LIDAR-BASED SURFACE FOLLOWING CONTROL FOR UNMANNED AERIAL VEHICLES

by

Devin Simms

Bachelor of Engineering, Lakehead University (2014)

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Aerospace Engineering

Toronto, Ontario, Canada, 2017

LIDAR-BASED SURFACE FOLLOWING CONTROL FOR UNMANNED AERIAL VEHICLES

Devin Simms
Master of Applied Science, Aerospace Engineering, Ryerson University, Toronto (2017)

A simulation study is performed on a quadcopter which uses a LIDAR sensor to allow a quadcopter to navigate along and maintain a set distance from an unknown vertical surface. The dynamic equations of a quadcopter are linearized about the hovering equilibrium. For the purpose of design, all surfaces are assumed to be flat and any variations in shape are considered to be disturbances.

The design process begins with the development of a potential field control design to allow the quadcopter to autonomously follow a flat surface, while maintaining a desired distance from the surface. To allow the quadcopter to follow a curved surface, the potential field technique is modified to maintain the x_b axis parallel to the surface. Finally a wall following technique that directly uses the minimum range measurement to maintain the distance from the surface is developed.

To simulate the control designs, a non-linear quadcopter model is used along with a model of a 2D scanning LIDAR sensor. The potential field control technique tracks flat surfaces with no steady-state error, though when curved surface following is added, a tracking error problem occurs due to measurement noise. The wall following design proves to be the superior surface following technique with greater robustness to steady-state error and results in relatively small tracking errors when navigating sinusoidal surfaces and corners.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Anton de Ruiter for his patience, continuous support, motivation, and immense knowledge. Without his advice and thorough review, this thesis would not have been possible.
I would also like to thank Prof. Goetz Bramesfeld and Prof. Guangjun Liu for their comments during the defense of this thesis.
I would also like to thank Prof. Jason Etele, for helping with the project concept and providing the hardware, which helped shape the thesis to what it is today.
I would also like to thank both my parents and grandparents for their encouragement and limitless emotional support.
Finally, I owe a very special thank you to my girlfriend, Rena Li, for all her love and support.

Dedication

Dedicated
to my loving grandparents

for being
my biggest supporters.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Research in unmanned aerial vehicles (UAVs) has grown tremendously recently with increased demand for intelligent aircraft. UAVs find uses in everything from aerial photography, surveying, search and rescue, along with flight in any environment that is considered unsafe for a human pilot[26, 22, 28]. The quadcopter has posed a difficult control problem due to its requirement for complete independent speed control of four thrusters. The quadcopter is a six degree of freedom (DOF) aircraft that both generates lift and control from four on-board thrusters. Until recently, the quadcopter was viewed as an impractical aircraft since it is a highly non-linear system that relies on only four inputs to obtain six degrees of freedom. Due to the underactuated nature of the quadcopter, the rotational and translational dynamics of the system are coupled. Though with the advancements seen in control systems as well as both digital and power electronics, the quadcopter has moved to the forefront as a controls and robotics platform [14].

The act of navigating a mobile robot, such as a quadcopter, in an unknown and variable environment poses a very difficult design problem. The method of navigation often depends upon the information available about the surrounding environment, from such sources as sensors and previous knowledge. The purpose of this thesis is to design a controller that allows a quadcopter to navigate along an unknown surface, in an urban environment, using information obtained from a 2D scanning Light Detection and Ranging (LIDAR) sensor.

1.2 Literature Review

This section provides an overview of the current and relevant research that has been completed in the areas of path planning, obstacle avoidance, potential field control, scan matching, SLAM, and wall following.

1.2.1 Path Planning

UAVs are often used in complex environments and encounter many obstacles while navigating towards the desired goal. Various types of path planning techniques exist which are able to determine the best path given both static and dynamic obstacles, while also accounting for any constraints of the system. In Ref. [5] the authors use a modified central force optimization method to determine the optimal path for a UAV, while accounting for the length, height, and smoothness of the path.

UAVs often encounter dynamic obstacles such as other aircraft that need to be avoided. The authors of Ref. [41] developed a Geometric Reinforcement Learning algorithm which can be tuned using a parameter K to find the ideal balance between safety and economy. The proposed algorithm is able to avoid collisions with other UAV's and allows for information sharing between surrounding UAV's. This information sharing adds to each UAV's knowledge of surrounding obstacles, allowing for a more optimal solution.

When operating in dynamic environments the computation time may not allow for the calculation of a new optimal path for every new obstacle. To overcome this problem the authors of Ref. [31] develop an anytime algorithm. The quality of path determined by the anytime algorithm increases with the amount of computation time.

Since it is assumed that no prior information about the wall is available, a path planning technique is not suitable and a reactive approach is necessary such as the one developed in this thesis.

1.2.2 Obstacle Avoidance

A very important role of a path planning algorithm is to detect and avoid obstacles. Many different methods of obstacle avoidance are shown in literature. In Ref. [10] the authors use image processing techniques in OpenCV, on a base station computer, to process LIDAR data and determine an obstacle free path.

UAV's are often used in dynamic environments where a collision with a moving obstacle, such as anther aircraft, is a big concern. In Ref. [42] the authors develop a collision avoidance algorithm by determining a collision avoidance vector and then align the UAV's relative velocity vector with it.

In Ref. [12] the authors describe a reactive path planning algorithm, which is designed to avoid static obstacles. Unlike other techniques where a global map and prior knowledge of the surrounding are often required, the reactive path planning algorithm relies on only the field of view of the obstacle detection sensors to determine an obstacle free path.

1.2.3 Potential Field Control

Referring to the work completed in Ref. [25], with relatively low computational overhead, potential fields can be used for both path planning and obstacle avoidance. The use of potential fields for path control can be compared to the behavior of electric charges. The controlled system and any obstacles are treated as positive charges, while the control goal is treated as a negative charge. This results in the system being attracted to the control goal while simultaneously being repelled from any obstacles. In practice, this is completed by generating a potential field and then calculating its gradient. The gradient of the potential field is then used as the driving force for the system and is often used as velocity inputs to the controller.

Potential field control on its own provides a very convenient solution to the obstacle avoidance and path planning, though the basic algorithm poses some limitations. One of these limitations is the problem posed by local minima in the potential field. A local minima is often caused when multiple obstacle are in close proximity resulting in a situation where the controlled system is unable to pass between them. The problem of the local minimum has been well addressed in the literature with one such solution shown in Ref. [39], where a wall following technique is used to escape the local minimum.

Another limitation of potential field control is that it assumes the robot is able to move in any direction. This can cause problems for robots that have constraints such as wheeled robots. In Ref. [34] the authors propose combining potential field and kinematic control to account for the nonholonomic constraint. The authors prove through simulations that the proposed technique is able to account for the nonholonomic constraint while still avoiding obstacles, though further real-time implementation is required.

Potential field control has been shown to be successful in path planning and obstacle avoidance, though the trajectory it provides is not guaranteed to be the most optimal. In Ref. [4] the authors attempt to find a more optimal solution by including an additional control force and using optimal control to solve for a more optimal trajectory. The authors apply the new technique to the path planning of a UAV and the results show an improvement compared to the traditional method of potential field control.

