Using Model Reference Adaptive Control to Mitigate Ground Effect On A Mini Quadcopter

By

Sui Nam Chan

Thesis

Submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Mechanical and Aerospace Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Prof. Zhaodan Kong, Chair

Prof. Ron Hess

Prof. Jason Moore

Committee in Charge

2017

In dedication to my lovely wife ...

who has been a great supporter.

Contents

List of Figures

List of Tables

Abstract

Using Model Reference Adaptive Control
to Mitigate Ground Effect On A Mini Quadcopter

In this thesis, a study of ground effects on a quadrotor aerial vehicle was presented. Experimental results showed that thrust generated by the rotors increased linearly as the vehicle got closer to the ground, which was different from the single rotor vehicle's ground effect model used in other research. Furthermore, a quadcopter experienced ground effects sooner than the single rotor model predicted, and the ground effect was weaker when the vehicle was at close proximity to the ground. Then, a control architecture that utilizes a model reference adaptive controller was proposed to mitigate ground effect. Different position controllers were implemented on a physical system, which included a Crazyflie 2.0 and a computer. The controllers used in this study included a PID controller, a model reference adaptive controller (MRAC) with a linear ground effect model, and a MRAC that uses a set of radial basis functions (RBF), plus a bias term to approximate the ground effect function. The quadcopter took off and landed within the ground effect region multiple times, and its performances were compared. From the flight tests, the MRACs outperformed the PID controller. Among the MRACs, the one with RBF tracked the reference model better with less mean square error. It also responded quicker with less rise time. Furthermore, the MRAC with RBF performed more consistently compared to the one using linear ground effect model.

Acknowledgments

I would not be able to finish this thesis without the help from others. First, I would like to express my special thanks and highest appreciation to Professor Zhaodan Kong for his patience, his guidance throughout this project, and his help in getting me a summer internship at Aurora Flight Science. I learned a lot that summer and met my mentor, Navid Dadkhah. I would also like to give my thanks to Professor Ron Hess and Professor Jason Moore for their inputs about the project approach and evaluating my work. Furthermore, I am thankful for Professor C.P. Case van Dam, who provided suggestions about the ground effect experiment. Lastly, I would like to thank Gregory Bales, Gang Chen, and Peng Wei for brainstorming ideas and troubleshooting issues for this work.

Nomenclature

$0_{x\times x}$	A x-by-x zero matrix
$1_{x\times x}$	A x-by-x identity matrix
γ	Crazyflie's motor constant ratio, $\gamma = k_m/k_f$
ω	Body angular rate $[rad/s]$
ϕ	Roll angle [radian]
ψ	Yaw angle [radian]
θ	Pitch angle [radian]
F	Forces acting on the vehicle in inertial frame [N]
I	A 3-by-3 Moment of inertia matrix $[kg - m^2]$
I_{xx}	Crazyflie's moment of inertia along x-axise
I_{yy}	Crazyflie's moment of inertia along y-axis
I_{zz}	Crazyflie's moment of inertia along z-axis
k_ϕ	Attitude loop feedforward gain for roll angle command
k_ψ	Attitude loop feedforward gain for yaw angle command
k_θ	Attitude loop feedforward gain for pitch angle command
k_f	Crazyflie's motor force constant, $F = k_f PWM$
k_m	Crazyflie's motor moment constant, $M = k_m PWM$
k_{thrust}	Attitude loop feedforward gain for throttle
L	Distance from the center of a crazyflie's rotor to the vehicle's center line

M Moments acting on the vehicle in body frame [N-m]

m Crazyflie's mass with markers and frame

m_1 PWM signal for motor 1 $[N]$

m_2 PWM signal for motor 2 $[N]$

m_3 PWM signal for motor 3 $[N]$

m_4 PWM signal for motor 4 $[N]$

M_{ru} Input channels mixing matrix

P Desired thrust [PWM]

u_1 Total thrust from all motors $[N]$

u_{2x} Rolling moment $[N-m]$

u_{2y} Pitching moment $[N-m]$

u_{2z} Yawing moment $[N-m]$

u_{fwd} Outputs from feedforward controller

u_{lqr} Outputs from LQR

v Vehicle's velocities in inertial frame $[m/s^2]$

CoG Center of gravity

MRAC Model reference adaptive controller

MSE Mean squared error

PID A proportionalintegralderivative controller

PWM Pulse-width modulation

RBF A radial basis function

ROS Robot Operating System

UAV Unmanned aerial vehicle

Chapter 1

Introduction

1.1 Motivation

Multirotor aerial vehicles have become increasingly popular over the past decade due to their high maneuverability and vertical takeoff capability. We have seen these vehicles in our daily life, from indoor hobby flying to outdoor professional photography, which are shown in Table 1.1. Recently, there are growing interests in using multirotor vehicles for low-altitude high risk applications. Dubai Police plans to purchase a hoverbike, which was developed by a Russian company called Hoversurf, to use as a patrol car[3]. Alec, from Delft University of Technology, delivers a defibrillator to patients in needed using a tricopter[4]. Researchers from Pontifical Xavierian University put a multispectral camera on a quadcopter for Multispectral mapping in agriculture[5]. These vehicles operate at low altitude, where the efficiency of the rotor system increases due to ground effect[6]. As a result, less power is needed when the vehicle is at close proximity to ground. However, ground effect varies significantly in flight due to several factors, including distance from ground, type of ground, and vehicle's speed. Extreme caution must be taken when operating within ground effect region, otherwise, there is little time for recovery operation due to proximity to ground, and the vehicle can crash to the ground. Therefore, a better understanding about ground effect is needed for multirotor vehicles, and we need a controller that can deal with ground effect in the whole flight profile. If not, accidents can happen that will cause property damage or even result in loss of life. Looking back in history, there

are a lot of accidents due to ground effect, from incidents to fatal accidents[7][8][9][10][11], and we should learn from our mistakes. However, there are shortcomings with current technology.


Table 1.1: Left: mini hexcopter for indoor flying (the coin is a size reference), Right: DJI Phantom 4 with HD camera for photography[2]

1.2 State of the Art

1.2.1 UAV Control

Proportional-integral-derivative (PID) controllers have been widely used in rotorcraft control due to its versatility and facile implementation. PID controllers were used to stabilize a multirotor aerial platform equipped with a overhanging robotic arm[13]. Other researchers used a PID controller to control the motors' speed on a quadcopter with high precision, which resulted in better performace[14][15]. Another study used two proportional-integral (PI) controllers (Euler angle controllers and angular rate controller) to provide attitude control of a quadcopter with external disturbances[17]. At this point, research on multirotor aerial vehicles mainly focuses on simple tasks, such as stabilization when the vehicle is stationary or moving at low speed. As interest in multirotor vehicles grows due to its unique capability, new applications are proposed that require trajectory following and aggressive maneuverability. These requirements give rise to multilayer control architecture and nonlinear control due to the vehicle's nonlinear nature. Researchers utilized the flatness property of multirotor vehicles and implemented a nonlinear feedforward controller in a hexacopter[18], which allowed the vehicle to follow complicated paths, including a figure eight-like trajectory and a circular path. Other researchers proposed a

nonlinear controller called "discrete-time one-step ahead prediction control" to deal with system nonlinearity and actuator constraints[19]. Other researchers implemented a nonlinear backstepping controller with integral action in a multirotor aerial vehicle to stabilize a suspended payload[20]. In experiments, the controller tracked a figure eight-like trajectory well under wind condition. Although these controllers perform really well in designed conditions, they suffer performance loss in the presence of parametric uncertainty. These controllers have constant gains which are designed for certain conditions.

1.2.2 Adaptive Control for UAV

As new applications emerge, plant variation becomes an issue, which brings renewed interest in adaptive control techniques. Adaptive control is able to handle uncertainties, because its gains vary in-flight according to an adaptive law. A quadcopter that used a linear proportional-derivative (PD) cascade controller was destabilized by changing its center of gravity (CoG), which is crucial to package delivery using unmanned aerial vehicles (UAV)[21]. The authors proposed an adaptive tracking controller, which was based on output feedback linearization, to solve the problem and proved that the controller provided great tracking performance with CoG shifted in flight. On the other hand, researchers focused on altitude control for quadcopters, where changes in mass greatly reduced the system performance[22]. The authors implemented a model reference adaptive control (MRAC) based on a simplified model to assist an existing PI heave-velocity stabilizer, and showed that the proposed controller was able to retain its performance when the mass of a quadcopter changed in flight. Another study used a nonlinear H_∞ controller with the assistance of a model predictive controller to solve path following problem. The authors proved that the controller is robust to external disturbance on all six degrees of freedom with structural and parametric uncertainties[23]. The optimal path tracking problem was solved by using linear quadratic tracking (LQT) algorithm, which is similar to LQR but its gains vary in time[24]. It is robust to model uncertainty and external disturbance, but it required matrix inversion, which is computationally heavy.

Figure 1.1: Soviet Ekranoplan, a ground effect vehicle that flies a few meters above the surface[1]

1.2.3 Ground effect for UAV

Ground effect increases the efficiency of the rotor system when the vehicle operates near the ground or other flat surfaces. This is caused by interference of the airflow by the flat surface, which decreases the downward velocity of air and reduces the induced drag[6]. In other word, vehicles that operate within ground effect require less power to hover. However, extra caution is needed when entering or exiting ground effect zone, otherwise, the sudden change in thrust can result in a fatal accident. Due to its potential benefits and dangers, ground effect on aerial vehicles is a widely studied topic. For example, mathematical models of ground effect have been developed for fixed wing aircraft[26] and helicopters[27][28][29][30]. Also, vehicles that take advantage of ground effect have been developed, such as the Soviet Ekranoplan shown in Figure 1.1. However, ground effect research for multirotor vehicles is limited[31][32], and they used the ground effect model for single rotor vehicle from[30], which is shown in Equation 1.1. Although experiments showed that the single rotor model predict the general trend for thrust in ground effect, detail was lacking[31].

$$\frac{F}{F_0} = \frac{1}{1 - \frac{R^2}{16Z^2}} \tag{1.1}$$

where F is the thrust when the vehicle is under ground effect, F_0 is the thrust without the ground effect, R is the radius of the propeller, and Z is the distance from the flat surface.

1.3 Goal

In this paper, I would like to address the issues listed in the previous sections by combining PID, LQR, and MRAC to mitigate ground when a multirotor aerial vehicle approaches the ground, such as takeoff and landing.

1.4 Methods

First, a ground effect model for quadcopters is obtained using experiments. Then, a set of nonlinear dynamic equations for quadcopters are derived using Newton-Euler equations and are linearized at hovering state. Attitude and position controller are designed using linear control method: LQR technique and PID method. A model reference adaptive controller is added to altitude loop to handle ground effect. Lastly, these controllers are implemented to the system, which consists of a mini quadcopter called Crazyflie 2.0, which is developed by Bitcraze, and a computer as a ground station. Flight tests are performed to evaluate the proposed control architecture when dealing with ground effect.

Chapter 2

System Model

In this section, the hardware platform used, Crazyflie 2.0 developed by Bitcraze, is introduced first. Then, the reference frames used in this paper are defined. Lastly, a derivation of the rigid body dynamic model of the Crazyflie UAV is provided using Newton-Euler equations.

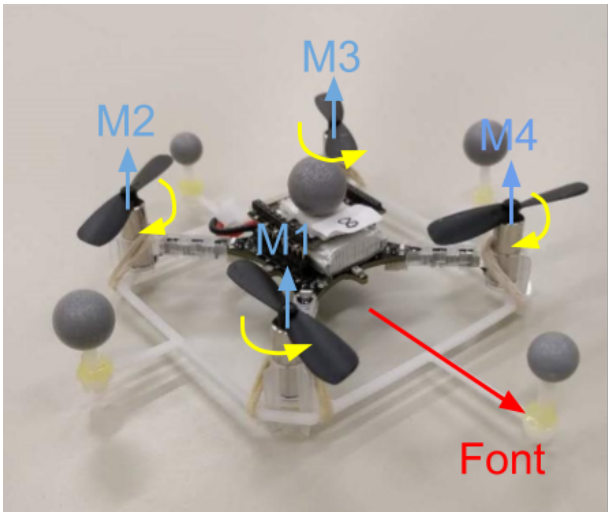

Figure 2.1: A Crazyflie 2.0 with IR reflective markers. M1, M2, M3, and M4 are the names of the motor, and the yellow arrows show the spinning direction for each motor.

2.1 Crazyflie 2.0

The Crazyflie 2.0 from Bitcraze is a mini versatile flying platform, which weighs only 27 grams. Due to its size and open source development kit, it is a great platform to test

new control algorithm in indoor environment. In this paper, a modified Crazyflie is used as shown in Figure 2.1. From the figure, the sliver spheres are reflective markers, which are used by motion capture systems called Optitrack, to track object's local position and orientation. The reflective markers are attached to a Crazyflie through a specially designed frame.

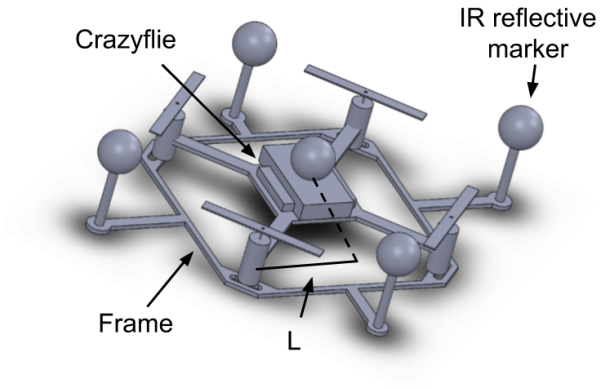

Figure 2.2: A 3D model of the modified Crazyflie, which includes IR reflective markers for motion capture system to track the vehicle. L is the distance from rotor to center line

2.1.1 Parameters

To determine the system mass and inertia, a 3D model was constructed using SolidWorks (a 3D CAD software). First, the system was taken apart and the weights of each part were recorded. Then, each piece was modeled as simple shapes (like a block or sphere) with the recorded mass in CAD software. Lastly, the pieces were assembled together to create a completed 3D model of the modified Crazyflie, which is shown in Figure 2.2. Using SolidWork's toolbox, mass and moment of inertia were calculated, and they are listed in Table 2.1.

Motor constants, k_f and k_m, define how much thrust/moment a motor generates for a given PWM signal. γ and k_m are obtained from [33], while an experiment is performed to find k_f. To determine k_f, a Crazyflie is placed inversely on top of a scale, as shown in Figure 2.3. Then, the throttle is increased from 0% to 100% in an increment of 10%. The corresponding motor signal (pulse-width modulation signal, PWM) and thrust are recorded, which is shown in Figure 2.4. Then, a linear model is fitted to the data to relate

Table 2.1: Crazyflie 2.0 Parameters

Parameters	Description	Value
m	Mass with markers and frame	0.3812 kg
L	Distance from rotor to center line	0.035 m
I_{xx}	Moment of inertia along x-axis	2.661×10^{-5} kg-m^2
I_{yy}	Moment of inertia along y-axis	2.858×10^{-5} kg-m^2
I_{zz}	Moment of inertia along z-axis	5.799×10^{-5} kg-m^2
k_f	Motor force constant, $F = k_f \times PWM$	2.083×10^{-6} N
k_m	Motor moment constant, $M = k_m \times PWM$	7.707×10^{-9} N-m
γ	Motor constant ratio, $\gamma = k_m/k_f$	3.700×10^{-3} N-m/N

PWM signal to thrust, and the result is shown in Equation (2.1). Lastly, the slope of the line is defined as motor's force constant, k_f.

