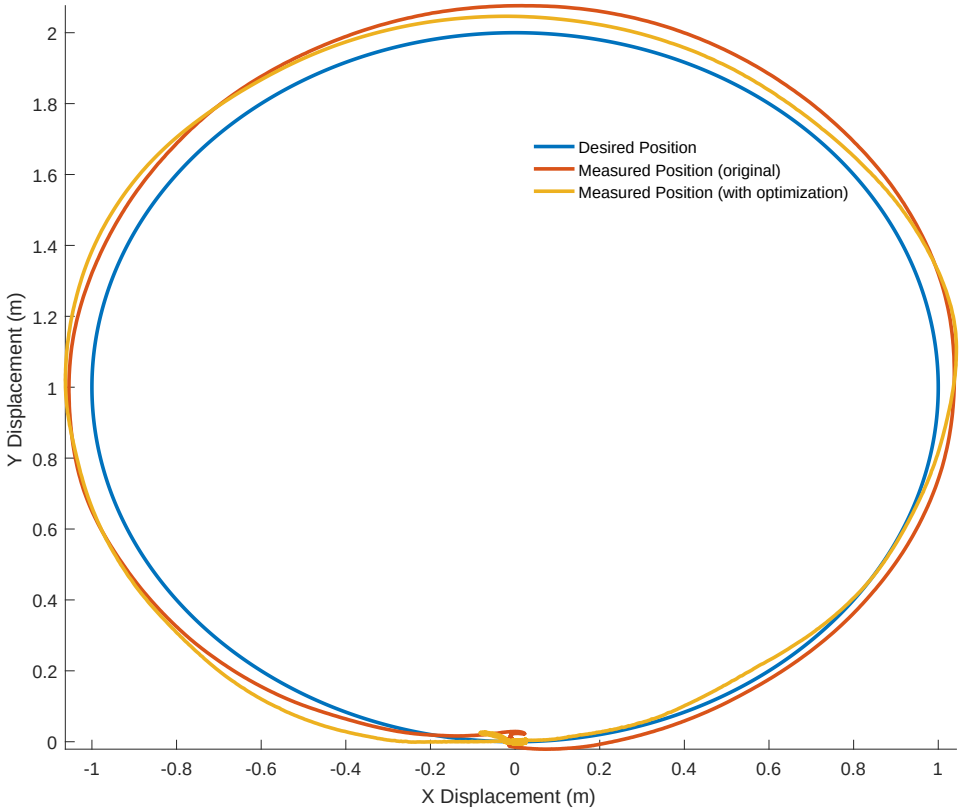

Figure 8.10: Position tracking comparison between original implementation and optimization with a deactivated actuator