1.2.4 Scan Matching

Given range measurements from LIDAR a convenient method to obtain the pose of a vehicle given two sets of LIDAR scan measurements, a reference scan and a current scan, is used to determine the rotational and translational movements which makes the current LIDAR scan measurements match those of a reference scan. In Ref. [9] the authors use a polar scan matching (PSM) approach to match LIDAR scans. The results of PSM approach are compared to those of an iterated closest point (ICP) algorithm. Through simulation, PSM is shown to obtain very similar results to the ICP algorithm with less than one third the number of iterations.

In Ref [11] a hybrid design is proposed, in an effort to increase efficiency and robustness of the scan matching algorithm. The design combines the efficiency of the feature-based scan matching method and robustness, to feature lacking environments, of the Iterative Closest Point (ICP) based scan matching method.

In Ref. [13] the authors propose new fast scanning technique using corresponding vector sampling and consensus (CVSAC). The method is based on random sampling and consensus (RANSAC) and iterative closest point (ICP) and named FAST CVSAC. Another fast scanning algorithm is developed in [24] where the authors use characteristic points of curvature functions to allow for efficient alignment between two scans.

The technique of scan matching may also be applied to image data obtained from a camera. In Ref. [20] the image frames from an omni-directional camera, mounted on a mobile robot, are compared to determine the pose of the robot. In Ref. [35] the authors merge LIDAR scan matching and omni-directional vision to form a more robust mapping algorithm.

A vision-based scan matching technique was not considered for this project due to their relatively large computational demand and typically low computing power of most quadcopters.

1.2.5 Simultaneous Location And Mapping (SLAM)

When prior knowledge is not known about an environment and pose information is not easily obtainable, such as in GPS denied environments, SLAM provides a very useful solution to the navigation problem. Through the use of state estimation, SLAM is able to both generate an estimate of pose of the vehicle while also generating a map of the surrounding environment. The key component to the SLAM algorithm is the state estimation filter, which is typically a Kalman filter, extended Kalman filter (EKF), particle filter, or information filter[29]. The state estimation filter is usually used to propagate both the past pose of the vehicle along with the pose of surrounding landmarks using a mathematical model of the vehicle[40].

Many variations of the SLAM algorithm can be found in the literature with application from ground robots to UAV's. In Ref. [17] a light and fast SLAM algorithm is developed for mobile robotics in indoor environments. The light weight algorithm relies on line segments that are extracted from LIDAR readings to complete a fundamental map of the environment. The orthogonal nature of most indoor environments is used to allow for special treatment of parallel and perpendicular lines to reduce memory usage. The design results in fast performance with a relatively low amount of memory usage.

Typically SLAM algorithms rely on a dynamic model of the vehicle, though in Ref. [36] the author makes use of a differential model, where the robots motion is approximated by straight line segments between each data sample. The differential model is combined with an extended Kalman filter to provide a more robust algorithm then that of typical dynamic model design.

An EKF-SLAM design applied to the navigation of UAV's is shown in Ref. [6], where an extended Kalman filter based SLAM algorithm is used to estimate the position of a quadcopter. The design is simulated in a 3D environment where the quadcopter is required to navigate relative to four landmarks.

A common problem with the SLAM algorithm is the large computational complexity that results as the numbers of landmarks increases. In Ref. [33] and Ref.[8] the authors use smoothing and information filtering techniques, in place of the commonly used EKF, in attempt to reduce the computational complexity of the algorithm. The proposed techniques result in a significant computational reduction while resulting in comparable results to that of the traditional EKF-SLAM technique.

In reference [3] a vision-based SLAM algorithm is developed which uses only a single monocular camera. The proposed algorithm relies on the architectural orthogonality of the surrounding environment to allow for the calculation of absolute range and bearing information without the need for range sensors. A stereo vision-based SLAM algorithm is shown in Ref. [37]. The algorithm was tested on a mobile robot platform and proved to be accurate even when subjected to large translations and rotations.

1.2.6 Wall Following

The field of wall-following control has seen a lot of attention in the past with the bulk of the work focusing on ground mobile robotic systems, with little work being completed with UAVs. Other than the work completed in Ref.[23], where a wall following algorithm is applied to an aerial vehicle, little research has been documented in the area of wall following for aerial vehicles. In Ref. [23], the author develops a simple wall following algorithm for a quadcopter with four proximity sensors, one on each side, which allows the aircraft to navigate around indoor in an GPS denied, environment.

Wall following algorithms often encounter problems dealing with situations where measurement data is noisy, in instances such as the robot passing a doorway. In Ref. [7], an observer is used to process the measurement data and help reduce the impact of noise and gaps in measurement data on the wall following performance of the robot.

The idea of a hybrid control system is proposed in Ref.[19], where a continuous controller is used to control both the linear and angular velocities. A discrete controller is used to make high level decisions and is responsible for determining the correct velocities needed to ensure the robot maintains the desired distance for the wall.

Fuzzy controllers have been proven to be an effective means of controlling a wall following robot in the work completed in Ref. [21] and Ref. [15]. The authors use fuzzy logic controllers, provided with a rule bases to decide on motion commands for the robot based on range measurements that are provided as inputs. Learning control techniques have been applied to the wall following problem by tuning controllers gains to improve the robots wall following. The work completed in Ref. [2] shows the development of a fuzzy logic controller to allow a robot to follow a wall and then, a genetic program is used to fine tune the rule base to allow for better performance.

Non-linear control techniques have also been applied to the wall following control problem in the work completed in Ref. [7]. The author selects a Lyapunov function candidate and proceeds to develop both a full state feedback and observer based controllers based on it.

The work shown in Ref. [38] proposes a simplified control scheme in which only the minimum range measurement is needed. The minimum range measurement provides both the distance to the wall, as well as the heading, which are both needed to control the mobile robot.

1.3 Objective, Approach and Thesis Structure

The control goal for this thesis is to allow a UAV to navigate along and remain at a constant distance from an unknown vertical surface.

Given the methods presented in section 1.2 both potential field control and wall following were examined, with the finial design for this project using a wall following technique. To facilitate wall following the control law will only receive the magnitude and angle of the minimum range measurement for the LIDAR sensor, similar to the work completed in Ref. [38]. Unlike the controller in Ref. [38], the proposed wall following controller will be considered for a UAV.

Chapter 2 contains an overview of the dynamics of a quadcopter along with both the non-linear and linear dynamics models used for design and simulation. Chapter 3 details the developed wall following algorithms. The simulation process and resulting performance of the various control designs are shown in Chapter 4. Leaving, conclusions and recommendations for future work for Chapter 5.

Chapter 2

Problem Formulation

2.1 Non-linear Model

Referring to the work by [16] this section shows the derivation of a non-linear dynamic model for the quadcopter system. Referring to Figure 2.1, two reference frames are used to define the dynamics of the quadcopter. The first is an inertial frame, $\{x_i,\, y_i,\, z_i\}$, with an origin fixed to the Earth at the position of the center of mass of the quadcopter at time t_0. The second is a body fixed frame, $\{x_b,\, y_b,\, z_b\}$, with the origin fixed at the center of mass of the quadcopter.