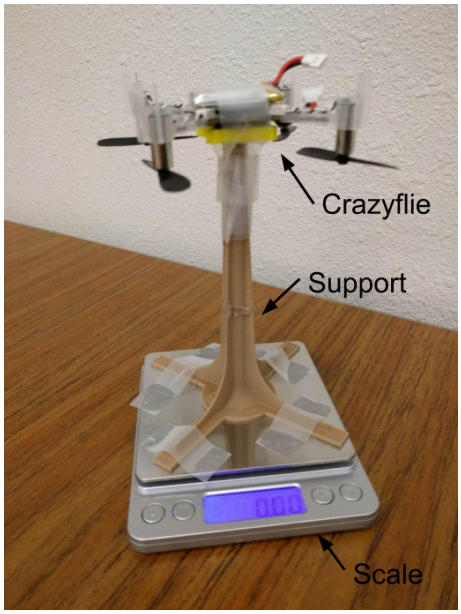

Figure 2.3: A thrust measurement test rig

$$y = 2.083 \times 10^{-6}x - 0.00396 \tag{2.1}$$

where x is PWM signal and y is thrust (N).

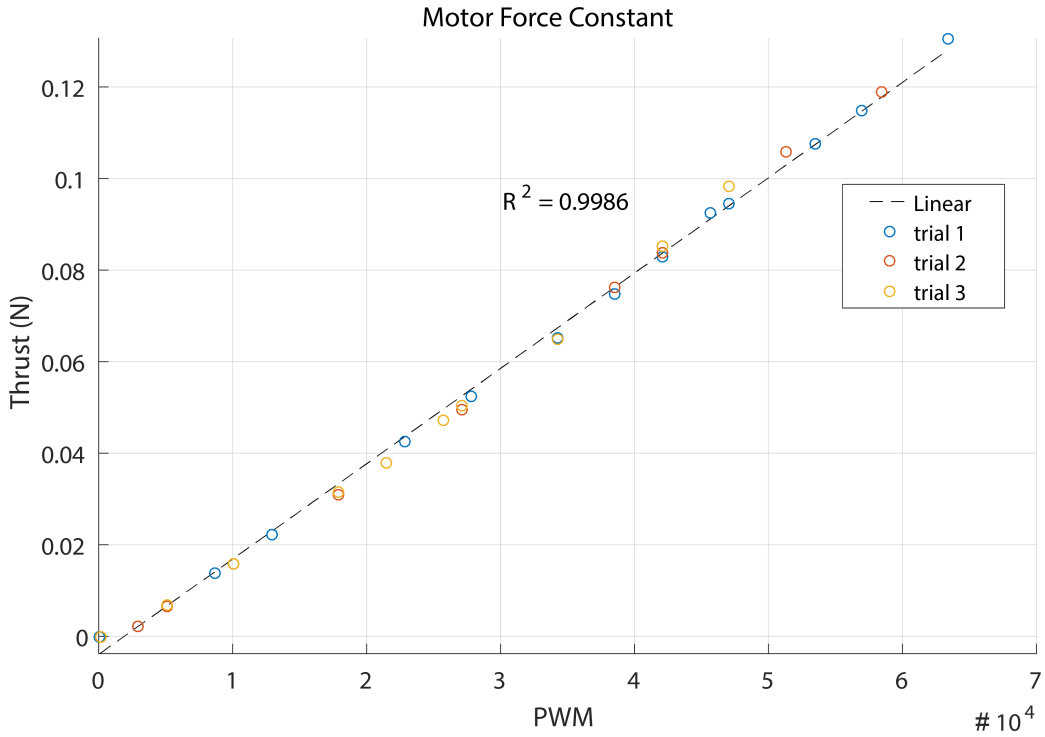

Figure 2.4: Thrust curve

2.1.2 Thrust and moments

Let m_1, m_2, m_3, m_4, represent PWM control signals for motor 1, 2, 3, 4 respectively, and u_1 be the total thrust from all motors $[N]$. u_1 is described in Equation (2.2) using k_f. Noted that, pointing upward is negative, which is discussed in the following section, Reference Frame.

$$u_1 = k_f \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.2}$$

Then, u_{2x}, u_{2y}, u_{2z} are defined as rolling, pitching, yawing moment respectively $[N{-}m]$. Rolling and pitching moments are caused by thrust differences from different motors (front vs back pairs and left vs right side) multiplied by a moment arm (L), while yawing moment is caused by different torque from motors (clockwise vs counter-clockwise). The

relationships are described in Equation (2.3) using k_f and $\gamma = k_m/k_f$.

$$\begin{bmatrix} u_{2x} \\ u_{2y} \\ u_{2z} \end{bmatrix} = k_f \begin{bmatrix} -L & -L & L & L \\ L & -L & -L & L \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.3}$$

2.2 Reference Frame

2.2.1 Inertial Frame

A conventional right hand Cartesian coordinate system is used, where z-axis points downward. The origin of the inertial frame is defined as the center of the laboratory, and the inertial frame is fixed on the ground. As shown in Figure 2.5, the unit vector X_e points to the east wall, Y_e points to the south wall, and Z_e points to the ground.

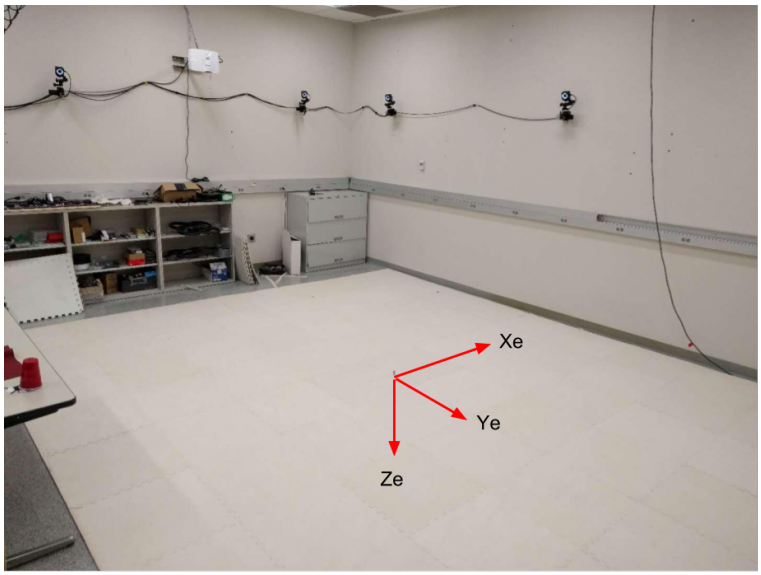

Figure 2.5: Inertial frame

2.2.2 Vehicle Frame

The vehicle frame's origin is located at the center of mass of the quadcopter, while its axes are aligned with the inertial frame.

2.2.3 Body Frame

The body frame has the same origin as the vehicle frame, but its unit vectors aligned with the vehicle's forward direction as shown in Figure 2.6. X_b points to the front of the vehicle, Y_b points to the right, and Z_b points downward.

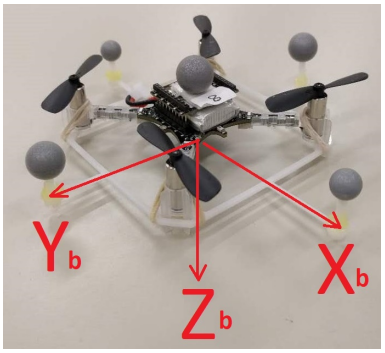

Figure 2.6: Reference Frame

2.2.4 Tait-Bryan Angles

To rotate from vehicle frame to body frame, three rotations are involved, and its angles are commonly known as Tait-Bryan angles: roll (ϕ), pitch (θ), and yaw (ψ).

Vehicle-1 Frame

From vehicle frame, rotate in the positive right-handed direction about Z_v by ψ and get to vehicle-1 frame. The transformation from the vehicle frame to the vehicle-1 frame is given by

$$\boldsymbol{p}^{v1} = R_v^{v1}(\psi)\,\boldsymbol{p}^v$$

where

$$R_v^{v1}(\psi) = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Vehicle-2 Frame

Then, rotate in the positive right-handed direction about Y_1 by θ to vehicle-2 frame, which is described by the following equation.

$$\boldsymbol{p}^{v2} = R_{v1}^{v2}(\theta)\,\boldsymbol{p}^{v1}$$

where

$$R_{v1}^{v2}(\theta) = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix}$$

Body Frame

Lastly, rotate in the positive right-hand direction about X_2 by ϕ to body frame, and the transformation is given by

$$\boldsymbol{p}^b = R_{v2}^b(\phi)\,\boldsymbol{p}^{v2}$$

where

$$R_{v2}^b(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix}$$

By combining the rotation matrics from previous sections in the correct order, the transformation from vehicle frame to body frame is given as follow.

$$R_v^b(\phi,\theta,\psi) = R_{v2}^b(\phi)\,R_{v1}^{v2}(\theta)\,R_v^{v1}(\psi) \tag{2.4}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \tag{2.6}$$

$$\tag{2.7}$$

Similarly, the rotation matrix from body frame to vehicle frame is

$$R_b^v(\phi,\theta,\psi) = R_v^b(\phi,\theta,\psi)^T \tag{2.8}$$

$$= \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2.9}$$

$$\tag{2.10}$$

The relationship between each reference frame can be summarized in Figure 2.7.

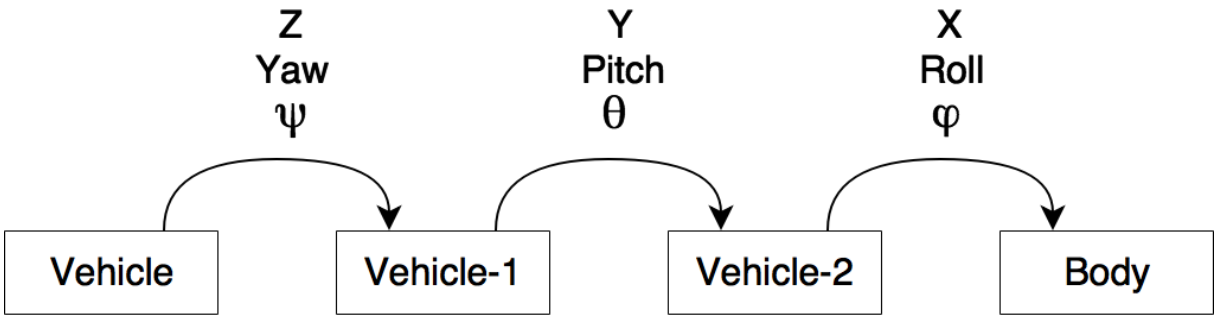

Figure 2.7: The relationship between each reference frames

Tait-Bryan Angular Rates and Body Axis Rates

From Figure 2.7, it is obvious that yaw rate is in the vehicle-1 frame, pitch rate is in the vehicle-2 frame, and roll rate is in body frame. Therefore, the transformation from Tait-Bryan angular rates ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$) to body axis rate, $\boldsymbol{\omega}^b$ is given by the following.

For yaw rate ($\dot{\psi}$), it is in vehicle-1 frame, so it is rotated to vehicle-2 frame first and then body frame. It is described with the equation below.

$$\boldsymbol{\omega}^b = R_{v1}^b(\phi,\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.11}$$

where

$$R_{v1}^b(\phi,\theta) = R_{v2}^b(\phi) R_{v1}^{v2}(\theta)$$

Similarly, pitch rate ($\dot{\theta}$), is in vehicle-2 frame. Thus, a single rotation brings it to body

frame.

$$\boldsymbol{\omega}^b = R^b_{v2}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \tag{2.12}$$

Lastly, a rotation is not needed for roll rate ($\dot{\phi}$) since it is already in the body frame.

$$\boldsymbol{\omega}^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

By combining Equation (2.11), (2.12), and (2.13), the transformation from Tait-Bryan angular rates to body axis rates can be represented as the following.

$$\boldsymbol{\omega}^b = R^b_{\dot{\phi},\dot{\theta},\dot{\psi}} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.14}$$

where

$$R^b_{\dot{\phi},\dot{\theta},\dot{\psi}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + R_{roll} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + R_{roll} R_{pitch} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

$$= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\sin(\theta) \end{bmatrix} \tag{2.16}$$

Similarly, the rotation matrix from body axis rates to Tait-Bryan angular rates is

$$R^{\dot{\phi},\dot{\theta},\dot{\psi}}_b = \left(R^b_{\dot{\phi},\dot{\theta},\dot{\psi}}\right)^T \tag{2.17}$$

$$= \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \tag{2.18}$$

2.3 Equations of Motion

2.3.1 Nonlinear equations

Let F $[N]$ denote the forces acting on the vehicle in Earth frame, m $[kg]$ denote the mass of the vehicle, a $[m/s^2]$ denote the vehicle's acceleration in Earth frame, and M $[N\text{-}m]$ denote the moments acting on the vehicle in body frame. Using Newton-Euler equations, the following equations are obtained:

$$\boldsymbol{F} = m\dot{\boldsymbol{v}} \tag{2.19}$$

$$\boldsymbol{M} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} \tag{2.20}$$

In Equation (2.19), $\boldsymbol{F}$ are the external forces acting on the vehicle, which include gravity, g, in vehicle frame and total motor thrust, u_1, in the body frame. By transforming u_1 from body frame to vehicle frame, the equation becomes

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} R_b^v \begin{bmatrix} 0 \\ 0 \\ u1 \end{bmatrix} \tag{2.21}$$

Combining the equation above with Equation (2.9), it becomes

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ u1 \end{bmatrix} \tag{2.22}$$

$$= \frac{1}{m} \begin{bmatrix} u_1 \left(\cos\left(\phi\right) \sin\left(\theta\right) \cos\left(\psi\right) + \sin\left(\phi\right) \sin\left(\psi\right)\right) \\ u_1 \left(\cos\left(\phi\right) \sin\left(\theta\right) \sin\left(\psi\right) - \sin\left(\phi\right) \cos\left(\psi\right)\right) \\ mg + u_1 \left(\cos\left(\phi\right) \cos\left(\theta\right)\right) \end{bmatrix} \tag{2.23}$$

In Equation (2.20), $\boldsymbol{M}$ represents the external moments acting on the vehicle, which includes rolling moment, u_{2x}, pitching moment, u_{2y}, and yawing moment, u_{2z}. To keep

things simple, a symmetrical body is assumed, such that the cross product terms in moment of inertia, I, are zeros. The following equations are obtained.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}} (I_{yy} - I_{zz}) qr \\ \frac{1}{I_{yy}} (I_{zz} - I_{xx}) pr \\ \frac{1}{I_{zz}} (I_{xx} - I_{yy}) pq \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} u_{2x} \\ u_{2y} \\ u_{2z} \end{bmatrix} \tag{2.24}$$

To obtain equations of motion in terms of motor control signals (PWM), Equation (2.2) and (2.3) are substituted into Equation (2.23) and (2.24) respectively, which results in the following equations.

1) Forces

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} u_1 (\cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi)) \\ u_1 (\cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi)) \\ mg + u_1 (\cos(\phi) \cos(\theta)) \end{bmatrix} \tag{2.25}$$

2) Moments

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}} (I_{yy} - I_{zz}) qr \\ \frac{1}{I_{yy}} (I_{zz} - I_{xx}) pr \\ \frac{1}{I_{zz}} (I_{xx} - I_{yy}) pq \end{bmatrix} + k_f \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} -L & -L & L & L \\ L & -L & -L & L \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.26}$$

$$= \begin{bmatrix} \frac{1}{I_{xx}} (I_{yy} - I_{zz}) qr \\ \frac{1}{I_{yy}} (I_{zz} - I_{xx}) pr \\ \frac{1}{I_{zz}} (I_{xx} - I_{yy}) pq \end{bmatrix} + \begin{bmatrix} \frac{k_f}{I_{xx}} (-Lm_1 - Lm_2 + Lm_3 + Lm_4) \\ \frac{k_f}{I_{yy}} (Lm_1 - Lm_2 - Lm_3 + Lm_4) \\ \frac{k_f}{I_{zz}} (\gamma m_1 - \gamma m_2 + \gamma m_3 - \gamma m_4) \end{bmatrix} \tag{2.27}$$

Equation (2.25) is in terms of angles, ϕ, θ, ψ, while Equation (2.27) is in terms of Body axis angles, p, q, r. Therefore, a third set of equations is needed to relate angles and body axis angles, which is Equation (2.18). A summary of nonlinear equations of motion is shown in Table 2.2.