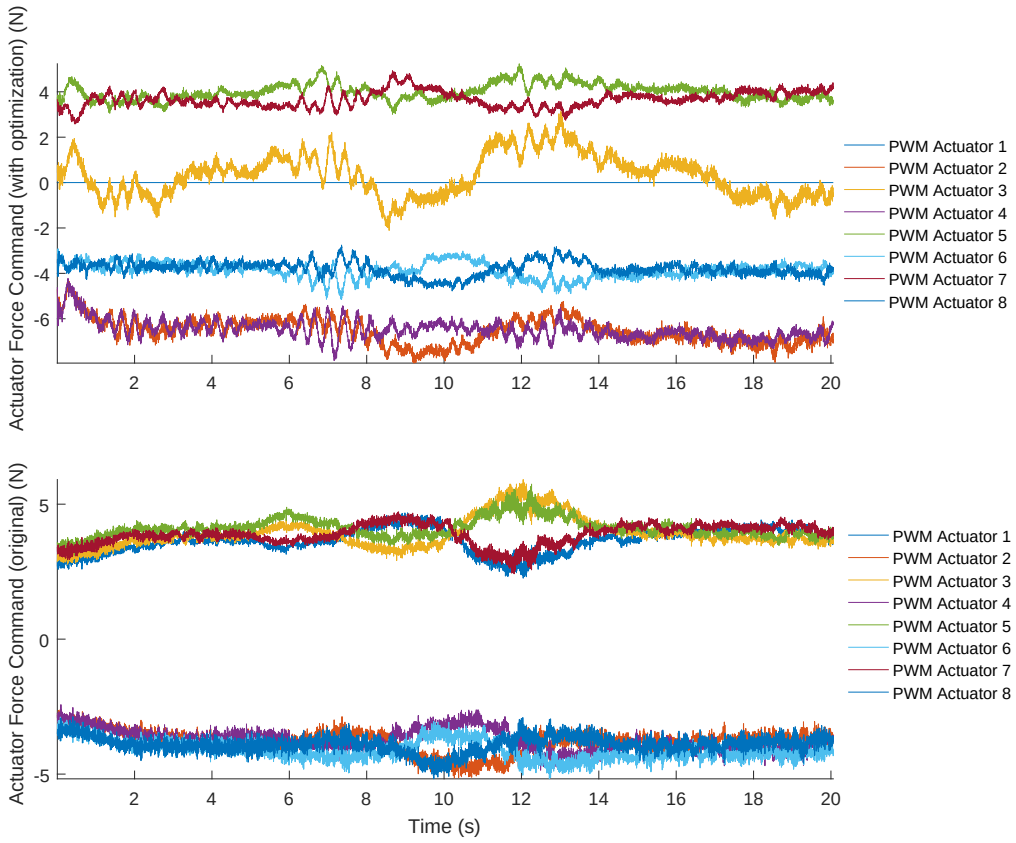

Figure 8.11: Comparison of motor commands between implementation with a disabled actuator and original

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis presented the design and implementation of a fully actuated aerial vehicle capable of following an arbitrary trajectory both in position and attitude. Inspiration for such a vehicle came from the challenges associated with the underactuation of conventional multi-rotors. Since most of these vehicles have all propellers lying in the same plane, they sacrifice lateral actuation to gain effective cancellation of the gravity force. For sensing applications where the drone does not need to physically interact with its environment, this configuration produces an efficient vehicle capable of following limited, but sufficient, trajectories. However, multi-rotors capable of direct interaction with their environment promise powerful applications for the future of this technology. Using conventional underactuated multi-rotor drones in interactive applications can require complex control strategies that are computationally expensive and cannot always guarantee stability as the vehicle may face scenarios where it is unable to provide the necessary force-torque.

After reviewing the current literature in Chapter 2 on fully actuated aerial vehicles,

Chapter 3 sought to find a propeller configuration that would provide an even distribution of actuation over the entire force torque space. For symmetricity, the vehicle frame was set as a cube with eight actuators that are evenly distributed on eight arms that connect the vertices of the cube to its center. In the determination of an optimal propeller configuration, each actuator was given one degree of freedom to rotate about the arm to which it was attached. This problem was formulated as a nonlinear non-convex optimization that seeks to find a configuration that maximizes the amount of force that can be produced in the direction of least actuation. Ultimately this chapter used a novel optimization formulation to provide a detailed justification for using the configuration proposed in [5]. Sincere gratitude is extended to Mohammad Jafarinasab for the role he played to help develop the optimization formulation for the system configuration.

A novel static model was developed in Chapter 4 to map a desired actuator thrust to a PWM command. First, a model for the brushless DC motors used on the omnicopter was presented and identified. This model was then extended to include disturbance effects generated by other actuator airflows. In this way, the PWM command necessary for an actuator to produce a desired thrust was dependent on the battery voltage and disturbing actuators. A generic mesh of grid points was used to verify that a linear model for the γ parameters provided an acceptable approximation of the relationship. A linear model was proposed and fitted through a system identification and verified using a second set of data. This was accomplished using a test bed where the omnicopter was mounted on a 6 DOF force-torque sensor, which measured system responses to inputs generated from a real-time Simulink platform. The test bed was also used to verify the model empirically. A

desired force-torque was sent to the omnicopter and compared against the measured force-torque from the force sensor. Although the error between measured and desired force-torque varied greatly between different commands, it remained within the compensation abilities of a controller.