Rotors 1 and 3 rotate counter clockwise with speeds Ω_1 and Ω_3, respectively. Rotors 2 and 4 rotate clockwise with speeds Ω_2 and Ω_4, respectively. The thrust generated by each rotor is proportional to the square of the rotor speed, as shown below.

$$T_i = K_t \Omega_i^2, \tag{2.1}$$

where K_t is the thrust constant of the respective rotor and the total thrust produced along the z_b axis is the sum of the four thrusts, T_i, provided by the thrusters as shown below.

$$u_1 = \sum_{i=1}^{4} T_i. \tag{2.2}$$

The torques about the x_b and y_b axes are shown in Equations 2.3 and 2.4 respectively,

$$u_2 = K_t L(\Omega_4^2 - \Omega_2^2), \tag{2.3}$$

$$u_3 = K_t L(\Omega_3^2 - \Omega_1^2), \tag{2.4}$$

where L is the distance from the aircrafts center of mass to the center of each of the 4 rotors. The torque generated by each rotor about the z_b is proportional to the square of the rotor's speed. The total torque

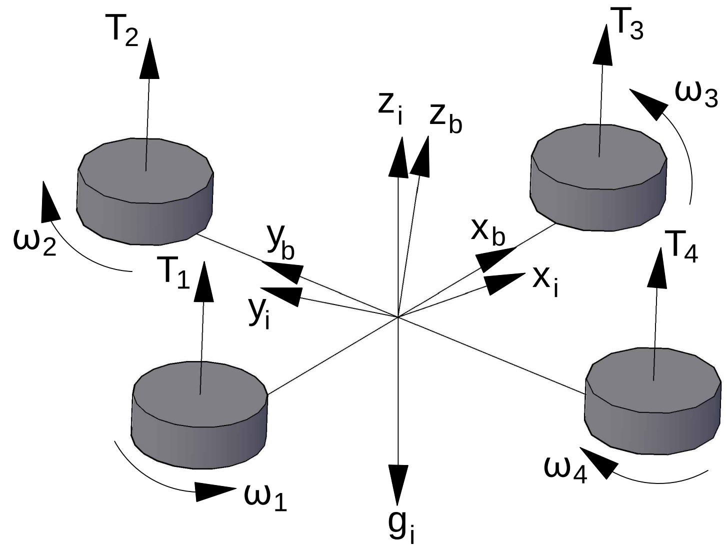

Figure 2.1: A diagram of the quadcopter system showing the forces acting on the aircraft along with both the inertial and body reference frames.

generated about z_b can be found using Equation 2.5.

$$u_4 = K_{tor}(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2), \tag{2.5}$$

where K_{tor} is the torque constant of the rotors. Equations 2.2 through 2.5 can be combined in a single matrix calculation as shown in Equation 2.6.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K_t & K_t & K_t & K_t \\ 0 & -LK_t & 0 & LK_t \\ -LK_t & 0 & LK_t & 0 \\ K_{tor} & -K_{tor} & K_{tor} & -K_{tor} \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{2.6}$$

Using Equation 2.1, Equation 2.6 can be modified and expressed using the individual rotor thrusts as shown in Equation 2.7, below.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -L & 0 & L \\ -L & 0 & L & 0 \\ \frac{K_{tor}}{K_t} & -\frac{K_{tor}}{K_t} & \frac{K_{tor}}{K_t} & -\frac{K_{tor}}{K_t} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}. \tag{2.7}$$

The translational motion of the quadcopter is described in Equation 2.8.

$$m\ddot{\mathbf{r}}_{\mathbf{i}} = m\mathbf{g}_{\mathbf{i}} + \mathbf{C_{bi}}^T\mathbf{t_b}, \tag{2.8}$$

where

$$\mathbf{t_b} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \tag{2.9}$$

$$\mathbf{g_i} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}. \tag{2.10}$$

In Equations 2.8-2.10, m is the mass of the quadcopter, $\mathbf{r_i} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the position vector of the quadcopter's center of mass in the inertial frame, $\mathbf{t_b}$ is the thrust vector in the body frame, with T representing the total trust produced and $\mathbf{g_i}$ is the gravitational vector in the inertial frame. $\mathbf{C_{bi}}$ is the coordinate transformation matrix relating inertial to body coordinates. It is parameterized by a 3-2-1 Euler sequence, with the rotation angles roll, pitch and yaw represented by ϕ, θ and ψ, respectively.

$$\mathbf{C_{bi}} = \mathbf{C_x}(\phi)\mathbf{C_y}(\theta)\mathbf{C_z}(\psi), \tag{2.11}$$

where

$$\mathbf{C_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \tag{2.12}$$

$$\mathbf{C_y}(\theta) = \begin{bmatrix} \cos\theta & 1 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.13}$$

$$\mathbf{C_z}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.14}$$

$$\mathbf{C_{bi}} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}. \tag{2.15}$$

The rotational dynamics of the quadcopter are expressed in Equation 2.16.

$$\mathbf{I_b}\dot{\boldsymbol{\omega}}_{\boldsymbol{b}}^{\boldsymbol{bi}} + \boldsymbol{\omega}_{\boldsymbol{b}}^{\boldsymbol{bi}^x}\mathbf{I_b}\boldsymbol{\omega}_{\boldsymbol{b}}^{\boldsymbol{bi}} = \mathbf{j_b}, \tag{2.16}$$

where $\mathbf{I_b}$ is the inertia matrix expressed in the body frame, $\mathbf{j_b}$ is the torque vector in the body frame and $\boldsymbol{\omega}_{\boldsymbol{b}}^{\boldsymbol{bi}} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ is the angular velocity of the body frame with respect to the inertial frame, expressed in the body frame, with its cross matrix shown in Equation 2.17.

$$\boldsymbol{\omega_b^{bi}}^x = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2.17}$$

The rotational kinematics are given by Poisson's Equation as shown in Equations 2.18.

$$\dot{\mathbf{C}}_{\mathbf{bi}} = -\boldsymbol{\omega_b^{bi}}^x \mathbf{C}_{\mathbf{bi}}. \tag{2.18}$$

After substituting 2.15 into 2.18 the following expression can be found for angular velocity in terms of Euler angle rates.

$$\boldsymbol{\omega_b^{bi}} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.19}$$

After assuming both small roll and pitch angles along with small angular rates, Equation 2.19 can be reduced to the form shown below.

$$\boldsymbol{\omega_b^{bi}} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.20}$$

Due to the symmetry of the aircraft the body fixed frame is assume to be a principal axes frame allowing for the following form of the inertia matrix to be used.

$$\mathbf{I_b} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{2.21}$$

The planar symmetry of the aircraft also allows for the assumption $I_{xx} = I_{yy}$ to be applied.