Table 2.2: Summary: Nonlinear Equation of Motion

Forces:	$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} u_1 \left(\cos\left(\phi\right)\sin\left(\theta\right)\cos\left(\psi\right) + \sin\left(\phi\right)\sin\left(\psi\right)\right) \\ u_1 \left(\cos\left(\phi\right)\sin\left(\theta\right)\sin\left(\psi\right) - \sin\left(\phi\right)\cos\left(\psi\right)\right) \\ mg + u_1 \left(\cos\left(\phi\right)\cos\left(\theta\right)\right) \end{bmatrix}$
Moments:	$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}} \left(I_{yy} - I_{zz}\right) qr \\ \frac{1}{I_{yy}} \left(I_{zz} - I_{xx}\right) pr \\ \frac{1}{I_{zz}} \left(I_{xx} - I_{yy}\right) pq \end{bmatrix} + \begin{bmatrix} \frac{k_f}{I_{xx}} \left(-Lm_1 - Lm_2 + Lm_3 + Lm_4\right) \\ \frac{k_f}{I_{yy}} \left(Lm_1 - Lm_2 - Lm_3 + Lm_4\right) \\ \frac{k_f}{I_{zz}} \left(\gamma m_1 - \gamma m_2 + \gamma m_3 - \gamma m_4\right) \end{bmatrix}$
Rotation:	$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\left(\phi\right)\tan\left(\theta\right) & \cos\left(\phi\right)\tan\left(\theta\right) \\ 0 & \cos\left(\phi\right) & -\sin\left(\phi\right) \\ 0 & \frac{\sin\left(\phi\right)}{\cos\left(\theta\right)} & \frac{\cos\left(\phi\right)}{\cos\left(\theta\right)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$
Kinematic	$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 1_{3\times 3} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$

2.3.2 Linearization

The nonlinear differential equations in Table 2.2 are linearized about hovering state, which results in a linear system of equations with 12 states, $\boldsymbol{x}$, and 4 inputs, $\boldsymbol{u}$.

Consider that a system of nonlinear first order differential equation as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right)$$

Then, the linearized system will be

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$

where

$$A = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x_{trim}}, \boldsymbol{u_{trim}}}$$
$$B = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x_{trim}}, \boldsymbol{u_{trim}}}$$

Using Sympy[34], a symbolic math Python library, the following equations are obtained. Noted that the position and attitude dynamics are decoupled.

$$
\begin{aligned}
&\text{State vector:} && \boldsymbol{x} = \begin{bmatrix} p & q & r & \phi & \theta & \psi \end{bmatrix}^T \\
&\text{Input vector:} && \boldsymbol{u} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \end{bmatrix}^T \\
&\text{Trim States :} && \boldsymbol{x_{trim}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \psi_0 \end{bmatrix}^T \\
&\text{Trim Inputs :} && \boldsymbol{u_{trim}} = \tfrac{mg}{4}\tfrac{1}{k_f}\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T
\end{aligned}
$$

2.3.2.1 Attitude Dynamics

$$
\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ 1_{3\times3} & 0_{3\times3} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} -\frac{Lk_f}{4I_{xx}} & -\frac{Lk_f}{4I_{xx}} & \frac{Lk_f}{4I_{xx}} & \frac{Lk_f}{4I_{xx}} \\ \frac{Lk_f}{4I_{yy}} & -\frac{Lk_f}{4I_{yy}} & -\frac{Lk_f}{4I_{yy}} & \frac{Lk_f}{4I_{yy}} \\ \frac{\gamma k_f}{4I_{zz}} & -\frac{\gamma k_f}{4I_{zz}} & \frac{\gamma k_f}{4I_{zz}} & -\frac{\gamma k_f}{4I_{zz}} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.28}
$$

Note: $0_{3\times3}$ is a 3×3 zero matrix and $1_{3\times3}$ is a 3×3 identity matrix.

2.3.2.2 Position Dynamics

$$
\begin{aligned}
&\text{State vector:} && \boldsymbol{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T \\
&\text{Input vector:} && \boldsymbol{u} = \begin{bmatrix} \phi & \theta & u_1 \end{bmatrix}^T \\
&\text{Trim States :} && \boldsymbol{x_{trim}} = \begin{bmatrix} x_0 & y_0 & z_0 & 0 & 0 & 0 \end{bmatrix}^T \\
&\text{Trim Inputs :} && \boldsymbol{u_{trim}} = \tfrac{mg}{k_f}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T
\end{aligned}
$$

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & 1_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -g\sin(\psi_0) & -g\cos(\psi_0) & 0 \\ g\cos(\psi_0) & -g\sin(\psi_0) & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ u_1 \end{bmatrix} \tag{2.29}
$$

Note: $0_{3\times3}$ is a 3×3 zero matrix and $1_{3\times3}$ is a 3×3 identity matrix.

2.4 Matlab Simulation

A *Matlab Simulink*^{TM} block diagram model, shown in Figure 2.8, is created using the equations and system parameters in the previous sections. The block diagram model includes two controllers (position and attitude), a vehicle model, a sensor block, and a state estimator. The vehicle model uses nonlinear dynamic equations discussed in previous sections and next chapter to emulate Crazyflie. The model accepts the motors' PWM control signal and outputs system states. Then, the sensor block models the inertial measurement unit on-board the Crazyflie, which converts true states from vehicle model to position, accelerations, and angular rate and injects noise to the signals. Lastly, the states estimator takes sensor's measurement and estimates the system states. A simple complementary filter is used because it is implemented in Crazyflie's stock firmware. This Simulink block model is used in Chapter 4 to tune controllers.

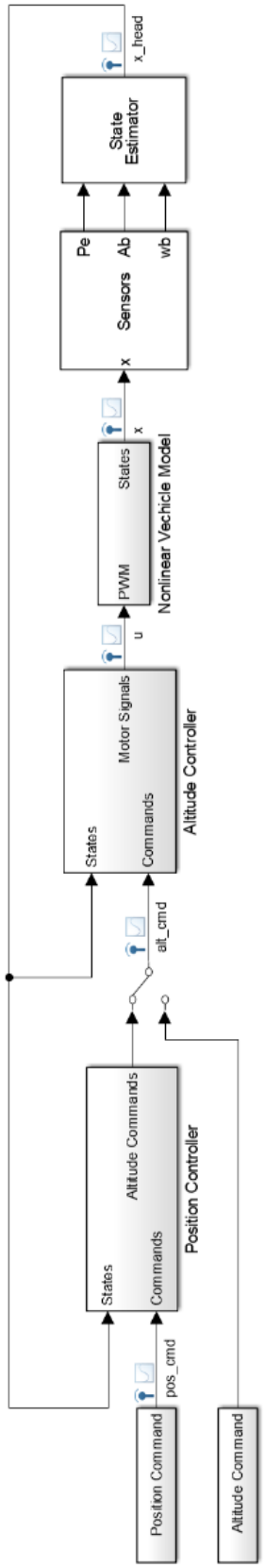

Figure 2.8: *Matlab Simulink*^{TM} model

Chapter 3

Ground Effects

To compensate for ground effects in controller design, an understanding of ground effect is desired. In this case, I am interested in how the total thrust varies when a quadcopter is at close proximity to the ground. From [31], the proposed model for a quadcopter with zero forward speed is shown as Equation 1.1. However, a validation is needed, and a ground test is proposed due to its simplicity; several assumptions are made to emulate in-flight condition.

1. Ground effect is affected mainly by the distance from ground, not vehicle's vertical velocity. Therefore, testing done statically on ground will have the same result as tests done dynamically in-flight.

2. Small objects placed at the center of the quadcopter have minimal affect on air flow, since it is away from the rotors.

To measure the thrust generated by the rotors, the quadcopter is placed upside down on a scale with a support as shown in Figure 3.1. The support elevates the quadcopter from the scale by three times the diameter of its rotor, such that intake air flows freely without interference. Then, the scale is placed on a flat platform, where an ultrasonic sensor is attached. The ultrasonic sensor measures distance from the platform to the "ground". Then, the sensor is connected to a microcontroller for data collection and filtering. The test rig is shown in Figure 3.1. Lastly, the test rig is placed under a long flat table and atop of a platform jack as shown in Figure 3.2. The long table emulates

the "ground", and measurement of ground effect at different height is done by adjusting the platform jack.

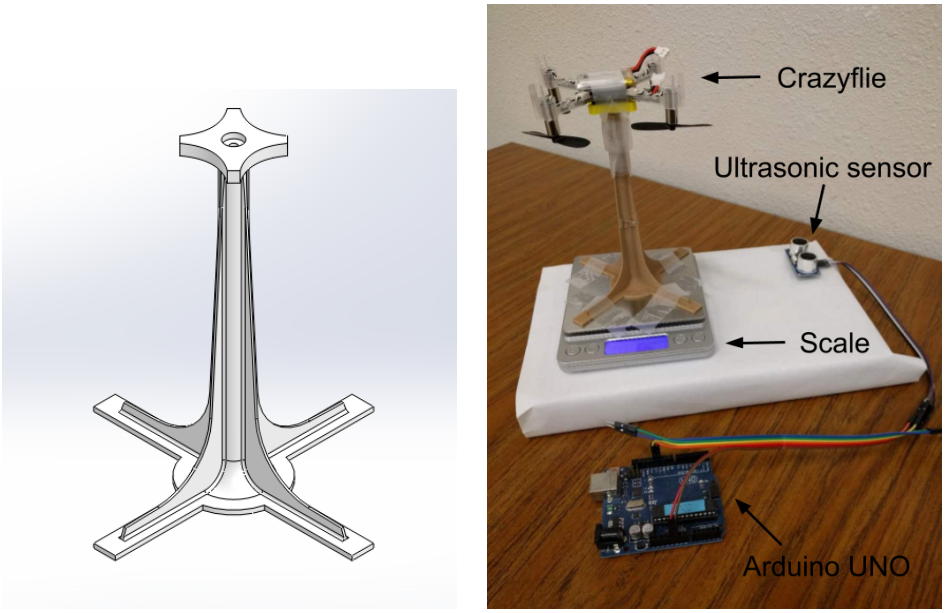

Figure 3.1: Left: a 3-D model for the support, Right: a ground effect test platform

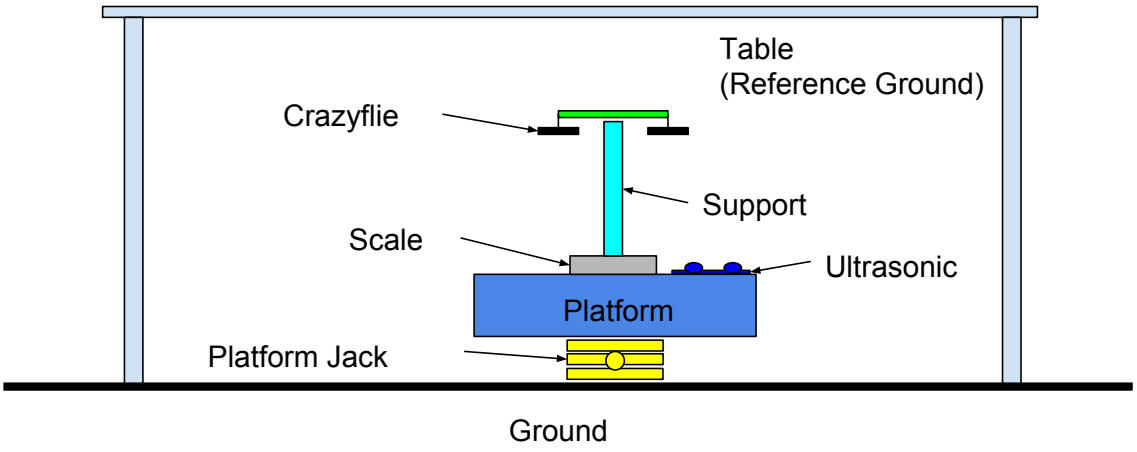

Figure 3.2: Ground effect experiment

3.1 Experimental Method

Before the experiment begins, the calibration of the ultrasonic sensor is carried out, and its setup is shown in Figure 3.3. Then, the ground effect experiment begins. The steps

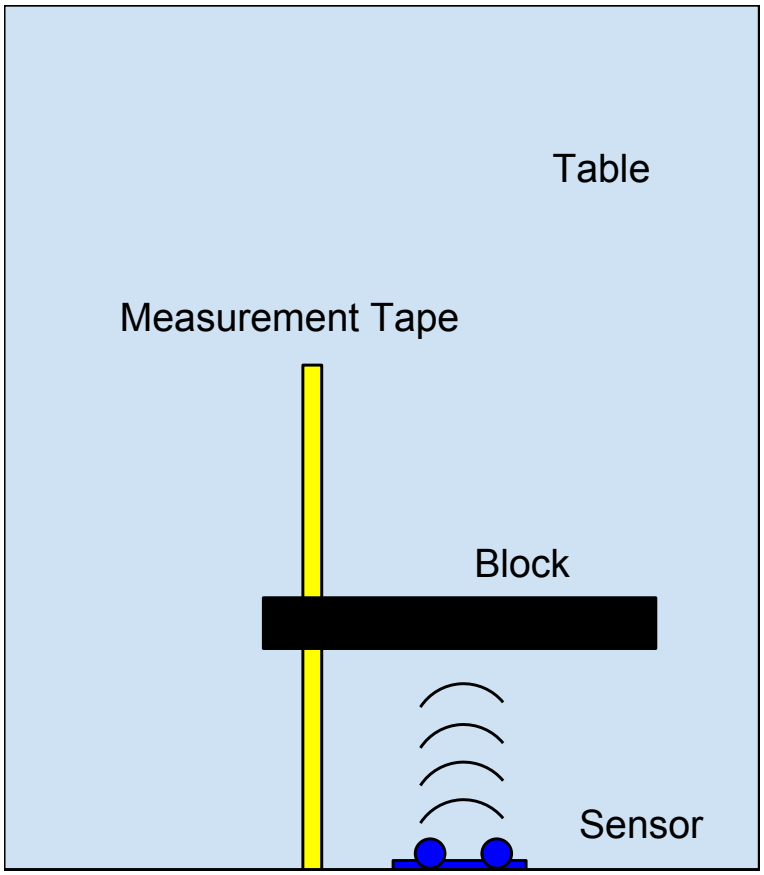

Figure 3.3: Ultrasonic sensor calibration

taken during the calibration and experiment are listed below.

1. ***Ultrasonic sensor calibration***

   (a) Place a flat object 10cm away from the sensor using the measurement tape. (Note: the minimum distance for the sensor is 5cm).

   (b) Record the tape measurement and ultrasonic sensor reading.

   (c) Repeat Step a - b at an increment of 2cm until 30cm away from the sensor.

2. ***Ground effect measurements***

   (a) Make sure the Crazyflie is fully charged

   (b) Place the test rig in place and zero the scale reading

   (c) Turn the knob on the platform jack to raise the test platform until the Crazyflie slightly touchs the table.

(d) Check if the scale still reads zero. If not, lower the platform jack.

(e) Set the throttle to a desired PWM value (around 10000 out of 65536) and check if the propellers are spinning.
(Note: If the throttle is too high, it will drain the battery out before the experiment ends.)

(f) Record the thrust measured by the scale, (F), and the distance with ultrasonic sensor, (Z_{sensor}).

(g) Turn the knob on the platform jack to lower the test platform.

(h) Repeat Step (e) - (g) at an increment 1cm until the distance is 20cm.

(i) Carefully relocate the test platform to a new location such that nothing blocks the intake air flow of the rotors.

(j) Record the scale's reading, which is the thrust outside ground effect (F_0).

(k) Repeat Step (a) - (j) for the quadcopter with two rotors (motor 1 and 3) active and four rotors active.

3.2 Method

Three sets of data are collected during the experiments: 1) thrust in ground effect region as F, 2) distance from the test platform to the "ground" as Z_{sensor}, and 3) thrust out of ground effect region as F_0. First, sensor reading, Z_{sensor}, is converted to distance [cm], Z_{actual}, using Equation (3.1), which comes from ultrasonic sensor calibration process. Then, Z_{actual} is subtracted by the distance between the ultrasonic sensor to the Crazyflie's rotors, which is a fixed distance, to get the distance from the rotors to the table, Z. Then, F/F_0 and Z/R are computed to compare different configuration, where R is the radius of the rotor. Lastly, a model is fitted to the data from each trial using the least square method, and the data from other trials are used to compute the square of residuals, which is defined as $(y_{model} - y_{experiment})^2$. The model that has the least residual is selected to prevent over-fitting.