The full actuation of the system allowed for the decoupling of the position and attitude controllers. This simplified the motion control problem and leads to the development of two independent PID controllers for position and attitude tracking in Chapter 5. The position controller included a gravity compensation feed-forward term to reduce altitude error in position tracking, while the attitude controller employed a cascaded architecture where the outer loop used attitudinal error to generate a desired angular velocity, which was followed by the inner loop controller. A detailed verification of the controllers was conducted using the SimMechanics toolbox to accurately model the dynamics of the omnicopter . Parameters such as sensor delay times and loop rates were also included to provide a realistic test bed to evaluate the controller performance. These simulations demonstrated that simple P/PID architectures can provide robust trajectory tracking performance.

Besides simplifying the motion control problem, the omnicopter offers two degrees of actuator redundancy meaning that a potentially infinite number of viable actuator thrust combinations exist to achieve a desired force-torque. The solution proposed in [5] uses the pseudoinverse of the Jacobian to provide a closed-form solution to the control allocation problem. However this solution does not utilize the null-space of the Jacobian and therefore may provide an unrealizable solution to desired force-torque that lies within the attainable set. The novel control allocation approach proposed in Chapter 6 was formulated as a convex optimization to minimize the omnicopter 's energy consumption subject to the propeller thrust limits. Experimental results demonstrated that this new formulation can

bring energy savings of up to 6%. In addition, the solver provides the system with fault tolerance. In a scenario where an actuator fails, the omnicopter can maintain full actuation and safely land.

Both the model and optimal solver were executed in soft real-time to fly the omnicopter in an experimental test bed. This thesis demonstrated the viability of a fully-actuated aerial vehicle capable of following any arbitrary trajectory using simple control methods that are both robust and can be executed in real time.

9.2 Future Work

As the cost to build small aerial vehicles continues to be driven down, these robots are becoming competitive replacements across a broad envelope of applications. Enhancing this technology from sensing use cases to interaction with the external environment promises to open new opportunities for research and development. This work explored the possibilities of a fully actuated aerial vehicle and demonstrated that such a system can achieve stable flight and follow arbitrary trajectories. However, this vehicle provides many avenues of inquiry that may be explored in the future.

9.2.1 Modeling Improvements

One avenue requiring further investigation involves constructing a more accurate actuator model of the omnicopter . In this thesis, a simplified static model was constructed to capture effects among interacting airflows. This model was shown to work well in real flight cases. However, test bed results indicate that this model produces net force/torques that deviate from the desired by up to 30% in many cases, while worst case scenarios can see deviation

of a specific force component by up to 100%. The controller was robust to these significant modeling errors allowing for good tracking of the trajectories shown in Section 7. However, improvements in the model would further reduce trajectory error allow for the tracking of more aggressive maneuvers.

As discussed in Section 4.3, the experimental test bed only indirectly measures the thrust produced by each actuator. This was accomplished by multiplying the net force/torque measured by the force sensor and multiplying it with the pseudo-inverse of the Jacobian as shown in Equation 4.20. Unfortunately, this indirect measurement introduces a source of error as the derived and true Jacobian transforms are not exactly the same. A sensible upgrade to the test bed would include installing a force sensor underneath each actuator to directly measure thrust.

In future work, a more accurate model could be re-derived from a true fluid simulation of the omnicopter using a software such as Comsol. Care should be take to ensure such a model is not too computationally taxing as it must be implemented in real time on a low power flight computer. However, this new model may include nonlinear coupling terms between actuators that contribute to better model performance.

9.2.2 New Propeller Configurations

The work in this thesis explored a propeller configuration that achieves one of the most even distributions of force/torque possible using an intuitive, semi-symmetric design. Although this allows the omnicopter to assume any attitudinal configuration without significant loss in actuation, the configuration suffers from reduced efficiency compared to conventional designs since components of some actuator thrusts cancel with others. Furthermore, since

canceling the gravity vector generally accounts for the majority of the vehicle's power consumption, conventional configurations orient all propellers in the same plane to provide greatest actuation along this direction. Since the omnicopter's configuration is not optimized to produce force along any particular direction, it suffers from shorter flight times than an equivalent standard quad-rotor or octo-rotor.