Applying the approximations in 2.19 and 2.21 to Euler equations in 2.16 leads to the following rotational dynamic equations, where $\mathbf{j_b} = \begin{bmatrix} u_2 & u_3 & u_4 \end{bmatrix}^T$.

$$I_{xx}\ddot{\phi} + (I_{zz} - I_{yy})\dot{\theta}\dot{\psi} = u_2, \tag{2.22}$$

$$I_{yy}\ddot{\theta} + (I_{xx} - I_{zz})\dot{\phi}\dot{\psi} = u_3, \tag{2.23}$$

$$I_{zz}\ddot{\psi} + (I_{yy} - I_{xx})\dot{\phi}\dot{\theta} = u_4. \tag{2.24}$$

Equations 2.2 and 2.8 lead to the following form of translational dynamic equations.

$$m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta)u_1, \tag{2.25}$$

$$m\ddot{y} = (-\cos\psi\sin\phi + \sin\psi\cos\phi\sin\theta)u_1, \tag{2.26}$$

$$m\ddot{z} = (\cos\phi\cos\theta)u_1 - mg. \tag{2.27}$$

Rearranging Equations 2.22 through 2.27 lead to the following non-linear state-space model.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \frac{u_2-(I_{zz}-I_{yy})\dot{\theta}\dot{\psi}}{I_{xx}} \\ \frac{u_3-(I_{xx}-I_{zz})\dot{\phi}\dot{\psi}}{I_{yy}} \\ \frac{u_4}{I_{zz}} \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ (\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta)\frac{u_1}{m} \\ (-\cos\psi\sin\phi + \sin\psi\cos\phi\sin\theta)\frac{u_1}{m} \\ \cos\phi\cos\theta\frac{u_1}{m} - g \end{bmatrix}, \tag{2.28}$$

$$\mathbf{y_m} = \begin{bmatrix}\phi & \theta & \psi & x & y & z\end{bmatrix}^T, \tag{2.29}$$

where

$$\mathbf{x} = \begin{bmatrix}\phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & x & y & z & \dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T, \tag{2.30}$$

$\mathbf{y_m}$ is a vector containing the measured states, which in this case are the Euler angles and position.

2.2 Linearization

In this section Equations 2.30 through 2.29 will be linearized about the hovering equilibrium point shown in Equation 2.31.

$$\begin{bmatrix}\phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & \dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T. \tag{2.31}$$

The linearization uses the following approximation, where $\mathbf{x} = \mathbf{x_0} + \boldsymbol{\delta}\mathbf{x}$ and $\mathbf{u} = \mathbf{u_0} + \boldsymbol{\delta}\mathbf{u}$.

$$f(\mathbf{x},\mathbf{u}) \approx f(\mathbf{x_0},\mathbf{u_0}) + \frac{\boldsymbol{\partial} f(\mathbf{x_0},\mathbf{u_0})}{\boldsymbol{\partial}\mathbf{x}}\boldsymbol{\delta}\mathbf{x} + \frac{\boldsymbol{\partial} f(\mathbf{x_0},\mathbf{u_0})}{\boldsymbol{\partial}\mathbf{u}}\boldsymbol{\delta}\mathbf{u}. \tag{2.32}$$

For an equilibrium at $\mathbf{x_0}$, as shown in Equation 2.31, $\mathbf{u_0}$ needs to be found to satisfy $f(\mathbf{x_0},\mathbf{u_0}) = 0$. This leads to the following form for the linearized equations.

$$\boldsymbol{\delta}\dot{\mathbf{x}} = \frac{\boldsymbol{\partial} f(\mathbf{x_0},\mathbf{u_0})}{\boldsymbol{\partial}\mathbf{x}}\boldsymbol{\delta}\mathbf{x} + \frac{\boldsymbol{\partial} f(\mathbf{x_0},\mathbf{u_0})}{\boldsymbol{\partial}\mathbf{u}}\boldsymbol{\delta}\mathbf{u}. \tag{2.33}$$

Since it is desired to have the quadcopter hover at any location, only a partial linearization will be completed by splitting up the equations. The state vector is partitioned as

$$\mathbf{x_1} = \begin{bmatrix}\phi & \theta & \psi & \dot\phi & \dot\theta & \dot\psi & \dot x & \dot y & \dot z\end{bmatrix}^T, \tag{2.34}$$

$$\mathbf{x_2} = \begin{bmatrix}x & y & z\end{bmatrix}^T. \tag{2.35}$$

Defining the partial control input

$$\bar{\mathbf{u}}_\mathbf{2} = \begin{bmatrix}u_2 & u_3 & u_4\end{bmatrix}^T. \tag{2.36}$$

The system dynamics in 2.28 are partitioned as

$$\dot{\mathbf{x}}_\mathbf{1} = f_1(\mathbf{x_1},\bar{\mathbf{u}}_\mathbf{2}) = \begin{bmatrix}\dot\phi \\ \dot\theta \\ \dot\psi \\ \frac{u_2-(I_{zz}-I_{yy})\dot\theta\dot\psi}{I_{xx}} \\ \frac{u_3-(I_{xx}-I_{zz})\dot\phi\dot\psi}{I_{yy}} \\ \frac{u_4}{I_{zz}} \\ \dot x \\ \dot y \\ \dot z\end{bmatrix}, \tag{2.37}$$

$$\dot{\mathbf{x}}_\mathbf{2} = f_2(\mathbf{x_1},u_1) = \begin{bmatrix}(\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta)\frac{u_1}{m} \\ (-\cos\psi\sin\phi + \sin\psi\cos\phi\sin\theta)\frac{u_1}{m} \\ \cos\phi\cos\theta\frac{u_1}{m} - g\end{bmatrix}. \tag{2.38}$$

For $\mathbf{x_1} = 0$ to be an equilibrium point the two conditions, $f_1(\mathbf{x_1},\bar{\mathbf{u}}_{\mathbf{2,0}}) = 0$ and $f_2(\mathbf{x_1},u_{1,0}) = 0$, must be satisfied, which leads to $\bar{\mathbf{u}}_{\mathbf{2,0}} = 0$ and $u_{1,0} = mg$. Consequently, $u_1 = mg + \delta u_1$ and $\mathbf{u_2} = \boldsymbol{\delta}\mathbf{u_2}$, leading to the following linearized equations.

$$\dot{\mathbf{x}}_\mathbf{1} = \frac{\boldsymbol{\partial} f_1(\mathbf{0},\mathbf{0})}{\boldsymbol{\partial}\mathbf{x_1}}\mathbf{x_1} + \frac{\boldsymbol{\partial} f_1(\mathbf{0},\mathbf{0})}{\boldsymbol{\partial}\bar{\mathbf{u}}_\mathbf{2}}\boldsymbol{\delta}\bar{\mathbf{u}}_\mathbf{2}, \tag{2.39}$$

$$\dot{\mathbf{x}}_{\mathbf{2}} = \frac{\boldsymbol{\partial} f_2(\mathbf{0}, mg)}{\boldsymbol{\partial}\mathbf{x}_{\mathbf{1}}}\mathbf{x}_{\mathbf{1}} + \frac{\boldsymbol{\partial} f_2(\mathbf{0}, mg)}{\partial u_1}\delta u_1, \tag{2.40}$$

where

$$\frac{\boldsymbol{\partial} f_1(\mathbf{0}, \mathbf{0})}{\boldsymbol{\partial}\mathbf{x}_{\mathbf{1}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.41}$$

$$\frac{\boldsymbol{\partial} f_1(\mathbf{0}, \mathbf{0})}{\boldsymbol{\partial}\bar{\mathbf{u}}_{\mathbf{2}}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.42}$$

$$\frac{\boldsymbol{\partial} f_2(\mathbf{0}, mg)}{\boldsymbol{\partial}\mathbf{x}_{\mathbf{1}}} = \begin{bmatrix} 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.43}$$

$$\frac{\boldsymbol{\partial} f_2(\mathbf{0}, mg)}{\boldsymbol{\partial}\mathbf{u}_{\mathbf{1}}} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m} \end{bmatrix}. \tag{2.44}$$