3.3 Analysis

3.3.1 Calibration

The data collected during calibration is plotted in Figure 3.4, shown as blue dots. x is reading from ultrasonic sensor, and y is the tape measurement. A line is fitted to the data and its equation is shown in Equation (3.1).

$$y = 0.9874x + 1.5795 \tag{3.1}$$

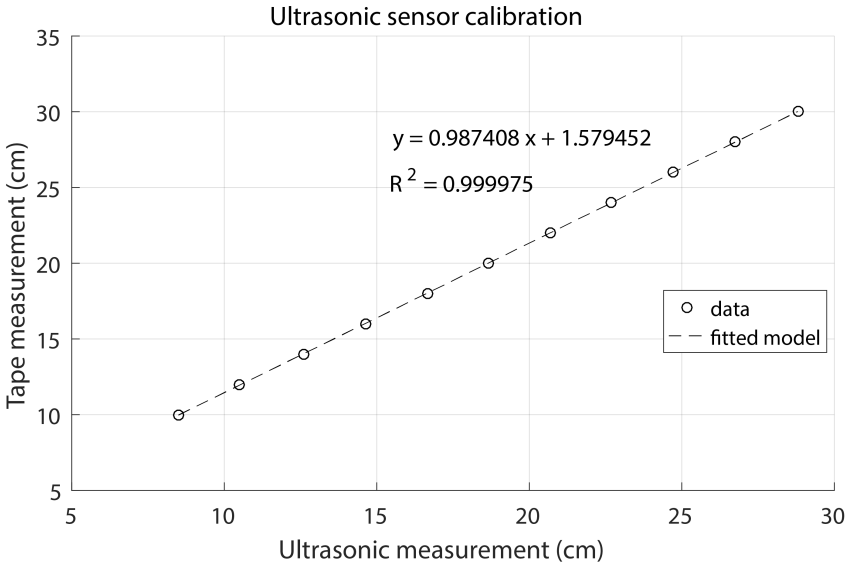

Figure 3.4: Ultrasonic sensor calibration curve

3.3.2 Ground Test

Using the data from the experiments, F/F_0 and Z/R are computed and plotted in figures below. Figure 3.5 shows the results with 2 rotors active, while Figure 3.6 has 4 propellers on. For both configurations, similar trends are observed. As the vehicle approaches the ground, F/F_0 increases from 1 to 1.2 and higher, which means more thrust is generated at close proximity to ground. This is an indication of ground effect. An exponential trend is observed for the 2-propeller case, where the vehicle experiences a 2% increase in thrust while it is 1.5 times its rotor's radius away from the ground. On the other hand, the 4-propeller case shows a linear trend, and the vehicle only needs to reach 2.5 times its

rotor radius to experience the same magnitude of increase in thrust. Models are fitted to both datasets for comparison. For the 2-rotor case, the proposed model, Equation 1.1, is used, while a linear model is used for the 4-propeller case. A summary of models used is given below, and the plots are shown in Figures 3.5 and 3.6.

1. The proposed model for quadcopter from [31], which comes from **single rotor vehicle**:

$$\frac{F}{F_0} = \frac{1}{1 - \frac{R^2}{16Z^2}} \tag{3.2}$$

2. Curve fitting using the proposed model for quadcopter with **2 rotors** active:

$$\frac{F}{F_0} = \frac{1}{1 - \frac{R^2}{26.4Z^2}} \tag{3.3}$$

3. Curve fitting using linear model for quadcopter with **4 rotors** active:

$$\frac{F}{F_0} = \begin{cases} -0.0623\frac{Z}{R} + 1.1868, & \text{if } \frac{Z}{R} < 2.9969 \\ 1.0000, & \text{otherwise} \end{cases} \tag{3.4}$$

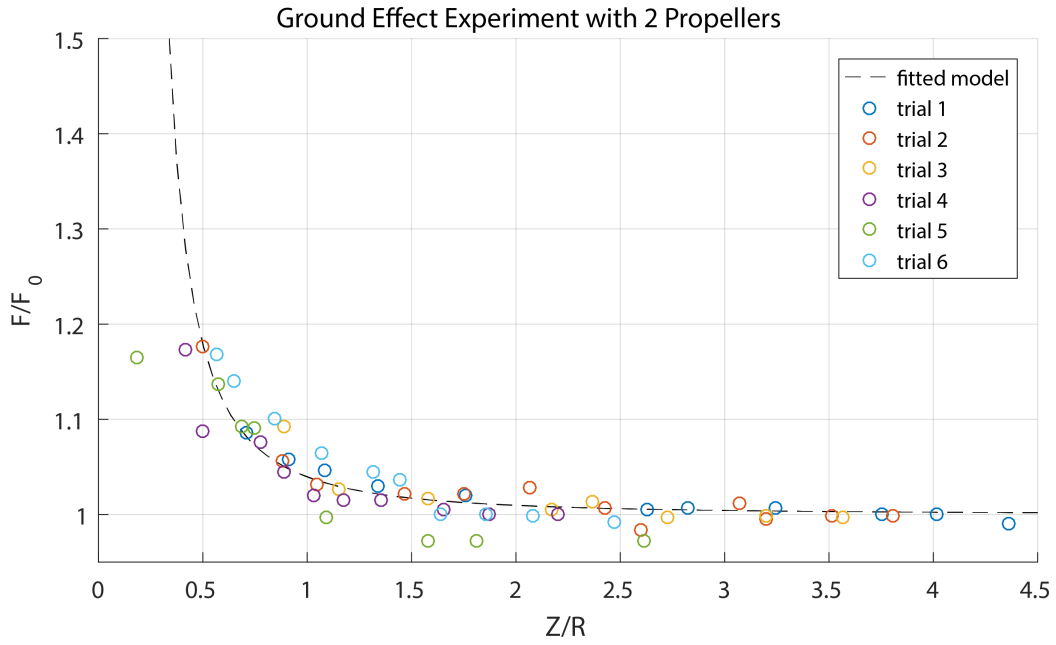

Figure 3.5: Ground effect ground experiment with 2 propellers

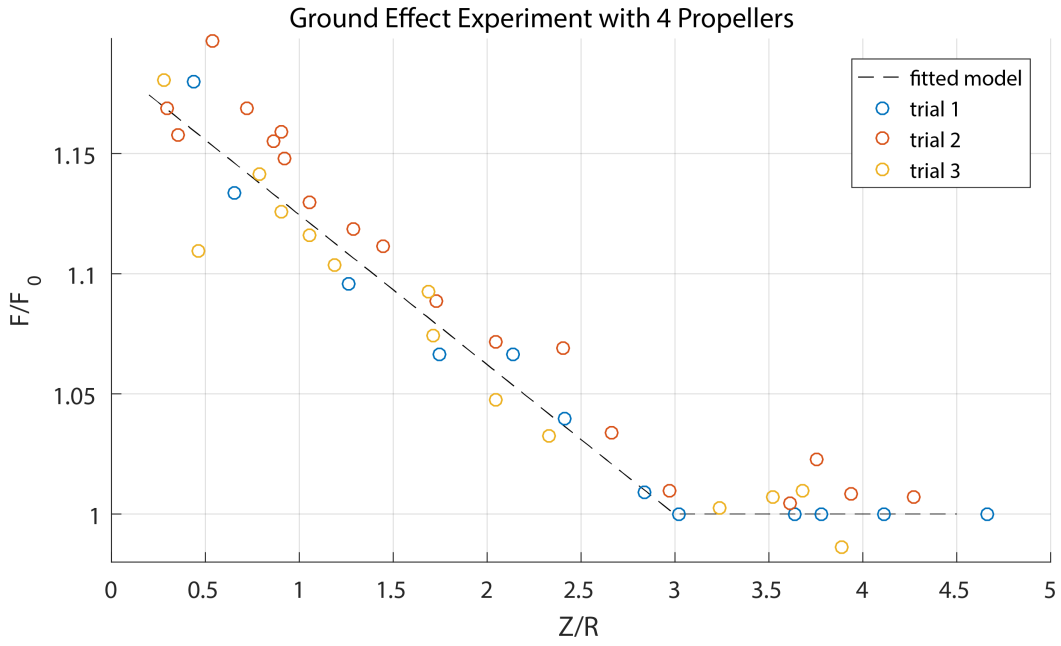

Figure 3.6: Ground effect ground experiment with 4 propellers

The models for each case (Equation 1.1, 3.3, and 3.4) are plotted together in Figure 3.7 for comparison. Obviously, the single rotor case and the 2-rotor case show a similar exponential pattern, but the 2-rotor case experiences less ground effect, which may be caused by the interaction of rotors' airflow. However, the 4-rotor case shows a linear relationship, which is different from the others. This discrepancy can be caused by the difference in rotor's spinning direction. All active rotors spin in the same direction for the single rotor and the 2-rotor case (M1 and M3, see Figure 2.1), while two pairs of rotors spin in opposite direction for the 4-rotor case (see Figure 2.1). To ensure that the discrepancy is caused by the actual aerodynamics not the experiment, a flight test is proposed to verify the data collected in the experiment.

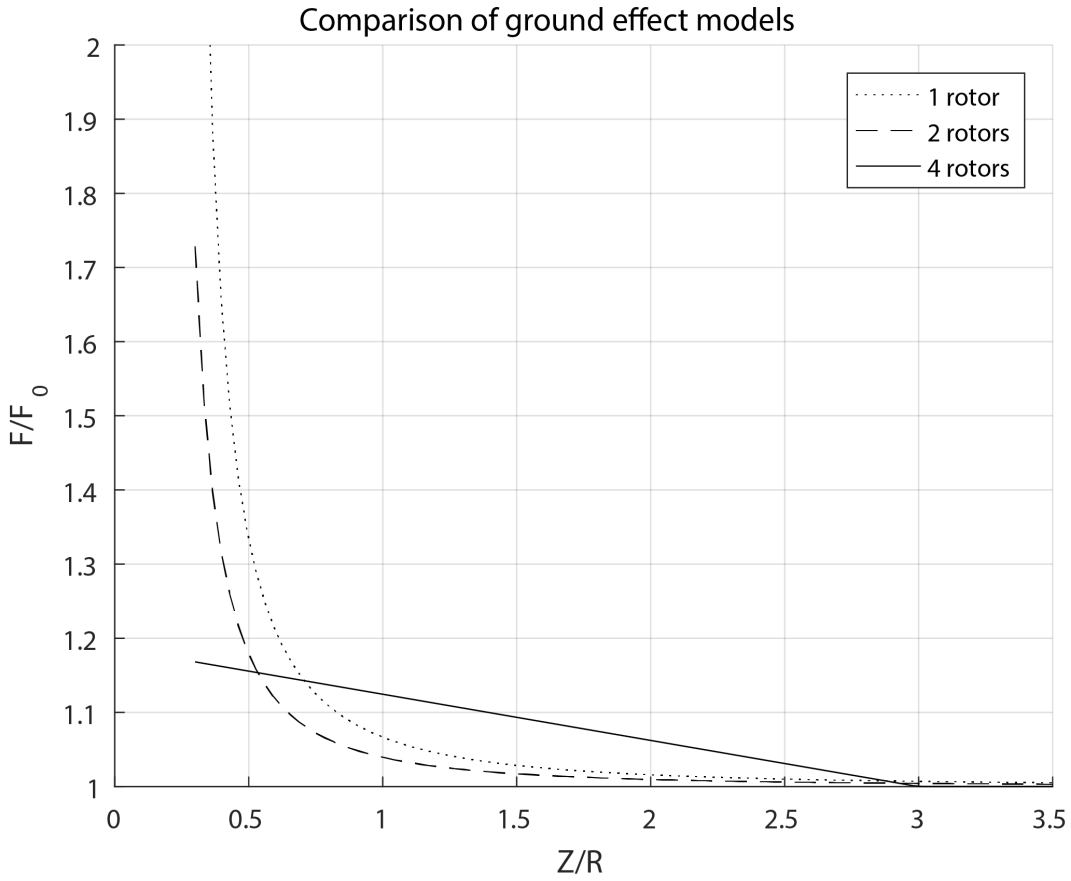

Figure 3.7: Comparison of different ground effect models

3.3.3 Flight Test

In the test, a quadcopter takes off outside of the ground effect region, where $Z/R \approx 12$. Then, the vehicle descends to the ground through multiple steps. During each step, the vehicle hovers for 10 seconds at certain altitude. The vehicle trajectory from the test is shown in Figure 3.8. During the test, these parameters are recorded: the motor's PWM control signal, altitude, and weight of the vehicle. Note that only the data from the second half of the 10-second duration is used to ensure the vehicle is hovering without vertical motion, such that no net force acts on the vehicle and thrust equals to vehicle's weight. Using this information, the thrust with ground effect, F, is the weight of the vehicle; the thrust without ground effect, F_0, is calculated using the recorded motor control signal and the theoretical thrust equation, Equation (2.1). Then, F/F_0 and Z/R are computed and compared with ground test result, which is shown in Figure 3.9. Both flight test and ground test data show a linear pattern although the flight test result is offset a bit vertically from the ground test. This is probably due to underestimated vehicle weight or overestimated thrust from motors.