This thesis did not take efficiency into account but rather explored what is possible in the realm of arbitrary trajectory tracking. For future applications, the optimization of the propeller configuration should be reformulated to include costs that maximize flight time at the expense of a more uneven distribution of actuation. Such configurations should be able to counteract acceleration due to gravity with increased efficiency while still providing actuation along the lateral plane of the vehicle. In many applications, it is unnecessary for the drone to track an arbitrary position and attitude command. However, there may be great benefits to still having a fully actuated drone, where actuation in the lateral xy body plane can be significantly less than along the body z vector. This could prove especially useful for systems with robotic manipulators as it would allow them to provide reaction forces when interaction with objects without the need to tilt.

Appendix A

Appendix

A.1 Derivation of the Generalized Jacobian

The Jacobian, J, uses the geometry of the omnicopter along with the actuator configuration $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_8]$ (discussed in Section 3.1.2) to provide a mapping between the individual actuators thrusts and their resultant net force torque. The net force-torque vector can be expanded to individual force and torque components about the omnicopter's x, y, and z axes as

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{F_b} \\ \boldsymbol{\tau_b} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ T_x \\ T_y \\ T_z \end{bmatrix} \tag{A.1}$$

The general mapping presented in Equation 2.1, can be rewritten specifically for the omnicopter as

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ T_x \\ T_y \\ T_z \end{bmatrix} = J \begin{bmatrix} f_1 \\ \vdots \\ f_8 \end{bmatrix} \tag{A.2}$$

where f_i is the force produced by actuator U_i. This implies that the Jacobian should be a 6×8 matrix. Further inspection reveals that it can be broken into two parts

$$J = \begin{bmatrix} J_F \\ J_\tau \end{bmatrix} \tag{A.3}$$

such that

$$\boldsymbol{F_b} = J_F * \mathbf{f} \tag{A.4}$$

$$\boldsymbol{\tau_b} = J_\tau * \mathbf{f} \tag{A.5}$$

where $\boldsymbol{F_b}$ and $\boldsymbol{\tau_b}$ are the respective force and torque produced in the omnicopter body frame. Equations A.4 and A.5 demonstrate the Jacobian can be split into the top three rows, J_F and bottom three rows J_τ, which respectively map the input actuator space into output force and torque. The mapping from input space to output force is derived first.

Refering to Section 3.1.2 for the definitions of each actuator frame, it can be noted that each frame, U_i is given freedom to rotate about its x axis by some θ_i. Each actuator frame can be expressed in the omnicopter body frame, B_0 through inspection of the omnicopter's geometry. For example, the first actuator, with 0 degrees of rotation about x, would be written as

$$U_1^{B_0} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \end{bmatrix} \tag{A.6}$$

Now the direction of thrust from the actuator should be expressed in terms of an arbitrary rotation, θ_1 about the actuator's x axis. This is accomplished by a post multiplication to rotate about the current x axis and is further multiplied by a unit y vector. This second multiplication selects the y axis out of the actuator frame as this is the direction of positive thrust that will be used to construct the Jacobian J_F.

$$\boldsymbol{U_{1y}} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}\sin\theta_1 - \frac{1}{\sqrt{6}}\cos\theta_1 \\ -\frac{1}{\sqrt{2}}\sin\theta_1 - \frac{1}{\sqrt{6}}\cos\theta_1 \\ \sqrt{\frac{2}{3}}\cos\theta_1 \end{bmatrix} \tag{A.7}$$