Combining Equations 2.29 and 2.41 through 2.44 results in the following state-space equations.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\delta}\mathbf{u}, \tag{2.45}$$

$$\mathbf{y}_{\mathbf{m}} = \mathbf{C}\mathbf{x} + \mathbf{D}\boldsymbol{\delta}\mathbf{u}, \tag{2.46}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.47}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \end{bmatrix}, \tag{2.48}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{2.49}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.50}$$

$$\boldsymbol{\delta}\mathbf{u} = \begin{bmatrix} \delta u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T. \tag{2.51}$$

Equations 2.45 through 2.51 above use the total thrust and torques as the control input. Using Equation 2.7, the individual thrusts, $\boldsymbol{\tau} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 \end{bmatrix}^T$, can be used as the control input by using a transformation matrix, $\mathbf{K}$.

$$\mathbf{K} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -L & 0 & L \\ -L & 0 & L & 0 \\ \frac{K_{tor}}{K_t} & \frac{-K_{tor}}{K_t} & \frac{K_{tor}}{K_t} & \frac{-K_{tor}}{K_t} \end{bmatrix}. \tag{2.52}$$

Defining

$$\boldsymbol{\tau} = \boldsymbol{\tau_0} + \boldsymbol{\delta\tau}, \tag{2.53}$$

where,

$$\boldsymbol{\tau_0} = \mathbf{K}^{-1}\mathbf{u_0} = \begin{bmatrix} \frac{mg}{4} & \frac{mg}{4} & \frac{mg}{4} & \frac{mg}{4} \end{bmatrix}^T, \tag{2.54}$$

$$\boldsymbol{\delta\tau} = \mathbf{K}^{-1}\boldsymbol{\delta}\mathbf{u}. \tag{2.55}$$

Applying the transformation matrix, from 2.52, and using 2.7, results in the following inputs.

$$\mathbf{u} = \mathbf{K}\boldsymbol{\tau_0} + \mathbf{K}\boldsymbol{\delta\tau}. \tag{2.56}$$

In terms of the new control input $\boldsymbol{\delta\tau}$, the linearized Equations in 2.45 and 2.46 become

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \hat{\mathbf{B}}\boldsymbol{\delta\tau}, \tag{2.57}$$

$$\mathbf{y_m} = \mathbf{C}\mathbf{x} + \hat{\mathbf{D}}\boldsymbol{\delta\tau}, \tag{2.58}$$

where

$$\hat{\mathbf{B}} = \mathbf{BK} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{-L}{I_{xx}} & 0 & \frac{L}{I_{xx}} \\ \frac{-L}{I_{yy}} & 0 & \frac{L}{I_{yy}} & 0 \\ \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} & \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix}, \tag{2.59}$$

$$\hat{\mathbf{D}} = \mathbf{DK} = \mathbf{0}. \tag{2.60}$$

2.3 LIDAR Model

2.3.1 2D Model

The LIDAR model used in this thesis is assumed to be a 2D sensor which provides a series of planner range measurements, $\boldsymbol{\Gamma_j}$, with corresponding angles, $\boldsymbol{\Theta_j}$. The angles of each measurement are measured positive counterclockwise with respect to the front of the LIDAR sensor.

$$\Theta_j = (j - \rho)r, \tag{2.61}$$

for distinct $j \in \{1, \ldots, p\}$, where j is the index of the measurement, p is the total number of measurements, ρ is the index of the measurement at the front of the sensor, and r is the angular resolution of the LIDAR measurements.

For the purposes of this thesis, the LIDAR sensor is fixed to a quadcopter, where small angles are assumed for roll and pitch. The roll and pitch angles are approximated by zero and the surface is assumed to be non varying with altitude. With these assumptions the LIDAR can be constrained to the $x_i y_i$ plane, with only three degrees of motion. The sensor is assumed to have translational motion along the x_i and y_i axes, along with rotational motion about the z_i axis.

With this LIDAR model a vertical surface can be approximated by a line, representing the intersection of the the $x_i y_i$ plane with the vertical surface. The formula for the line is given in the following equation.

$$y_s = f(x_s), \tag{2.62}$$

where x_s and y_s and the horizontal components of position in the inertial frame. Each LIDAR mea-

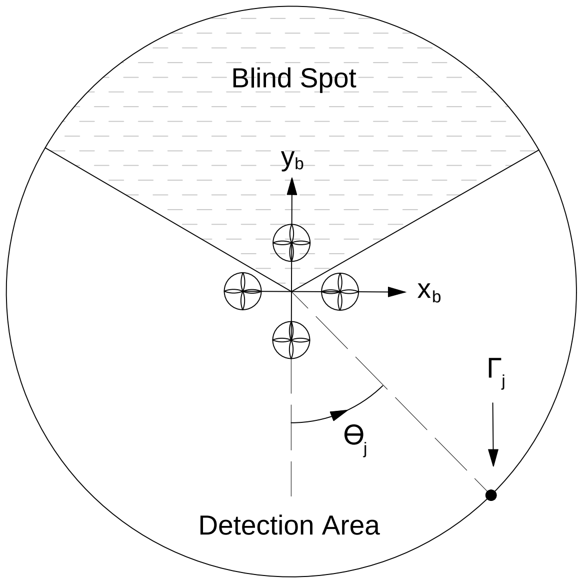

Figure 2.2: A diagram illustrating the LIDAR measurements.

surement is determined by finding the intercept of the LIDAR measurement, which is represented as a straight line, with the function, shown in Equation 2.62, that defines the surface. The intercept can be found using many different techniques, though in this thesis an iterative method is chosen. The algorithm iteratively increases the scanning range of the LIDAR until an intercept is found, as shown in the equations below.

$$x_k = x + (g_{k-1} + \Delta g)\cos(\Theta_j + \psi - \frac{\pi}{2}), \tag{2.63}$$

$$y_k = y + (g_{k-1} + \Delta g)\sin(\Theta_j + \psi - \frac{\pi}{2}), \tag{2.64}$$

for distinct $k \in \{1, \ldots, \frac{\gamma}{\Delta g}\}$, where k is the iterative index, g_{k-1} is the past iterative range, γ is the maximum range measurement, x_k and y_k represent the current horizontal position, in the inertial frame, of the iterative range, x and y represent the horizontal position, in the inertial frame, of the LIDAR sensor, Δg is the resolution of the algorithm, ψ is the yaw angle of the quadcopter and the $\frac{\pi}{2}$ is subtracted to align the front of LIDAR sensor with the y_b axis, as shown in Figure 2.2.

An intercept is found, when the following equation holds true.

$$\frac{f(x_k) - y_k}{f(x_{k-1}) - y_{k-1}} < 0. \tag{2.65}$$

When an intercept is found the range measurement can be found as follows.

$$\Gamma_j = g_{k-1} + \nu_j, \tag{2.66}$$

where ν is the measurement noise which is assumed to have a Gaussian distribution.