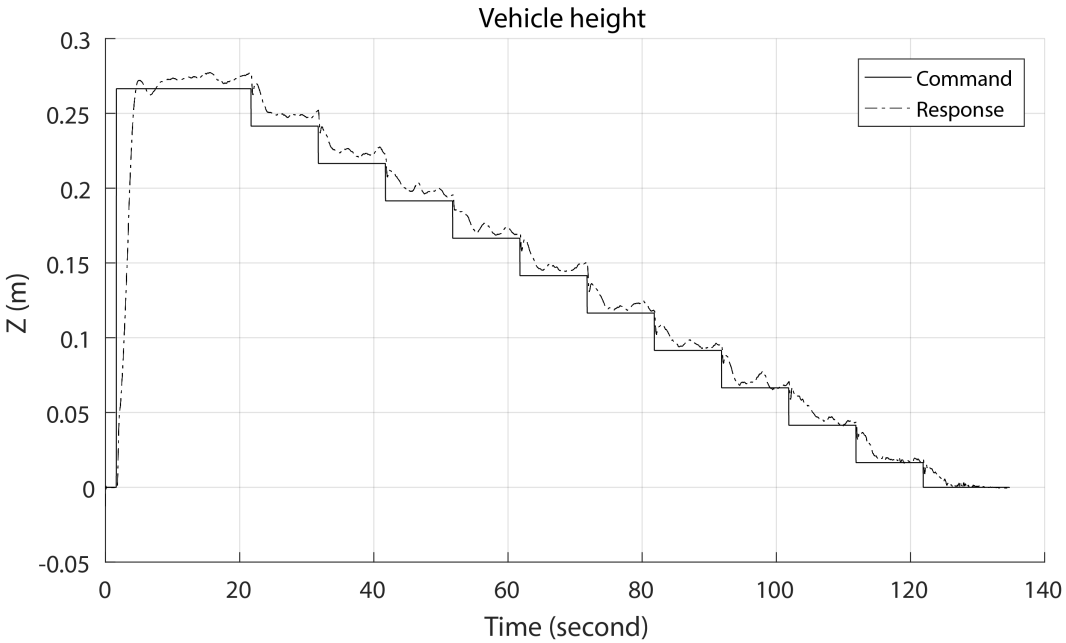

Figure 3.8: Flight test vehicle height from ground

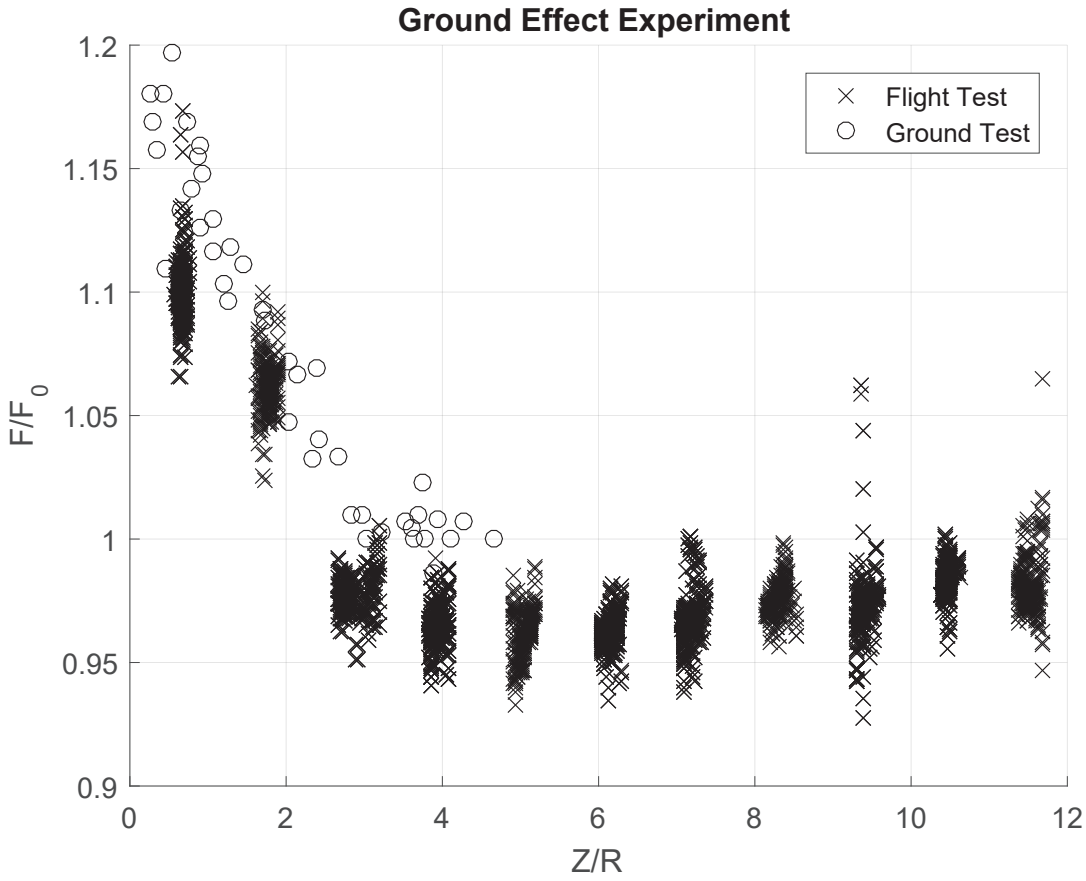

Figure 3.9: Ground test vs Flight test

Chapter 4

Control Architecture

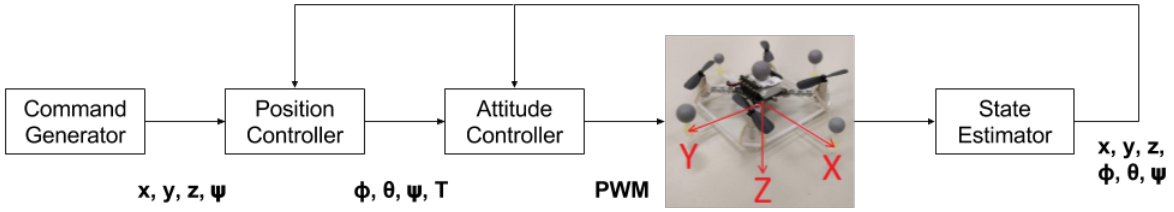

Figure 4.1: Control architecture overview

The goal of the proposed controller is to mitigate ground effect while providing waypoint following for a quadcopter. The linearized quadcopter dynamic model from Chapter 2 is used to simplify control design, since high maneuverability is not the focus of this controller. Looking at the dynamic model, it is obvious that attitude and position dynamics are decoupled. Therefore, a multilayer control architecture along with a MRAC is proposed to solve the waypoint following problem. It consists of a attitude (inner) loop and a position (outer) loop. First, the LQR technique is used to design a full state feedback controller to stabilize the quadcopter and reject disturbance, and a simple feedforward controller is used for attitude command tracking. Then, PID controllers are designed separately for x, y, and altitude. Since ground effect has a huge effect on altitude when the vehicle operates in close proximity to ground, a MRAC is added to altitude PID controller. The overall structure is illustrated in Figure 4.1. A desired position [meter] and heading [radian] (x, y, z, ψ) is given to the position controller, which outputs the desired

Tait-Bryan angles [radian] and thrust [PWM] (ψ, θ, ψ, P). Then, the attitude controller takes those outputs and computes the motor's control signals [PWM], and they are sent to the motor. In the following sections, details for each controller are provided.

4.1 Attitude Control

The attitude controller consists of a LQR and a feedforward controller. First, a full state feedback controller is designed using the LQR technique. Then, feedforward gains are computed based on the closed loop model from the LQR. The full state feedback controller takes the vehicle's current Tait-Bryan angles (ϕ, θ, ψ) and body angular rate (p, q, r) as inputs and outputs the motors' control signals. On the other hand, the feedforward controller takes attitude and thrust commands (ϕ_c, θ_c, ψ_c, P_c) as inputs and outputs the motors' control signals as well. By adding both outputs, the total control outputs are generated as shown in Equation (4.1). Figure 4.2 provides an illustration for the general structure.

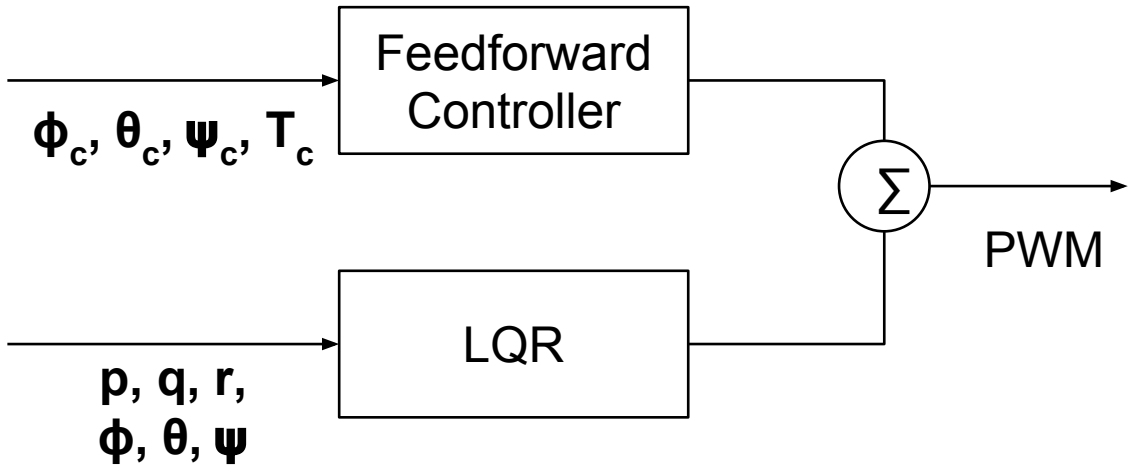

Figure 4.2: Overview of Attitude controller

$$u = u_{lqr} + u_{fwd} \tag{4.1}$$

4.1.1 LQR

Consider a system in the following form, and let x be the state vector, u be the input vector, and A, B, C, D be system dependent matrices.

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} + D\boldsymbol{u} \end{aligned} \tag{4.2}$$

Define a quadratic cost function which depends on system states and control inputs.

$$J = \int_0^\infty \left(\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{u}^T R \boldsymbol{u}\right) dt \tag{4.3}$$

where the cost weight matrices (Q, R) are

$$Q = Q^T \geq 0 \text{ and } R = R^T \geq 0 \tag{4.4}$$

The LQR algorithm minimizes the quadratic cost function, which gives a controller that rejects disturbance with minimal control effort. Given that the system (A, B, C, D) is controllable and observable and the cost weight matrices (Q, R) are provided, the resultant feedback control law is

$$\boldsymbol{u}_{lqr} = -K_{lqr}\boldsymbol{x} \tag{4.5}$$

where

$$K_{lqr} = R^{-1}B^T P \tag{4.6}$$

and P is found by solving the Riccati equation.

$$A^T P + PA + Q = 0 \tag{4.7}$$

By combining Equation (4.2) and (4.6), the closed loop model is shown in Equation (4.8). Notice that the closed loop equations do not have a command tracking term, y_{cmd}, since LQR is a full state feedback controller, which rejects disturbance but not necessarily tracks a command. Although integral feedback terms can be added to provide such capability, a simple attitude controller is desired for embedded system. Therefore, a feedforward controller is chosen for command tracking.

$$\begin{aligned} \dot{\boldsymbol{x}} &= (A - BK_{lqr})\,\boldsymbol{x} \\ \boldsymbol{y} &= (C - DK_{lqr})\,\boldsymbol{x} \end{aligned} \tag{4.8}$$

4.1.2 Feedforward

The feedforawrd controller takes a command vector, $\boldsymbol{y_{cmd}} = \begin{bmatrix} \phi_c & \theta_c & \psi_c & P_c \end{bmatrix}$. Then, it is multiplied by a constant 4-by-4 matrix, M_{ru}, to map the commands to PWM signals for motors, $\boldsymbol{u_{fwd}}$. This relationship is described in Equation (4.9).

$$\boldsymbol{u_{fwd}} = M_{ru}\boldsymbol{y_{cmd}} \tag{4.9}$$

where

$$M_{ru} = \begin{bmatrix} -k_\phi & k_\theta & k_\psi & k_{thrust} \\ -k_\phi & -k_\theta & -k_\psi & k_{thrust} \\ k_\phi & -k_\theta & k_\psi & k_{thrust} \\ k_\phi & k_\theta & -k_\psi & k_{thrust} \end{bmatrix} \tag{4.10}$$

By combining Equation (4.1), (4.2), (4.5), and (4.9), the closed loop attitude model is shown below.

$$\begin{aligned} \dot{\boldsymbol{x}} &= (A - BK_{lqr})\,\boldsymbol{x} + BM_{ru}\boldsymbol{y_{cmd}} \\ \boldsymbol{y} &= (C - DK_{lqr})\,\boldsymbol{x} + DM_{ru}\boldsymbol{y_{cmd}} \end{aligned} \tag{4.11}$$

Define C as a 3×6 matrix and D as a 3×4 zero matrix, such that the system outputs are Tait-Bryan angles, $\boldsymbol{y} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}$. Notice that the attitude dynamic equations does not include thrust (P), so it is just pass-through ($k_{thrust} = 1$).

$$C = \begin{bmatrix} 0_{3x3} & 1_{3x3} \end{bmatrix} \text{ and } D = 0_{3x4} \tag{4.12}$$

Then, Equation (4.11) becomes

$$\begin{aligned} \dot{\boldsymbol{x}} &= (A - BK_{lqr})\,\boldsymbol{x} + BM_{ru}\boldsymbol{y_{cmd}} \\ \boldsymbol{y} &= \begin{bmatrix} 0_{3x3} & 1_{3x3} \end{bmatrix} \boldsymbol{x} \end{aligned} \tag{4.13}$$

To satisfy the command following requirement, the closed loop DC gains from commands to system outputs must be 1. To find the DC gains, Equation (4.13) is converted to a transfer function matrix using the equation below:

$$\frac{Y}{Y_{cmd}}(s) = G(s) = C_{ref}\,(sI - A_{ref})\,B_{ref} \tag{4.14}$$

where

$$\begin{aligned} A_{ref} &= A - BK_{lqr} \\ B_{ref} &= BM_{ru} \\ C_{ref} &= \begin{bmatrix} 0_{3\times3} & 1_{3\times3} \end{bmatrix} \end{aligned} \tag{4.15}$$

Then, the transfer functions for each system output (ϕ, θ, ψ) are extracted and the coupled terms are neglected for simplicity. The transfer function for roll (G_ϕ) comes from the first row and first column of the transfer function matrix ($G(s)$). For others, it is summarized below:

$$\begin{aligned} G_\phi &= G(s)(1,1) \\ G_\theta &= G(s)(2,2) \\ G_\psi &= G(s)(3,3) \end{aligned} \tag{4.16}$$

Lastly, the feedforward gains are found by inverting the DC gains of the closed-loop transfer functions, such that the DC gains are 1.

$$\begin{aligned} k_\phi &= \frac{1}{DCgain\,(G_\phi)} \\ k_\theta &= \frac{1}{DCgain\,(G_\theta)} \\ k_\psi &= \frac{1}{DCgain\,(G_\psi)} \\ k_{thrust} &= 1 \end{aligned} \tag{4.17}$$

4.1.3 Design

The system matrices (A and B) come from the linearized attitude dynamic equations, Equation (2.28), (C and D) come from Equation (4.12), the system state vector is $x = \begin{bmatrix} p & q & r & \phi & \theta & \psi \end{bmatrix}^T$, and the input vector is $u = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \end{bmatrix}^T$. Notice that there are 6 states and 4 inputs, so Q and R are selected as a 6×6 and a 4×4 identity matrix respectively. They are then used in the Matlab model created in Chapter 2. Step inputs are used to excite the system, and its performance is recorded in terms of crossover frequency, steady state error, and overshoot. The cost weight matrices (Q, R) are updated after each trial, and the one with the best performance is selected for flight test, which

is shown in Equation (4.18). Notice that R_{lqr} is very small compared to Q_{lqr}, such that penalty for high control effort is limited and high motor control signal (PWM signal ranges from 0 to 65536) can be generated for stabilization. Furthermore, the crossover frequency is at 2.87 rad/s for roll and pitch loop, while the yaw loop is slower at 1.81 rad/s as shown in Figure 4.3 and 4.4.

$$Q_{lqr} = 100 \begin{bmatrix} 1_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10\,(1_{3\times3}) \end{bmatrix} \text{ and } R_{lqr} = 10^{-6} 1_{4\times4} \tag{4.18}$$

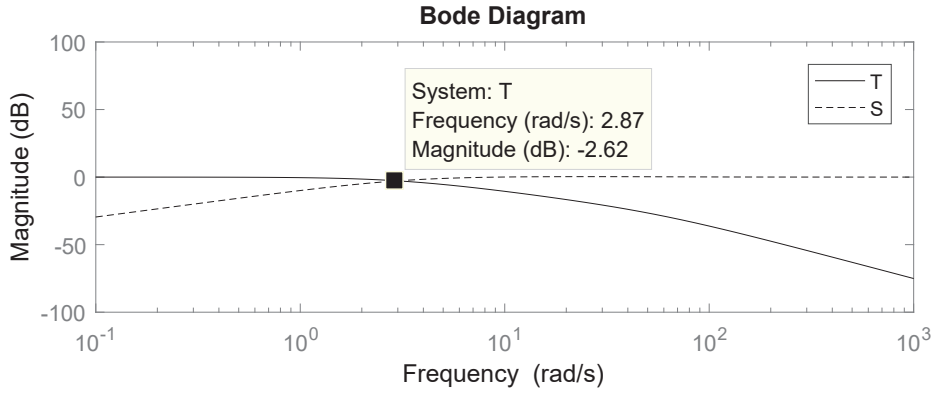

Figure 4.3: Frequency response for closed roll/pitch loop, T: complementary sensitivity and S: sensitivity

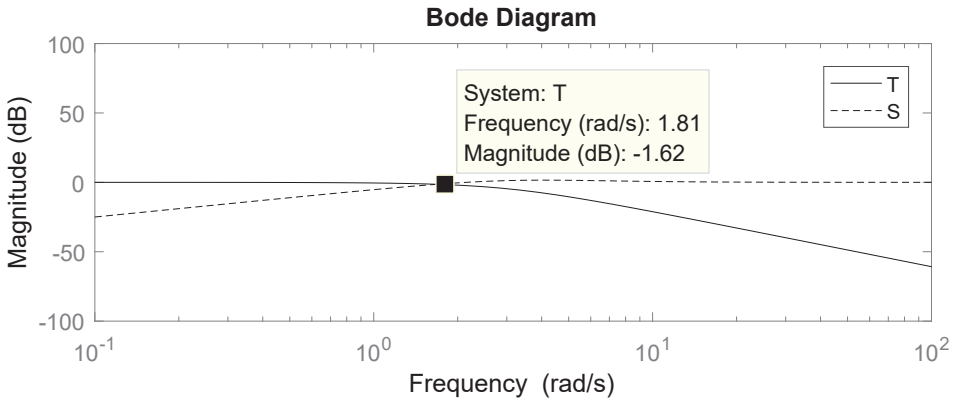

Figure 4.4: Frequency response for closed yaw loop, T: complementary sensitivity and S: sensitivity