The same can be done for the other seven actuators to produce the final result

$$J_F = \begin{bmatrix} \boldsymbol{U_{1y}} & \boldsymbol{U_{2y}} & \boldsymbol{U_{3y}} & \boldsymbol{U_{4y}} & \boldsymbol{U_{5y}} & \boldsymbol{U_{6y}} & \boldsymbol{U_{7y}} & \boldsymbol{U_{8y}} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\sin(t1)}{\sqrt{2}} - \frac{\cos\theta_1}{\sqrt{6}} & -\frac{\sin\theta_1}{\sqrt{2}} - \frac{\cos\theta_1}{\sqrt{6}} & \sqrt{\frac{2}{3}}\cos\theta_1 \\ \frac{\sin(\theta_2)}{\sqrt{2}} + \frac{\cos\theta_2}{\sqrt{6}} & \frac{\sin\theta_2}{\sqrt{2}} - \frac{\cos\theta_2}{\sqrt{6}} & \sqrt{\frac{2}{3}}\cos\theta_2 \\ \frac{\cos\theta_3}{\sqrt{6}} - \frac{\sin\theta_3}{\sqrt{2}} & \frac{\sin\theta_3}{\sqrt{2}} + \frac{\cos\theta_3}{\sqrt{6}} & \sqrt{\frac{2}{3}}\cos\theta_3 \\ -\frac{\sin\theta_4}{\sqrt{2}} - \frac{\cos\theta_4}{\sqrt{6}} & \frac{\cos\theta_4}{\sqrt{6}} - \frac{\sin\theta_4}{\sqrt{2}} & \sqrt{\frac{2}{3}}\cos\theta_4 \\ -\frac{\sin\theta_5}{\sqrt{2}} - \frac{\cos\theta_5}{\sqrt{6}} & \frac{\sin\theta_5}{\sqrt{2}} - \frac{\cos\theta_5}{\sqrt{6}} & -\sqrt{\frac{2}{3}}\cos\theta_5 \\ \frac{\cos\theta_6}{\sqrt{6}} - \frac{sin\theta_6}{\sqrt{2}} & -\frac{\sin\theta_6}{\sqrt{2}} - \frac{\cos\theta_6}{\sqrt{6}} & -\sqrt{\frac{2}{3}}\cos\theta_6 \\ \frac{sin\theta_7}{\sqrt{2}} + \frac{\cos\theta_7}{\sqrt{6}} & \frac{\cos\theta_7}{\sqrt{6}} - \frac{\sin\theta_7}{\sqrt{2}} & -\sqrt{\frac{2}{3}}\cos\theta_7 \\ \frac{\sin\theta_8}{\sqrt{2}} - \frac{\cos\theta_8}{\sqrt{6}} & \frac{\sin\theta_8}{\sqrt{2}} + \frac{\cos\theta_8}{\sqrt{6}} & -\sqrt{\frac{2}{3}}\cos\theta_8 \end{bmatrix}^T \tag{A.8}$$

The torque mapping matrix J_T consists of two components. The first, $J_{T_{thrust}}$, results from the cross product between an actuator's thrust vector with its moment arm, while the second, $J_{T_{pure}}$ is the pure torque produced by drag forces on the propeller.

$$J_\tau = J_{\tau_{thrust}} + J_{\tau_{pure}} \tag{A.9}$$

To find $J_{\tau_{thrust}}$, the positions of each motor must be defined as a column in P to find each actuator's moment arm on the center of mass. Each element of the column represents the x, y, and z components of the moment arm respectively.

$$P = \frac{r}{\sqrt{3}} \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} \tag{A.10}$$

The cross product is taken between each column in P and each column in J_F from A.8 to produce a corresponding column in $J_{\tau_{thrust}}$. Note that r is a constant representing the distance between the center of geometry of an actuator and the center of geometry of the omnicopter .

The derivation of $J_{\tau_{pure}}$ follows from Equation 2.3 and must take into account the direction each propeller spins to produce a positive thrust.

$$J_{\tau_{pure}} = k * \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} \circ J_F \tag{A.11}$$

The $\circ$ operator represents the Hadamard product [54], which is defined by

$$(A \circ B)_{i,j} = A_{i,j} B_{i,j} \tag{A.12}$$

This is an element-wise operation which is only defined when matrix A and B are of the same dimension. Note that k is the proportionality constant relating actuator thrust to drag torque in steady state as described from Equation 2.3. This value is identified in Section 4.1.