If no intercept is found then the algorithm will return Not-a-Number (NaN) for the current range measurement.

2.3.2 3D Model

The 2D model shown in section 2.3.1 can be expanded to a full 3D model using transformation matrices, as shown below. Referring to [18] a transformation matrix, $\mathbf{T_i}$, is defined as follows.

$$\mathbf{T_i} = \begin{bmatrix} \mathbf{C_i} & \boldsymbol{\chi}_i \\ 0 & 1 \end{bmatrix}, \tag{2.67}$$

where $\mathbf{C_i}$ is a rotation matrix and $\boldsymbol{\chi}_i$ is a translation vector.

Referring to Equation 2.15, the transformation matrix used to rotate from inertial to body coordinates is given by

$$\mathbf{T_0} = \begin{bmatrix} \mathbf{C_{bi}}^T & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.68}$$

Referring to Equation 2.14 the next transformation matrix, used to rotate about the z_i axis, is shown below.

$$\mathbf{T_1} = \begin{bmatrix} \mathbf{R_z}(\alpha) & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.69}$$

where

$$\alpha = \frac{\pi}{2} - \Theta_j. \tag{2.70}$$

The finial transformation matrix is given below.

$$\mathbf{T_2} = \begin{bmatrix} \mathbf{I} & \boldsymbol{\chi}_2 \\ 0 & 1 \end{bmatrix}, \tag{2.71}$$

where

$$\boldsymbol{\chi}_2 = \begin{bmatrix} (g_{k-1} + \Delta g) & 0 & 0 \end{bmatrix}^T. \tag{2.72}$$

The 3D LIDAR model can be calculated using the same procedure shown in section 2.3.1 by replacing

Equations 2.62, 2.63, 2.64, and 2.65 with Equations 2.73 through 2.75.

$$\begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} = \mathbf{T_0 T_1 T_2} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}, \tag{2.73}$$

$$y_s = f(x_s, z_s), \tag{2.74}$$

$$\frac{f(x_k, z_k) - y_k}{f(x_{k-1}, z_{k-1}) - y_{k-1}} < 0, \tag{2.75}$$

where z_k is the position, along the z_i axis, of the current iterative range, z is the position, along the z_i axis, of the LIDAR sensor, and z_s is the position along the z_i axis.

2.4 Problem Summary

The control objective of this thesis is to design a control law to allow a quadcopter aircraft to track an unknown vertical surface. The quadcopter must also maintain a specified distance from the unknown surface.

Chapter 3

Control Design

3.1 Overview

This chapter shows the control designs considered. The final wall following controller, shown in section 3.3, is designed allowing the quadcopter to maintain a desired horizontal velocity parallel to the surface, while maintaining a desired distance from the surface at a desired altitude. A block diagram of the control design can be seen in Figure 3.1.

To facilitate controlled flight of the quadcopter, attitude, angular and translational velocities and altitude measurements are assumed available. In addition, a LIDAR sensor provides the range measurements to the surrounding environment which are processed to determine the relative yaw angle and minimum distance to an unknown vertical surface.

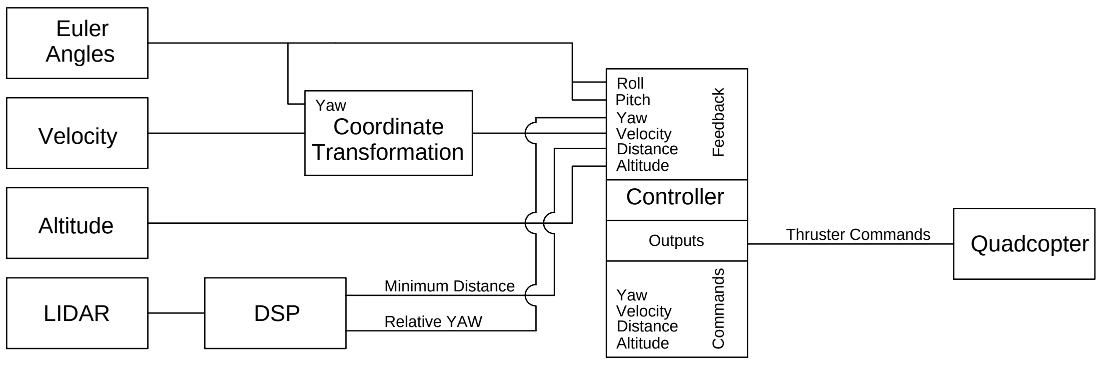

Figure 3.1: A schematic diagram of the quadcopter controller.

3.2 Potential Field Control

Potential field control was explored as a method to allow the quadcopter to autonomously track a vertical surface. Referring to the work completed in Ref. [25], with relatively low computational overhead, potential fields can be used for both path following and obstacle avoidance. The use of potential fields for path following can easily be compared to the behavior of electric charges. The controlled system and any obstacles are treated as positive charges, while the control goal is treated as a negative charge. This results in the quadcopter being attracted to the control goal while simultaneously being repelled from any obstacles. In practice, this is completed by generating a potential field and then calculating its gradient. The gradient of the potential field is then used as an input to the controller.

Equations 3.1 and 3.2 show examples of basic functions that can be used to calculate the individual potential fields for the control goal and obstacles, respectively.

$$f_{goal}(x,y) = (x_0 - x_1)^2 + (y_0 - y_1)^2, \tag{3.1}$$

$$f_{obstacle}(x,y) = \frac{1}{\sqrt{(x_0 - x_2)^2 + (y_0 - y_2)^2}}, \tag{3.2}$$

where (x_0, y_0), (x_1, y_1), and (x_2, y_2) are the coordinates corresponding to the position of the quadcopter, the control goal and obstacles, respectively. The total potential field can be calculated using Equation 3.3.

$$f_{total}(x,y) = \sum f_{goal}(x,y) + \sum f_{obstacle}(x,y). \tag{3.3}$$

The total potential field can then be used to calculate the reference velocities along both the the x_b and y_b axes by taking the derivate with respect to x and y, as shown in Equations 3.4 and 3.5.

$$v_x = \frac{\partial f_{total}(x,y)}{\partial x}, \tag{3.4}$$

$$v_y = \frac{\partial f_{total}(x,y)}{\partial y}. \tag{3.5}$$

For this thesis, the LIDAR will need to be processed to identify and determine the relative position of the control goal, as well as any obstacles. The information can then be used to generate a local potential field around the quadcopter. From there, the gradient of the potential field can be used to set the velocities for the aircraft, about the x_b and y_b axis. Since it is desired to follow a vertical surface, the potential field is defined accordingly. The potential field is constructed by using only the minimum range measurement, from the LIDAR sensor, to determine the desired goal location, as shown below.

$$R = \sqrt{(x_0 - x_{min})^2 + (y_0 - y_{min})^2}, \tag{3.6}$$

$$f_{total}(x, y) = (R - J)^2, \tag{3.7}$$

where

$$x_{min} = \frac{\Gamma_{min}}{1000} cos(\Theta_{min} + \frac{\pi}{2}), \tag{3.8}$$

$$y_{min} = \frac{\Gamma_{min}}{1000} sin(\Theta_{min} + \frac{\pi}{2}), \tag{3.9}$$

J is the desired offset from the surface for the quadcopter to fly, x_0 and y_0 are the components of a position vector for the quadcopter, x_{min} and y_{min} are the components of a position to the location of the minimum range measurement on the surface, in the body frame, and Γ_{min} and Θ_{min} the magnitude and angle of the minimum range measurement, respectively. In this case x_0 and y_0 will always be zero since the LIDAR sensor assumed to be centered at the body reference frames origin. Since the LIDAR sensor is rotated clockwise about the z_b by $\frac{\pi}{2}$, as shown in Figure 2.2, $\frac{\pi}{2}$ is added to Θ_{min}, in Equations 3.8 and 3.9, allowing the potential field to be calculated in the body frame.