Then, the feedback gain matrix is found by solving Equation (4.6) and (4.7).

$$K_{lqr} = \begin{bmatrix} -k_p & k_q & k_r & -k_\phi & k_\theta & k_\psi \\ -k_p & -k_q & -k_r & -k_\phi & -k_\theta & -k_\psi \\ k_p & -k_q & k_r & k_\phi & -k_\theta & k_\psi \\ k_p & k_q & -k_r & k_\phi & k_\theta & -k_\psi \end{bmatrix} \tag{4.19}$$

where

$$k_p = 5258.0500 \quad k_q = 5275.6847 \quad k_r = 8920.3470$$
$$k_\phi = 15811.3883 \quad k_\theta = 15811.3883 \quad k_\psi = 15811.3883$$

Using the results above from LQR and Equation (4.17), the feedforward gain matrix is

$$M_{ru} = \begin{bmatrix} -k_\phi & k_\theta & k_\psi & k_{thrust} \\ -k_\phi & -k_\theta & -k_\psi & k_{thrust} \\ k_\phi & -k_\theta & k_\psi & k_{thrust} \\ k_\phi & k_\theta & -k_\psi & k_{thrust} \end{bmatrix} \tag{4.20}$$

where $k_\phi = 15811.3883$, $k_\theta = 15811.3883$, $k_\psi = 15811.3883$, and $k_{thrust} = 1$

And the outputs of the attitude controller are

$$\boldsymbol{u} = K_{lqr}\boldsymbol{x} + M_{ru}\boldsymbol{y_{cmd}} \tag{4.21}$$

4.2 Position Control

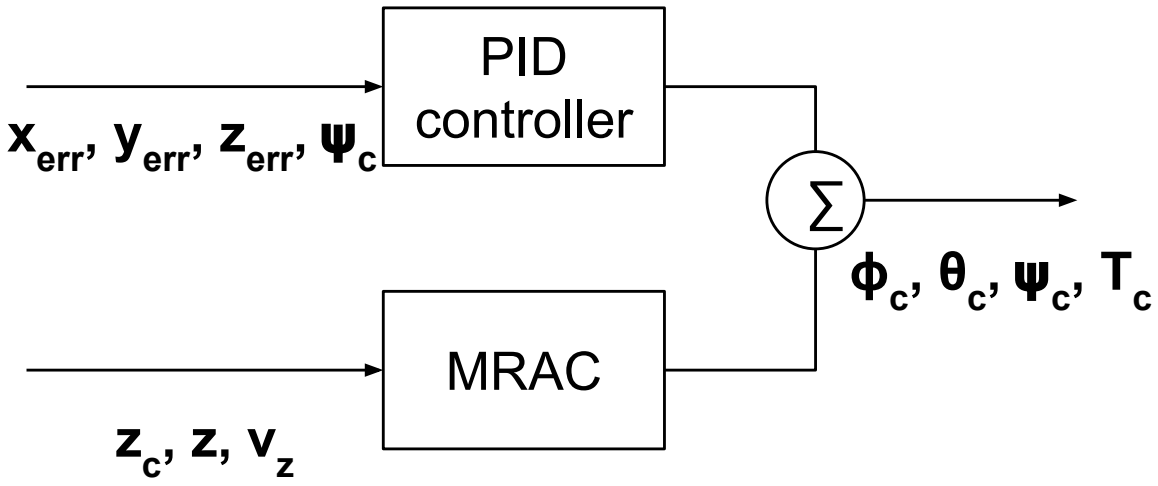

Figure 4.5: Overview of position controller

The position controller includes a PID controller and a MRAC. The PID controller takes position errors and commanded yaw angle (x_{err}, z_{err}, z_{err}, ψ_c) as inputs and outputs desired Tait-Bryan angles and thrust (ϕ_c, θ_c, ψ_c, T_c). The MRAC takes commanded altitude, measured altitude, and measured vertical velocity (z_c, z, $\dot{v}_z$) as inputs and outputs desired thrust (T_c). Then, outputs from each controller are added together, see Equation (4.22), and the result is the command for the inner loop as shown in Figure 4.5 above.

$$u_z = u_{pid} + u_{adaptive} \tag{4.22}$$

4.2.1 PID

A PID position controller is an error feedback controller, where error is multiplied by a constant gain, K_p; accumulated error is multiplied by K_i; rate of change in error is multiplied by K_d, as described in Equation (4.23). Error (e_x, e_y, e_z) is defined as the difference between the commanded and the measured position as shown in Equation (4.24). Notice that the position error fed to the PID controller is in Vehicle-1 frame, because the heading of the vehicle does not always align with the unit vector, x_e, in inertial frame, which causes tracking error and a stabilization issue. To compute error in Vehicle-1 frame, it is first computed in inertial frame. Then, it is rotated to Vehicle-1

frame using equation from Section 2.2.4, which is shown in Equation (4.25)

$$\boldsymbol{u} = K_p \boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(\tau)\, d\tau + K_d \frac{d\boldsymbol{e}(t)}{dt} \tag{4.23}$$

where $\boldsymbol{e} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T$ and $\boldsymbol{u} = \begin{bmatrix} \phi_c & \theta_c & \psi_c & T_c \end{bmatrix}^T$.

$$\begin{aligned} e_x &= x_c - x \\ e_y &= y_c - y \\ e_z &= z_c - z \end{aligned} \tag{4.24}$$

$$\begin{bmatrix} e_x^{v1} \\ e_y^{v1} \\ e_z^{v1} \end{bmatrix} = R_v^{v1}(\psi) \begin{bmatrix} e_x^{v} \\ e_y^{v} \\ e_z^{v} \end{bmatrix} \tag{4.25}$$

Design

To find the gains, (K_p, K_i, K_d), a simplified position dynamic model is created in *Matlab Simulink*^{TM}. The model includes linearized position dynamic equations from Equation (2.29), a second order actuator model (bandwidth = 10Hz, damping ratio = 1.0), and PID controllers. The inner loop is neglected in this model, because the DC gain of inner loop is 1 if the outer loop has a lower crossover frequency than the inner loop by 5 to 10 times according to successive loop closure technique[35].

Crossover frequency is chosen to be 0.3 rad/s, which is about 1/10 of the inner loop, and it is desired to have zero steady error and zero overshoot. Using MatLab's PID tuner, the PID gains are tuned and shown in Table 4.1. Then, the gains are implemented in the full model from Chapter 2 and Crazyflie. During flight test, the performance of the control is satisfied during the mission, but it has huge steady state error during takeoff and landing. Therefore, a model reference adaptive controller is included in altitude loop to mitigate ground effect.

Table 4.1: PID Position Controller Gains

	K_p	K_i	K_d
X and Y	0.388869225	-0.074870430	-0.504936570
Z	42692.2	8219.7	55434.8

4.2.2 Model Reference Adaptive Control

A model reference adaptive controller (MRAC) has adjustable parameters and a mechanism to update those parameters on-line, which ensures that the system states follows the model states with the presence of uncertainties and external disturbance. Since this thesis focuses on the application of MRAC, only a brief description on the equations used to design a MRAC for altitude loop is provided in the following sections. However, details can be found in [36].

Consider a system that contains unknown matched system uncertainty, $\boldsymbol{f}(\boldsymbol{x})$.

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\left(\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x})\right) \tag{4.26}$$

Assuming that $\boldsymbol{f}(\boldsymbol{x})$ can be written as a linear combination of N known locally Lipschitz-continuous basis functions, $\boldsymbol{\Phi}(\boldsymbol{x})$, with unknown constant coefficients, $\boldsymbol{\Theta}$.

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\Theta}^T \boldsymbol{\Phi}(\boldsymbol{x}) \tag{4.27}$$

Then, a reference model in the following form is chosen for desired performance.

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{ref}} &= A_{ref}\boldsymbol{x}_{\boldsymbol{ref}} + B_{ref}\boldsymbol{y}_{\boldsymbol{cmd}} \\ \dot{\boldsymbol{y}}_{\boldsymbol{ref}} &= C_{ref}\boldsymbol{x}_{\boldsymbol{ref}} + D_{ref}\boldsymbol{y}_{\boldsymbol{cmd}} \end{aligned} \tag{4.28}$$

Then, defining state tracking error as the difference between the system states and the model states as

$$\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_{\boldsymbol{ref}}(t) \tag{4.29}$$

Then, the control law is chosen to cancel the systen uncertainty.

$$\boldsymbol{u}_{\boldsymbol{adp}} = -\hat{\boldsymbol{\Theta}}^T \boldsymbol{\Phi}(\boldsymbol{x}) \tag{4.30}$$

where $\hat{\boldsymbol{\Theta}}$ is the estimated unknown constant coefficients for the matched uncertainty.

A radially unbounded quadratic Lyapunov function candidate is selected as the following:

$$V\left(e,\Delta\Theta\right)=e^{T}Pe+trace\left(\Delta\Theta^{T}\Gamma_{\Theta}^{-1}\Delta\Theta\right) \tag{4.31}$$

where $\Delta\Theta=\hat{\Theta}-\Theta$, P is the solution of the algebraic Lyapunov equation, $PA_{ref}+A_{ref}^{T}P=-Q$ with $Q=Q^{T}>0$, and Γ_{Θ} is rate of adaption.

By differentiating the equation above on both sides, $\dot{V}$ is found, the following condition must be met to ensure Lyapunov stability, $\dot{V}<0$. This adaptive law is updated on-line at each time step.

$$\dot{\hat{\Theta}}=\Gamma_{\Theta}\Phi\boldsymbol{e}^{T}PB \tag{4.32}$$

The equations used to implement the MRAC for altitude loop is summarized below.

Table 4.2: Summary of MRAC

Open loop	$\dot{\boldsymbol{x}}=A\boldsymbol{x}+B\left(\boldsymbol{u}+\Theta^{T}\boldsymbol{\Phi}\left(\boldsymbol{x}\right)\right)$
Reference model	$\dot{\boldsymbol{x}}_{\boldsymbol{ref}}=A_{ref}\boldsymbol{x}_{\boldsymbol{ref}}+B_{ref}\boldsymbol{y}_{\boldsymbol{cmd}}$
State tracking error	$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{ref}}$
Lyapunov equation	$PA_{ref}+A_{ref}^{T}P=-Q$
Adaptive law	$\dot{\hat{\Theta}}=\Gamma_{\Theta}\Phi\boldsymbol{e}^{T}PB$
Control input	$\boldsymbol{u}_{\boldsymbol{adp}}=-\hat{\Theta}^{T}\Phi\left(\boldsymbol{x}\right)$

Design

Reference Model

System identification is used to obtain the reference model, so that the closed loop system performs like the PID altitude controller. As a result, the system behaves consistently with or without the presence of ground effect. System states are recorded during flight tests, where the vehicle takes off and hovers at different heights. Then, the data is imported to $Matlab^{TM}$ System Identification software. Notice that data during takes off and landing is removed, so the model obtained does not account for ground effect. Then, second order state space models are fitted to each dataset, and the models are cross validated by other datasets. Finally, the model that has the least error is selected, which is shown in

Equation (4.33).

$$\begin{bmatrix} \dot{z} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5.416 & -7.027 \end{bmatrix} \begin{bmatrix} z \\ v_z \end{bmatrix} + \begin{bmatrix} -0.1145 \\ 6.273 \end{bmatrix} z_{cmd} \tag{4.33}$$

Uncertainty model

A set of system states dependent functions, $\mathbf{\Phi}(\boldsymbol{x})$, are selected, and a linear combination of those functions, $\hat{\mathbf{\Theta}}^T \mathbf{\Phi}(\boldsymbol{x})$, models the ground effect. Notice that $\mathbf{\Theta}$ is a vector of unknown constants, which are estimated on-line by the adaptive law. In this thesis, two different sets of functions are used: 1) polynomial functions and 2) radial basis functions.

1. Polynomial functions

   In Chapter 3, the ground effect experimental results show a linear pattern, which is in the form of $F = \theta_1 Z + \theta_0$. Therefore, the regressors are selected

   $$\Phi(\boldsymbol{x}) = \begin{bmatrix} Z & 1 \end{bmatrix} \tag{4.34}$$

   where Z is the distance from the ground.

   Simulation is used to pick the right Q and Γ_Θ, so the system states follow the reference states without oscillation. The result is shown below.

   $$\begin{aligned} \Gamma_\Theta &= 10^{-3} 1_{2\times 2} \\ Q &= \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{4.35}$$

2. Radial basis functions

   A radial basis function (RBF) is a real-valued function, which satisfies $\phi(x, c) = \phi(|x - c|)$. In other words, a RBF is symmetric about its center, c. A linear combination of those functions has been widely used to approximate given functions in different fields [37] [38]. Thus, a series of Gaussian RBFs, Equation (4.36), along with a basis term are used to approximate the ground effect.

   $$\phi(\boldsymbol{x}, c) = e^{-\epsilon^2 (x-c)^2} \tag{4.36}$$

The RBFs are chosen to have a center at 0 m, -0.2 m, -0.4 m, -0.6 m, -0.8 m, and -1.0 m to cover the whole flight envelop, and ϵ is chosen to be 6.67. The regressors are shown in Equation (4.37), and the RBFs are plotted in Figure 4.6

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} \phi_{(z,0)} & \phi_{(z,-0.2)} & \phi_{(z,-0.4)} & \phi_{(z,-0.6)} & \phi_{(z,-0.8)} & \phi_{(z,-1.0)} & 1 \end{bmatrix} \tag{4.37}$$

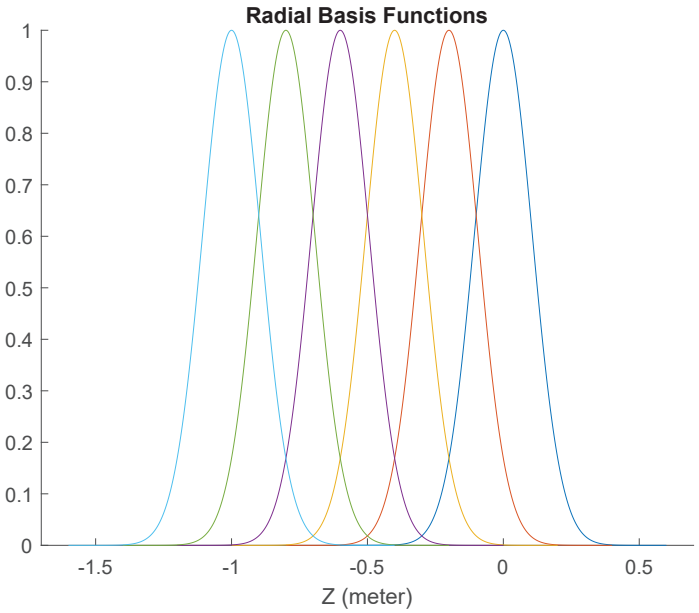

Figure 4.6: A set of radial basis functions used to approximate the ground effect function

Then, the same procedure is followed, and the result is

$$\begin{aligned} \Gamma_\Theta &= 10^{-3} 1_{7\times 7} \\ Q &= \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{4.38}$$

Chapter 5

Experiment Results

In this chapter, the control architecture proposed in the previous chapter is implemented on the test platform, Crazyflie 2.0. Then, takeoff/landing tests are performed to demonstrate performance impact on PID controller due to ground effect and how MRAC improves the performance.