The Jacobian submatrix, which maps the input actuator space to output torque can then

be written as

$$J_\tau = \begin{bmatrix} \frac{(-\sqrt{3}k+3r)\cos\theta_1+(3k+\sqrt{3}r)\sin\theta_1}{3\sqrt{2}} & -\frac{(\sqrt{3}k+3r)\cos t+(3k-\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_1 - r\sin\theta_1) \\ \frac{(\sqrt{3}k+3r)\cos\theta_2+(3k-\sqrt{3}r)\sin\theta_2}{3\sqrt{2}} & \frac{(-\sqrt{3}k+3r)\cos t+(3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_2 - r\sin\theta_2) \\ \frac{(\sqrt{3}k-3r)\cos\theta_3-(3k+\sqrt{3}r)\sin\theta_3}{3\sqrt{2}} & \frac{(\sqrt{3}k+3r)\cos t+(3k-\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_3 - r\sin\theta_3) \\ -\frac{(\sqrt{3}k+3r)\cos\theta_4+(3k-\sqrt{3}r)\sin\theta_4}{3\sqrt{2}} & \frac{(\sqrt{3}k-3r)\cos t-(3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_4 - r\sin\theta_4) \\ \frac{(\sqrt{3}k-3r)\cos\theta_5+(3k+\sqrt{3}r)\sin\theta_5}{3\sqrt{2}} & \frac{(\sqrt{3}k+3r)\cos t+(-3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_5 - r\sin\theta_5) \\ -\frac{(\sqrt{3}k+3r)\cos\theta_6+(-3k+\sqrt{3}r)\sin\theta_6}{3\sqrt{2}} & \frac{(\sqrt{3}k-3r)\cos t+(3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_6 - r\sin\theta_6) \\ -\frac{(\sqrt{3}k-3r)\cos\theta_7+(3k+\sqrt{3}r)\sin\theta_7}{3\sqrt{2}} & -\frac{(\sqrt{3}k+3r)\cos t+(-3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_7 - r\sin\theta_7) \\ \frac{(\sqrt{3}k+3r)\cos\theta_8+(-3k+\sqrt{3}r)\sin\theta_8}{3\sqrt{2}} & -\frac{(\sqrt{3}k-3r)\cos t+(3k+\sqrt{3}r)\sin t}{3\sqrt{2}} & \sqrt{\frac{2}{3}}(k\cos\theta_8 - r\sin\theta_8) \end{bmatrix}^T \tag{A.13}$$

From Section 3.1.2, a specific propeller configuration is chosen using a two level optimization and is presented in Equation 3.5. Substituting Equation A.8 and A.13 back into A.3 to form the complete general Jacobian, then substituting in Equation 3.5 yields the Jacobian for the chosen configuration.

$$J = \begin{bmatrix} -a & b & c & -ak - \frac{\sqrt{3}r(b-c)}{3} & bk - \frac{\sqrt{3}r(a+c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ b & a & -c & bk - \frac{\sqrt{3}r(a+c)}{3} & ak + \frac{\sqrt{3}r(b-c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ a & -b & c & ak + \frac{\sqrt{3}r(b-c)}{3} & -bk + \frac{\sqrt{3}r(a+c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ -b & -a & -c & -bk + \frac{\sqrt{3}r(a+c)}{3} & -ak - \frac{\sqrt{3}r(b-c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ a & -b & c & -ak - \frac{\sqrt{3}r(b-c)}{3} & bk - \frac{\sqrt{3}r(a+c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ -b & -a & -c & bk - \frac{\sqrt{3}r(a+c)}{3} & ak + \frac{\sqrt{3}r(b-c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \\ -a & b & c & ak + \frac{\sqrt{3}r(b-c)}{3} & -bk + \frac{\sqrt{3}r(a+c)}{3} & -ck - \frac{\sqrt{3}r(a+b)}{3} \\ b & a & -c & -bk + \frac{\sqrt{3}r(a+c)}{3} & -ak - \frac{\sqrt{3}r(b-c)}{3} & ck + \frac{\sqrt{3}r(a+b)}{3} \end{bmatrix}^T \tag{A.14}$$

where

$$a = 1/2 + 1/\sqrt{12}$$

$$b = 1/2 - 1/\sqrt{12}$$

$$c = 1/\sqrt{3}$$

Bibliography

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, p. 460, 2015.