Assuming a small angular resolution in the LIDAR measurements, the control designs robustness to noise in the LIDAR measurements can be improved by applying a moving average filter as follows[30].

$$\bar{\mathbf{\Gamma}}_i = \frac{1}{M} \sum_{j=1}^{M} \mathbf{\Gamma}_{i+j}, \tag{3.10}$$

for distinct $i \in \{1, \ldots, (p + 1 - M)\}$, where $\bar{\mathbf{\Gamma}}_i$ is the averaged range measurement, M is the number of measurements used in the average, p is the total number of range measurements, i is the index of the moving average, and j is the index of the range measurement.

$$\bar{\mathbf{\Gamma}}_\beta = \min(\bar{\mathbf{\Gamma}}_i), \tag{3.11}$$

where β is the index of the minimum average range measurement. The magnitude and angle of the minimum range measurement can then be found as shown below.

$$\Gamma_{min} = \bar{\mathbf{\Gamma}}_\beta, \tag{3.12}$$

$$\Theta_{min} = \mathbf{\Theta}_{(\frac{M-1}{2} + \beta)}. \tag{3.13}$$

Since, in the absence of any other obstacles, the gradient of the potential field reaches zero when the quadcopter is at the set distance from the surface, the quadcopter needs an additional driving force to push it along the surface. In the current design this driving force is simply provided by a bias velocity, v_{bias}, that is added to v_x.

To complete the potential field control design, a velocity controller is needed for the quadcopter. Using the linear model, shown in Equations 2.57 and 2.58, a horizontal velocity controller can be designed.

The linear equations need to be truncated by removing the x and y position states from the dynamic model, as shown below. The truncation is valid since the x and y position states are not coupled to any other state in the quadcopter model for the purposes of velocity control.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \hat{\mathbf{B}}\boldsymbol{\delta\tau}, \tag{3.14}$$

$$\mathbf{y_m} = \mathbf{C}\mathbf{x} + \hat{\mathbf{D}}\boldsymbol{\delta\tau}, \tag{3.15}$$

where

$$\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & \dot{x} & \dot{y} & z & \dot{z} \end{bmatrix}^T, \tag{3.16}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.17}$$

$$\hat{\mathbf{B}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{-L}{I_{xx}} & 0 & \frac{L}{I_{xx}} \\ \frac{-L}{I_{yy}} & 0 & \frac{L}{I_{yy}} & 0 \\ \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} & \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix}, \tag{3.18}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.19}$$

$$\hat{\mathbf{D}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.20}$$

Using Equations 3.14 through 3.20 an output feedback controller can be designed. Given the need for robustness to steady-state error and constant disturbances, an integral controller is chosen. Based on the work completed in Ref. [27], an integral control design is completed as follows. The augmented plant is constructed as shown below.

$$\dot{\bar{\mathbf{x}}} = \bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{B}}\boldsymbol{\delta\tau} + \bar{\mathbf{G}}\bar{\mathbf{w}}, \tag{3.21}$$

$$\bar{\mathbf{y}}_{\mathbf{m}} = \bar{\mathbf{C}}_{\mathbf{m}}\bar{\mathbf{x}} + \bar{\mathbf{D}}_{\mathbf{m}}\boldsymbol{\delta\tau}, \tag{3.22}$$

$$\mathbf{y} = \bar{\mathbf{C}}\mathbf{x} + \bar{\mathbf{D}}\boldsymbol{\delta\tau}, \tag{3.23}$$

where the augmented state is

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^T & \mathbf{p}^T \end{bmatrix}^T, \tag{3.24}$$

and the augmented system matrices are

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0_{10\times3}} \\ \bar{\mathbf{C}} & \mathbf{0_{3\times3}} \end{bmatrix}, \tag{3.25}$$

$$\bar{\mathbf{B}} = \begin{bmatrix} \hat{\mathbf{B}} \\ \mathbf{0_{3\times4}} \end{bmatrix}, \tag{3.26}$$

$$\bar{\mathbf{G}} = \begin{bmatrix} \mathbf{I} & \mathbf{0_{10\times3}} \\ \mathbf{0_{3\times10}} & -\mathbf{I} \end{bmatrix}, \tag{3.27}$$

$$\bar{\mathbf{C}}_{\mathbf{m}} = \begin{bmatrix} \mathbf{C} & \mathbf{0_{6\times3}} \\ \mathbf{0_{3\times10}} & \mathbf{I} \end{bmatrix}, \tag{3.28}$$

$$\bar{\mathbf{D}}_{\mathbf{m}} = \begin{bmatrix} \hat{\mathbf{D}} \\ \mathbf{0_{3\times4}} \end{bmatrix}, \tag{3.29}$$

with augmented input

$$\bar{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ \mathbf{y_r} \end{bmatrix}, \tag{3.30}$$

where

$$\bar{\mathbf{C}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.31}$$

$$\bar{\mathbf{D}} = \mathbf{0_{4\times4}}, \tag{3.32}$$

$\mathbf{y_m}$ is the measured output which includes the Euler angles, horizontal velocities and altitude, $\mathbf{p}$ are the error states, $\mathbf{w}$ is a disturbance, $\bar{\mathbf{y}}_\mathbf{m}$ is the augmented measured output. $\mathbf{y}$ is the output to be tracked and $\mathbf{y_r}$ is the desired reference signal. With this design the outputs to be tracked and also the required reference signals are the horizontal velocities and altitude, where v_x and v_y are the reference velocities and h is the reference altitude. This new augmented system is now used to design an output feedback controller, in the form shown below.

$$\dot{\mathbf{z}} = \mathbf{Ez} + \begin{bmatrix} \mathbf{F_y} & \mathbf{F_p} \end{bmatrix} \begin{bmatrix} \mathbf{y_m} \\ \mathbf{p} \end{bmatrix}, \tag{3.33}$$

$$\boldsymbol{\delta\tau} = \mathbf{Gz} + \begin{bmatrix} \mathbf{H_y} & \mathbf{H_p} \end{bmatrix} \begin{bmatrix} \mathbf{y_m} \\ \mathbf{p} \end{bmatrix}. \tag{3.34}$$

The controller is observer-based, such that the controller states $\mathbf{z}$ are the estimates of $\bar{\mathbf{x}}$. To obtain the final controller the above controller must be augmented with the error states $\mathbf{p}$, which can be expressed as shown below.