5.1 Software Architecture

The system composes of a Crazyflie and a ground station, which are connected through a radio channel using CrazyRadio. Figure 5.1 illustrates the overview of the system. The inner loop control algorithm described in previous chapter is implemented in C and runs on-board the Crazyflie, which includes a IMU data processing unit, a state estimator (which is a complementary filter), and a attitude controller. The IMU data processing unit and the state estimator are provided by the stock firmware running at 1000Hz, while the attitude controller from previous chapter is implemented in C and runs at 500Hz. On the other hand, the outer loop logic is implemented in the ground station, which has a Robot Operating System (ROS) installed to handle communication with Crazyflie and between components. In the outer loop, it includes a position controller, a waypoint generator, and Optitrack. The position controller described in Chapter 4 is implemented in C and runs at 200Hz. The waypoint generator creates position command based on current location and an internal clock. Then, it is sent to the position controller. The waypoint generator is implemented in Python and runs at 50Hz. Lastly, a motion capture

system, OptiTrack, is utilized to gather vehicle's position data. The source is available at `https://github.com/fjctp/crazyflie_mrac`.

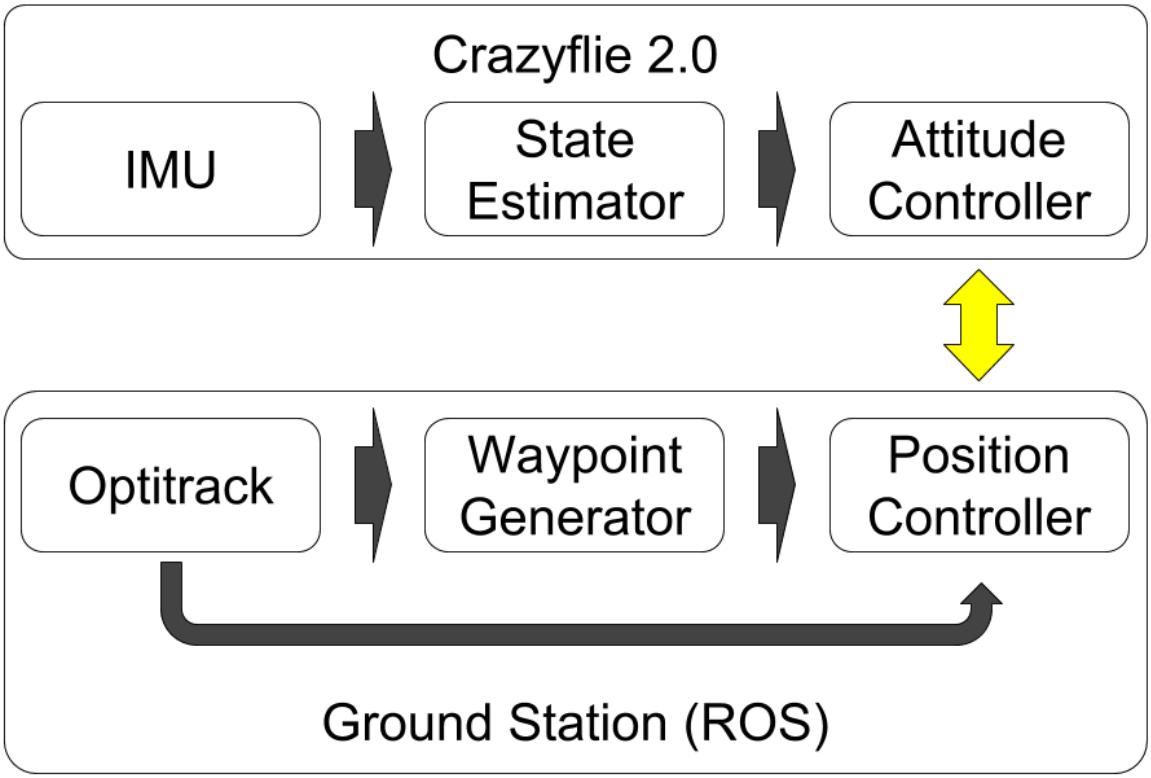

Figure 5.1: System overview

5.2 Experiment

An experiment is designed to study the impact on the performance of a position controller due to ground effect. During the experiment, the vehicle operates within the ground effect region, which is <10cm according to Chapter 3, and the vehicle takes off and lands for multiple times. The system states are recorded for analysis. Then, the whole experiment is repeated with different position controllers, which includes a PID controller, a PID controller with a MRAC using linear uncertainty model, and a PID controller with a MRAC using RBFs.

5.3 Results

Data collected during the experiment is resampled at 50Hz for analysis, because sensors run at different frequencies. Then, the vehicle's altitude is plotted in Figure 5.2 along with the commanded altitude and the reference altitude, where zero is the ground. Clearly, the adaptive controllers (red and yellow solid line) perform better than the PID controller (blue solid line). Both adaptive controllers reach the target altitude and track the reference model closely. However, the vehicle with a PID controller does not reach the target altitude during takeoff, and it does not touch-down when landing. Although a PID controller is robust to unmodeled dynamic (ground effect), it loses its performance. As a result, a manual intervention is needed at the end of the experiment to bring down the vehicle safely, which is shown in Figure 5.2.

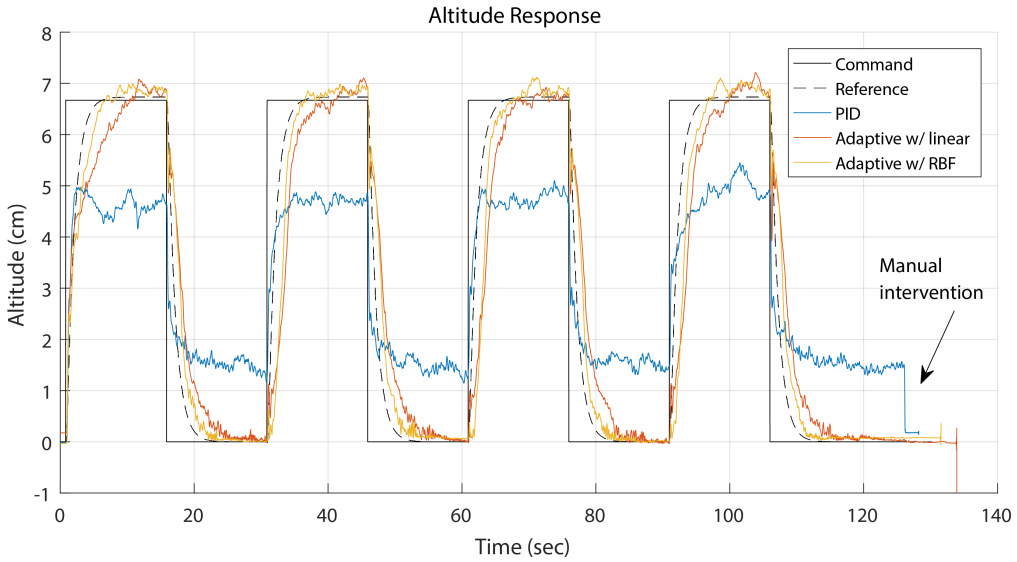

Figure 5.2: Vehicle's height above ground using different control algorithms (blue solid line: PID, red solid line: MRAC with linear model, yellow solid line: MRAC with RBFs)

To provide a better comparison between the controllers, Figure 5.2 is sliced into different sections when there is a change in command. For example, the first section is from 0 to 16 seconds, and the second section is from 16 to 31 seconds. Then, each section is identified as either a takeoff or landing operation depending on the command. For each operation, averages and standard derivations are computed at each time step to generate an average trajectory. The resultant curves are shown in Figures 5.3 (takeoff) and

5.4 (landing). Besides, performance indices for each section are also extracted, including steady state error with respect to the step command (SSE), mean squared error with respect to the reference model (MSE, using Equation (5.1)), rise time (time it takes for the vehicle's altitude to rise from 10% to 90% of the commanded altitude), and overshoot. Then, the averages and standard derivations are shown in Tables 5.1 and 5.2. Note that, the PID controller does not have a rise time and overshoot, because the vehicle does not reach 90% of the target altitude. The PID controller also does not have a mean square error, since it is not tracking the reference model.

$$MSE = \frac{1}{N}\sum (r - z)^2 \tag{5.1}$$

where r is the referenced altitude, z is the vehicle's altitude, N is number of time steps.

From Tables 5.1 and 5.2, MRACs clearly outperform PID as MRACs has less steady errors. Among the MRACs, MRAC with RBFs tracks the reference model better considering that it has a lower mean squared error. Furthermore, it responds faster and with less overshoot. Although MRAC with RBFs provides better performance, seven parameters are estimated on-line. On the other hand, only two parameters are estimated on-line for the MRAC using the linear model, and it still provides a significant improvement over the PID controller. It may not take significantly more computational power to estimate seven parameters on-line than two parameters on a computer, but it matters on an embedded system, which has limited power.

Table 5.1: Performance index for takeoff

	PID	PID w/ MRAC (Linear)	PID w/ MRAC (RBF)
SSE (cm)	1.87 ± 0.19	0.19 ± 0.08	0.21 ± 0.02
MSE (10^{-4})	—	1.10 ± 0.43	0.80 ± 0.44
Rise time (sec)	—	5.06 ± 0.58	3.48 ± 0.22
Overshoot (%)	—	6.14 ± 1.80	5.85 ± 0.92

Table 5.2: Performance index for landing

	PID	PID w/ MRAC (Linear)	PID w/ MRAC (RBF)
SSE (cm)	1.81 ± 0.04	0.39 ± 0.02	0.37 ± 0.04
MSE (10^{-4})	—	1.17 ± 0.09	0.68 ± 0.03
Rise time (sec)	—	6.42 ± 0.20	4.58 ± 0.13
Overshoot (%)	—	—	—

Considering the average trajectories shown in Figures 5.3 and 5.4, PID performs consistently in general while there are fluctuations for MRACs. The large variance for MRACs happens during takeoff, which begins at 0 second and lasts for 4 seconds in Figure 5.3. To better understand the cause, the takeoff sections in Figure 5.2 are plotted on top of each other, and the vehicle altitude is shown in Figure 5.5. Clearly, the deviation is caused by the initial takeoff, where the vehicle response much faster than sequential takeoffs. This is caused by not resetting the adaptive gains properly after each touchdown, which is shown in Figure 5.6.

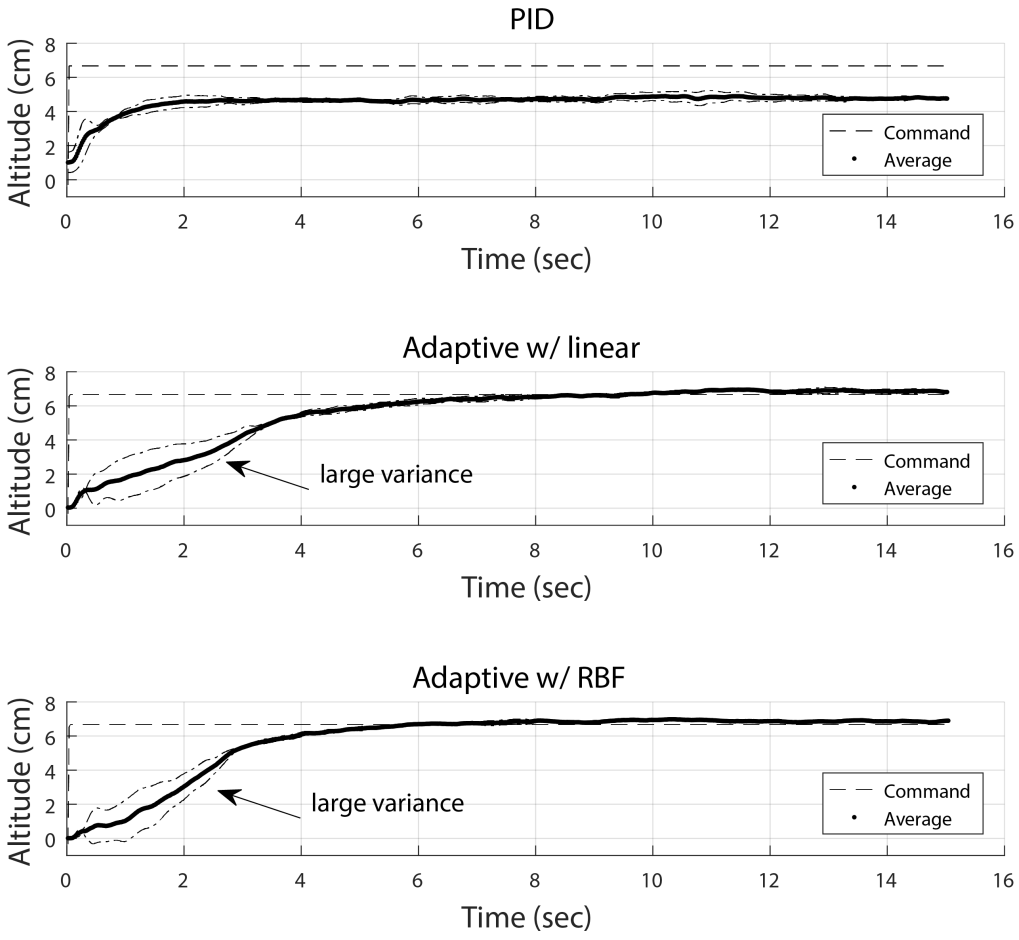

Figure 5.3: Average takeoff trajectories with different control algorithms (top: PID, middle: MRAC with linear model, bottom: MRAC with RBFs)

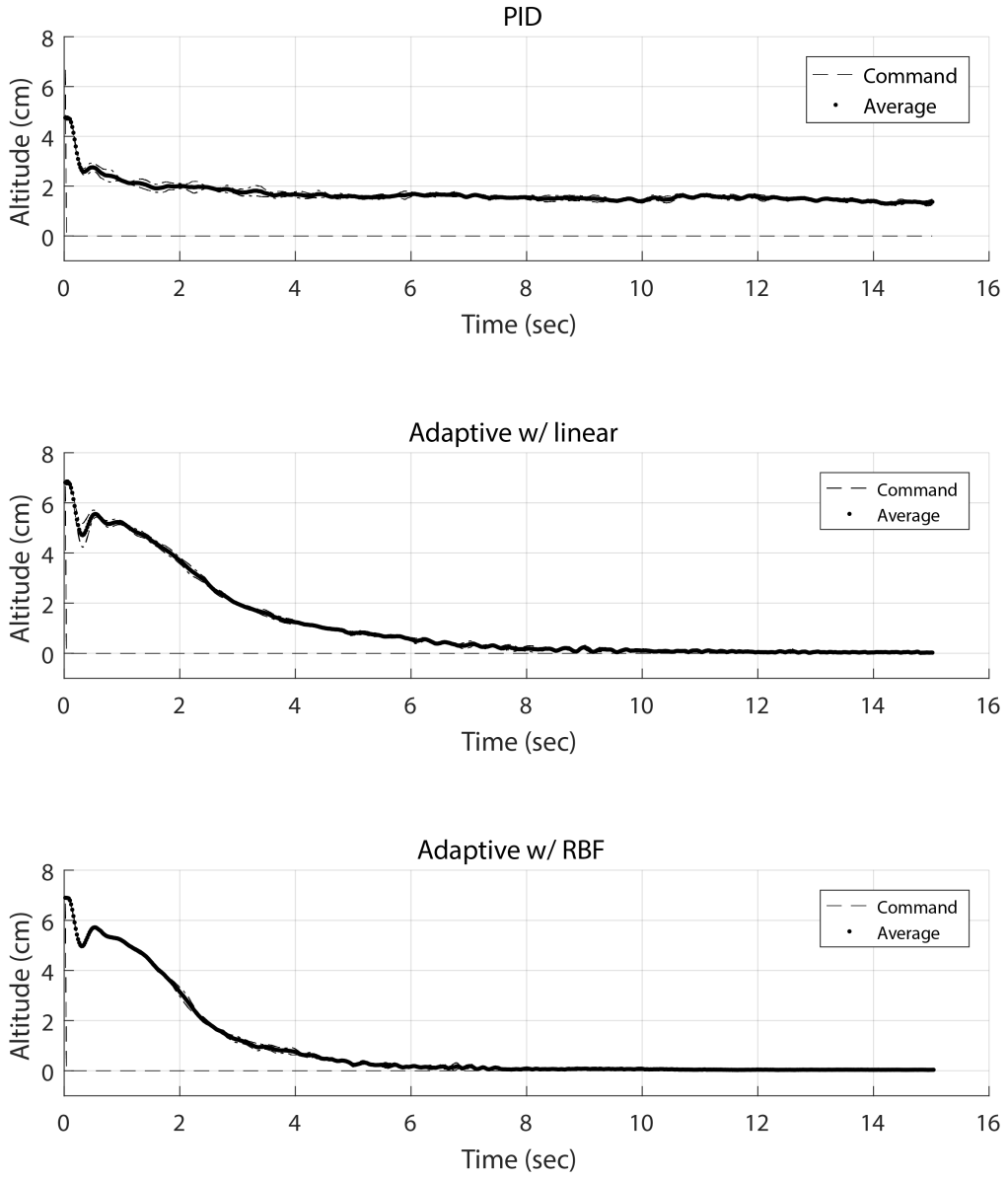

Figure 5.4: Average landing trajectories with different control algorithms (top: PID, middle: MRAC with linear model, bottom: MRAC with RBFs)