[2] A. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano, "Control of an aerial robot with multi-link arm for assembly tasks," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4916–4921.

[3] G. Heredia, A. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. Acosta, and A. Ollero, "Control of a multirotor outdoor aerial manipulator," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3417–3422.

[4] "Industrial cleaning drone," 2018. [Online]. Available: https://www.aerones.com/eng/drones/cleaning_drone/

[5] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3261–3266.

[6] T. M. Cowan, "Organizing the properties of impossible figures," *Perception*, vol. 6, no. 1, pp. 41–56, 1977.

[7] M. Jafarinasab, "Motion control of aerial robotic manipulators," Sept 2017.

[8] M. Verhelst, D. Sylvester, M. Takamiya, M. Clinton, K. Wilcox, and K. Nose, "F4: Building the internet of everything (ioe): Low-power techniques at the circuit and system levels," in *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*. IEEE, 2015, pp. 1–2.

[9] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The stanford testbed of autonomous rotorcraft for multi agent control (starmac)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2. IEEE, 2004, pp. 12–E.

[10] R. Buechi, *Fascination quadrocopter*. Norderstedt Books on Demand, 2011.

[11] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Intuitive 3d maps for mav terrain exploration and obstacle avoidance," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 473–493, 2011.

[12] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft uavs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 807–814.

[13] J. Paneque-Gálvez, M. K. McCall, B. M. Napoletano, S. A. Wich, and L. P. Koh, "Small drones for community-based forest monitoring: An assessment of their feasibility and potential in tropical areas," *Forests*, vol. 5, no. 6, pp. 1481–1507, 2014.

[14] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for

indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[15] P. Doherty and P. Rudol, "A uav search and rescue scenario with human body detection and geolocalization," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2007, pp. 1–13.

[16] M. Swan, "Uae students design drone to dissipate fog at airports," Jan 2015. [Online]. Available: https://www.thenational.ae/uae/uae-students-design-drone-to-dissipate-fog-at-airports-1.119528

[17] P. Pounds, R. Mahony, P. Hynes, and J. M. Roberts, "Design of a four-rotor aerial robot," in *Proceedings of the 2002 Australasian Conference on Robotics and Automation (ACRA 2002)*. Australian Robotics & Automation Association, 2002, pp. 145–150.

[18] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "Development of a quadrotor test bed—modelling, parameter identification, controller design and trajectory generation," *International Journal of Advanced Robotic Systems*, vol. 12, no. 2, p. 7, 2015.

[19] H. Liu, Y. Bai, G. Lu, and Y. Zhong, "Robust attitude control of uncertain quadrotors," *IET Control Theory & Applications*, vol. 7, no. 11, pp. 1583–1589, 2013.

[20] J. P. How, B. BEHIHKE, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE control systems*, vol. 28, no. 2, pp. 51–64, 2008.

[21] W. Wang, H. Ma, and C.-Y. Sun, "Control system design for multi-rotor mav," *Journal of Theoretical and Applied Mechanics*, vol. 51, no. 4, pp. 1027–1038, 2013.

[22] Z. Zhang and M. Cong, "Controlling quadrotors based on linear quadratic regulator," *Applied Science and Technology*, vol. 5, pp. 38–42, 2011.

[23] V. Utkin, J. Guldner, and J. Shi, *Sliding mode control in electro-mechanical systems*. CRC press, 2009.

[24] L. L. Tuan and S. Won, "Pid based sliding mode controller design for the micro quadrotor," in *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*. IEEE, 2013, pp. 1860–1865.

[25] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[26] W.-H. Zhu, Y.-G. Xi, Z.-J. Zhang, Z. Bien, and J. De Schutter, "Virtual decomposition based control for generalized high dimensional robotic systems with complicated structure," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 411–436, 1997.

[27] W.-H. Zhu, *Virtual decomposition control: toward hyper degrees of freedom robots*. Springer Science & Business Media, 2010, vol. 60.

[28] D. Mellinger, *Trajectory generation and control for quadrotors*. University of Pennsylvania, 2012.

[29] Z. Zuo, "Trajectory tracking control design with command-filtered compensation for a quadrotor," *IET control theory & applications*, vol. 4, no. 11, pp. 2343–2355, 2010.

[30] X. Xiang, C. Yu, Q. Zhang, and G. Xu, "Path-following control of an auv: Fully

actuated versus under-actuated configuration," *Marine Technology Society Journal*, vol. 50, no. 1, pp. 34–47, 2016.

[31] B. Convens, K. Merckaert, M. M. Nicotra, R. Naldi, and E. Garone, "Control of fully actuated unmanned aerial vehicles with actuator saturation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 715–12 720, 2017.

[32] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4006–4013.

[33] M. Odelga, P. Stegagno, and H. H. Bülthoff, "A fully actuated quadrotor uav with a propeller tilting mechanism: Modeling and control," in *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. IEEE, 2016, pp. 306–311.

[34] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi, "6d physical interaction with a fully actuated aerial robot," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5190–5195.

[35] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski, "Voliro: An omnidirectional hexacopter with tiltable rotors," *arXiv preprint arXiv:1801.04581*, 2018.

[36] Y. Long and D. J. Cappelleri, "Linear control design, allocation, and implementation for the omnicopter mav," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 289–294.

[37] J. Paulos, B. Caraher, and M. Yim, "Emulating a fully actuated aerial vehicle using two actuators."

[38] M. Mboup, C. Join, and M. Fliess, "Numerical differentiation with annihilators in noisy environment," *Numerical algorithms*, vol. 50, no. 4, pp. 439–467, 2009.

[39] W. Ding, J. Wang, S. Han, A. Almagbile, M. A. Garratt, A. Lambert, and J. J. Wang, "Adding optical flow into the gps/ins integration for uav navigation," in *Proc. of International Global Navigation Satellite Systems Society Symposium*. Citeseer, 2009, pp. 1–13.

[40] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a uav," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3309–3316.

[41] Z. He, R. V. Iyer, and P. R. Chandler, "Vision-based uav flight control and obstacle avoidance," in *American Control Conference, 2006*. IEEE, 2006, pp. 5–pp.

[42] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, 2006.

[43] D. Gebre-Egziabher, R. C. Hayward, and J. D. Powell, "Design of multi-sensor attitude determination systems," *IEEE Transactions on aerospace and electronic systems*, vol. 40, no. 2, pp. 627–649, 2004.

[44] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors,"

in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2001, pp. 2003–2011.

[45] S. W. Shepperd, "Quaternion from rotation matrix.[four-parameter representation of coordinate transformation matrix]," 1978.

[46] M. H. M. Ramli, "Multirotors system," *Journal of Malaysian Citation Centre*, vol. 6, no. 1, pp. 6–6, 2018.

[47] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay [fir/all pass filters design]," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, 1996.

[48] L. Zaccarian, "Dc motors: dynamic model and control techniques," *Lecture Notes., Roma, Italy*, 2012.

[49] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, 2011.

[50] M. Prabha, R. Thottungal, and S. Kaliappan, "Modeling and simulation of x-quadcopter control," *International Journal for Research in Applied Science & Engineering Technology (IJRASET).[online] Available at: http://www. ijraset. com/fileserve. php*, 2016.

[51] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[52] "Multiwii serial protocol," http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol.

[53] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[54] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.