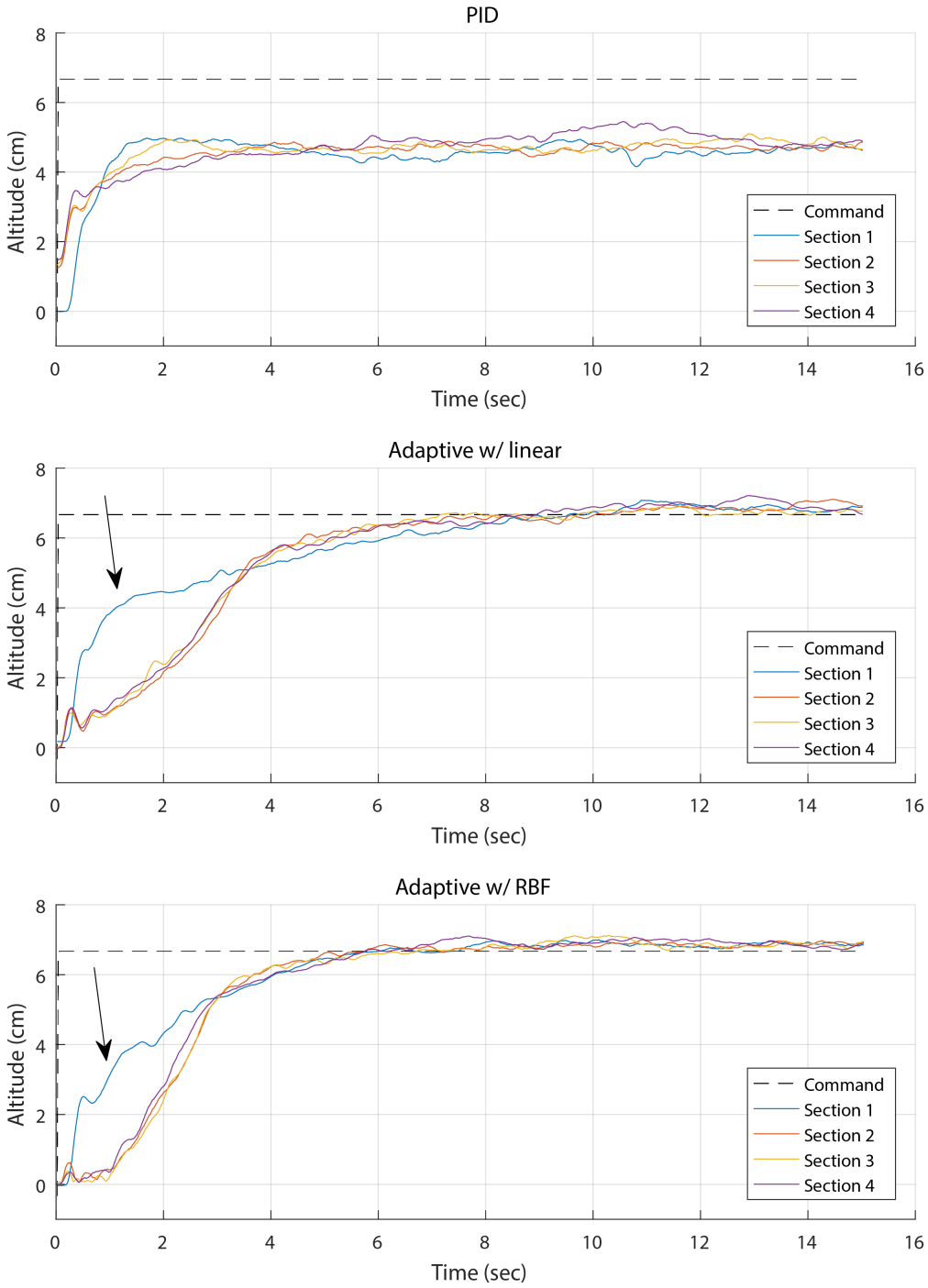

Figure 5.5: Takeoff trajectories with different control algorithms (top: PID, middle: MRAC with linear model, bottom: MRAC with RBFs)

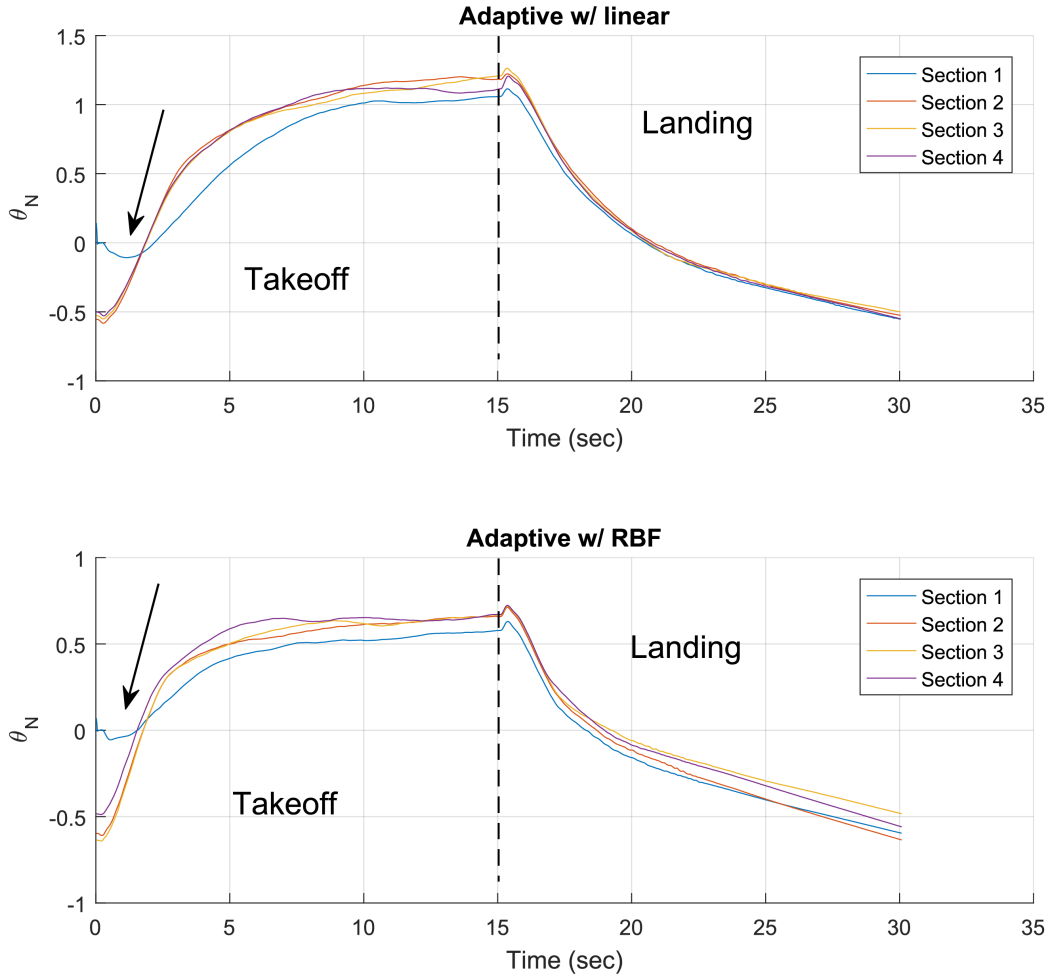

Figure 5.6: Adaptive gain bias term shows the adaptive controller is not properly reset after each touch down. (Note: only the bias term is shown to save space, but other gain terms also show a similar trend)

Chapter 6

Conclusion

In this thesis, a study of ground effects on a quadrotor aerial vehicle was presented. Experimental results showed that thrust generated by the rotors increased linearly as the vehicle got closer to the ground, which was different from the single rotor vehicle's ground effect model used in other research. Furthermore, a quadcopter experienced ground effects sooner than the single rotor model predicted, and the ground effect was weaker when the vehicle was at close proximity to the ground.

Then, a control architecture that utilizes a model reference adaptive controller was proposed to mitigate ground effect. Different position controllers were implemented on a physical system, which included a Crazyflie 2.0 and a computer. The controllers used in this study included a PID controller, a model reference adaptive controller (MRAC) with a linear ground effect model, and a MRAC that uses a set of radial basis functions (RBF), plus a bias term to approximate the ground effect function. The quadcopter took off and landed within the ground effect region multiple times, and its performances were compared. From the flight tests, the MRACs outperformed the PID controller. Among the MRACs, the one with RBF tracked the reference model better with less mean square error. It also responded quicker with less rise time. Furthermore, the MRAC with RBF performed more consistently compared to the one using linear ground effect model.

References

[1] S. Rodovnichenko. File:ekranoplan a-90 orlyonok - edit.jpg. [Online]. Available: https://commons.wikimedia.org/wiki/File:Ekranoplan_A-90_Orlyonok_-_edit.jpg

[2] Doodybutch. File:dji phantom 4 in flight march 2016.jpg. [Online]. Available: https://commons.wikimedia.org/wiki/File:DJI_Phantom_4_in_Flight_March_2016.jpg

[3] CNN. Watch dubai police test its latest gadget – a flying motorbike. [Online]. Available: http://www.cnn.com/2017/10/13/middleeast/dubai-police-hoversurf-flying-motorbike/index.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+rss%2Fedition_meast+%28RSS%3A+CNNi+-+Middle+East%29

[4] A. Momont. (2014, October) Ambulance drone. [Online]. Available: https://www.tudelft.nl/en/ide/research/research-labs/applied-labs/ambulance-drone/

[5] J. Navia, I. Mondragon, D. Patino, and J. Colorado, Eds., *Multispectral mapping in agriculture: terrain mosaic using an autonomous quadcopter UAV*, June 2016.

[6] C. PUB and U. De Federal Aviation Administration, *Helicopter Flying Handbook*. CreateSpace Independent Publishing Platform, 2013. [Online]. Available: https://books.google.com/books?id=wPEamwEACAAJ

[7] G. van Es, "Running out of runway analysis of 35 years of landing overrun accidents," National Aerospace Laboratory NLR, Executive summary NLR-TP-2005-498, September 2005.

[8] T. Sorensen, "Aviation accident final report," National Transportation Safety Board, non-fatal CEN16LA039, November 2016.

[9] B. C. Rayner, "Aviation accident final report," National Transportation Safety Board, non-fatal ERA16CA160, June 2016.

[10] M. F. Gallo, "Aviation accident final report," National Transportation Safety Board, fatal CEN11FA507, February 2013.

[11] A. Lemishko, "Aviation accident final report," National Transportation Safety Board, fatal ERA11FA272, August 2014.

[12] A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid, "Flight pid controller design for a uav quadrotor," *Scientific Research and Essays*, vol. 5, pp. 3660–3667, December 2010. [Online]. Available: http://www.academicjournals.org/SRE

[13] F. Ruggiero, M. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Perez, V. Lippiello, A. Ollero, and B. Siciliano, *A multilayer control for multirotor UAVs equipped with a servo robot arm.* IEEE, 06 2015, vol. 2015, pp. 4014–4020.

[14] C. E. Lin, T. Supsukbaworn, Y.-C. Huang, K.-T. Jhuang, C.-H. Li, C.-Y. Lin, and S.-Y. Lo, *Engine controller for hybrid powered dual quad-rotor system*, 11 2015, pp. 1513–1517.

[15] W. Giernacki, D. Horla, T. Sadalla, and J. P. Coelho, *Robust CDM and pole placement PID based thrust controllers for multirotor motor-rotor simplified model: The comparison in a context of using anti-windup compensation*, May 2016, pp. 1–5.

[16] F. Rinaldi, A. Gargioli, and F. Quagliotti, "Pid and lq regulation of a multirotor attitude: Mathematical modelling, simulations and experimental results," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 33–50, Jan 2014. [Online]. Available: https://doi.org/10.1007/s10846-013-9911-x

[17] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, Eds., *Dynamics modelling and linear control of quadcopter*, Nov 2016.

[18] J. Ferrin, R. Leishman, R. Beard, and T. McLain, Eds., *Differential Flatness Based Control of a Rotorcraft For Aggressive Maneuvers*, 09 2011.

[19] X. Chen and L. Wang, Eds., *Discrete-Time One-step Ahead Prediction Control (DOPC) of a Quadcopter UAV with Constraints*, 11 2016.

[20] K. Klausen, T. Fossen, and T. Johansen, "Nonlinear control with swing damping of a multirotor uav with suspended load," *Journal of Intelligent & Robotic Systems*, 03 2017.

[21] I. Palunko and R. Fierro, "Adaptive control of a quadrotor with dynamic changes in the center of gravity," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2626 – 2631, 2011, 18th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016440097

[22] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, Eds., *Robust Adaptive Control Design for Quadcopter Payload Add and Drop Applications*, 7 2015.

[23] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear h control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29 – 39, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109809004798

[24] E. C. Suicmez and A. T. Kutay, Eds., *Optimal Path Tracking Control of a Quadrotor UAV*, 5 2014.

[25] Z. X. Liu, C. Yuan, Y. M. Zhang, and J. Luo, Eds., *A Learning-Based Fuzzy LQR Control Scheme for Height Control of an Unmanned Quadrotor Helicopter*, 5 2014.

[26] R. W. STAUFENBIEL and U.-J. SCHLICHTING, "Stability of airplanes in ground effect," *Journal of Aircraft*, vol. 25, no. 4, pp. 289–294, 1988.

[27] D. P. Pulla, *A STUDY OF HELICOPTER AERODYNAMICS IN GROUND EFFECT*, 2006.

[28] W. Johnson, *Helicopter Theory*. Dover, 1980.

[29] S. BOUABDALLAH, *Design and Control of Quadrotors with Application to autonomous flying*, 2007.

[30] I. C. Cheesemen and W. E. Bennett, *The Effect of the Ground on a Helicopter Rotor in Forward Flight*, September 1955.

[31] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, *Influence of Aerodynamics and Proximity Effects in Quadrotor Flight*. Heidelberg: Springer International Publishing, 2013, pp. 289–302. [Online]. Available: https://doi.org/10.1007/978-3-319-00065-7_21

[32] S. Bouabdallah, *Design and Control of Quadrotors with Application to Autonomous Flying*, 2007.

[33] B. Landry, *Planning and Control for Quadrotor Flight through Cluttered Environments*, June 2015.

[34] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "Sympy: Symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017. [Online]. Available: https://doi.org/10.7717/peerj-cs.103

[35] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft Theory and Practice*. 41 William Street, Princeton, New Jersey 08540: Princeton University Press, 2012.

[36] E. Lavretsky and K. A. Wise, *Robust and Adaptive Control*, 1st ed. Springer, 2013.

[37] C. R. Gomes and J. A. C. C. Medeiros, "Neural network of gaussian radial basis functions applied to the problem of identification of nuclear accidents in a pwr nuclear power plant," *Annals of Nuclear Energy*, vol. 77, pp. 285–293, Mar. 2015. [Online]. Available: https://doi.org/10.1016/j.anucene.2014.10.001

[38] J.A.Rad, S.Kazem, and K.Parandc, "Optimal control of a parabolic distributed parameter system via radial basis functions," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, pp. 2559–2567, Aug. 2014. [Online]. Available: https://doi.org/10.1016/j.cnsns.2013.01.007