$$\dot{\mathbf{p}} = \mathbf{A_p p} + \mathbf{B_p e}, \tag{3.35}$$

where

$$\mathbf{e} = \mathbf{y} - \mathbf{y_r}, \tag{3.36}$$

$$\mathbf{y} = \begin{bmatrix} \dot{x} & \dot{y} & z \end{bmatrix}^T, \tag{3.37}$$

$$\mathbf{y_r} = \begin{bmatrix} v_x^* & v_y & h \end{bmatrix}^T, \tag{3.38}$$

$$v_x^* = v_x + v_{bias}. \tag{3.39}$$

For this particular design $\mathbf{A_p} = \mathbf{0_{3\times3}}$ and $\mathbf{B_p} = \mathbf{I_{3\times3}}$. Combining Equations 3.33 through 3.35

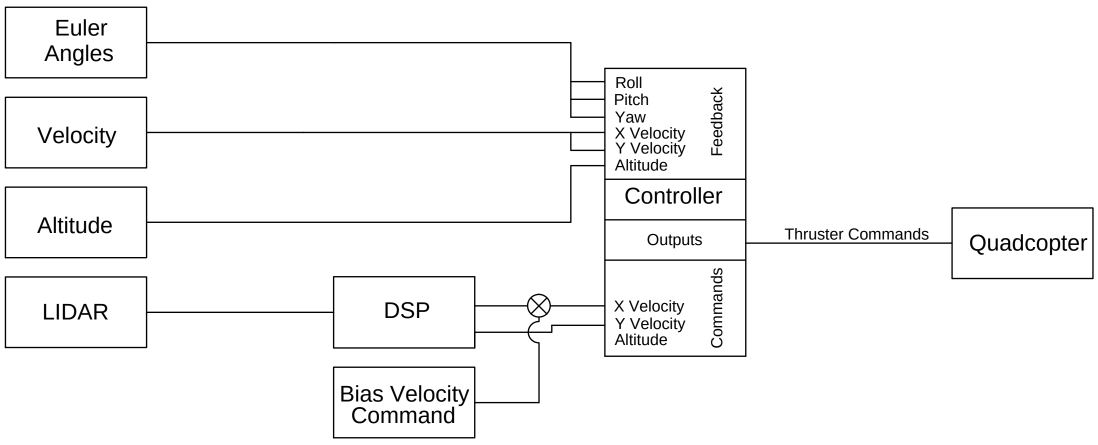

Figure 3.2: A schematic diagram of the quadcopter potential field controller.

results in the final controller shown below.

$$\begin{bmatrix}\dot{\mathbf{z}}\\ \dot{\mathbf{p}}\end{bmatrix} = \begin{bmatrix}\mathbf{E} & \mathbf{F_p}\\ \mathbf{0_{3\times 13}} & \mathbf{A_p}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{p}\end{bmatrix} + \begin{bmatrix}\mathbf{0_{13\times 3}} & \mathbf{F_y}\\ \mathbf{B_p} & \mathbf{0_{3\times 6}}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{y_m}\end{bmatrix}, \tag{3.40}$$

$$\boldsymbol{\delta\tau} = \begin{bmatrix}\mathbf{G} & \mathbf{H_p}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{p}\end{bmatrix} + \begin{bmatrix}\mathbf{0_{4\times 3}} & \mathbf{H_y}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{y_m}\end{bmatrix}, \tag{3.41}$$

where

$$\mathbf{y_m} = \begin{bmatrix}\phi & \theta & \psi & \dot{x} & \dot{y} & z\end{bmatrix}^T, \tag{3.42}$$

The complete control scheme is shown in Figure 3.2.

3.2.1 Curved Surface Following

In the previous potential field control design the quadcopter was demonstrated to hold a constant distance from a curved surface while maintaining a constant progression along the x_i axis. To allow the quadcopter to follow a curved surface, in any direction, there are two designs that were examined. Both of the proposed designs rely on a horizontal velocity controller with the addition of an integral yaw controller, which is formulated as follows. Based on Equations 3.14 through 3.20 the final output feedback controller is constructed as shown below.

$$\begin{bmatrix}\dot{\mathbf{z}}\\ \dot{\mathbf{p}}\end{bmatrix} = \begin{bmatrix}\mathbf{E} & \mathbf{F_p}\\ \mathbf{0_{4\times 13}} & \mathbf{0_{4\times 4}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{p}\end{bmatrix} + \begin{bmatrix}\mathbf{0_{13\times 4}} & \mathbf{F_y}\\ \mathbf{B_p} & \mathbf{0_{4\times 6}}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \mathbf{y_m}\end{bmatrix}, \tag{3.43}$$

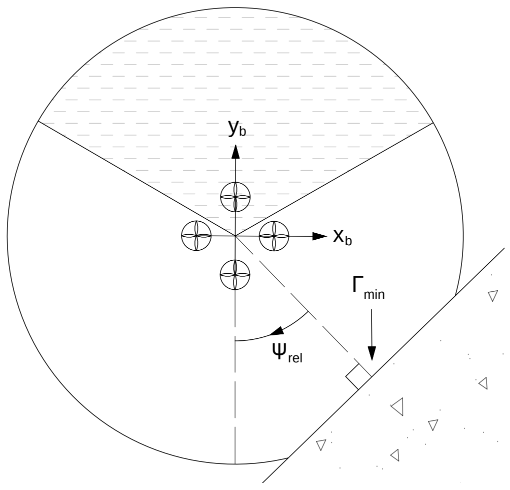

Figure 3.3: A diagram of the quadcopter flying along a surface.

$$\boldsymbol{\delta\tau} = \begin{bmatrix} \mathbf{G} & \mathbf{H_p} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{p} \end{bmatrix} + \begin{bmatrix} \mathbf{0_{4\times4}} & \mathbf{H_y} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{y_m} \end{bmatrix}, \tag{3.44}$$

$$\mathbf{y_m} = \begin{bmatrix} \phi & \theta & \psi & \dot{x} & \dot{y} & z \end{bmatrix}^T, \tag{3.45}$$

$$\mathbf{e} = \mathbf{y} - \mathbf{y_r}, \tag{3.46}$$

$$\mathbf{y} = \begin{bmatrix} \psi & \dot{x} & \dot{y} & z \end{bmatrix}^T, \tag{3.47}$$

$$\mathbf{y_r} = \begin{bmatrix} \psi_r & v_x^* & v_y^* & h \end{bmatrix}^T, \tag{3.48}$$

where $\mathbf{y}$ are the outputs to be tracked, $\mathbf{y_r}$ are the reference outputs, ψ_r is the reference yaw angle which in set to zero, and v_x^* and v_y^* are the reference velocities along the x_i and y_i axes, respectively. With this design the outputs to be tracked are the yaw angle, horizontal velocities, and altitude.

Both designs require the relative yaw angle of the quadcopter, ψ_{rel}, which is the angle formed between the normal line from the surface and the y_b axis as shown in Figure 3.3. This relative angle can be found from the LIDAR data by using the angle of the minimum range measurement, Θ_{min}, as shown below.

$$\psi_{rel} = -\Theta_{min}. \tag{3.49}$$

In the first design, shown in Figure 3.4, the velocity controller operates in the inertial frame meaning the bias velocity vector, $\mathbf{s_b} = \begin{bmatrix} v_{bias} & 0 & 0 \end{bmatrix}^T$, which always remain parallel to the surface, must be rotated before it can be applied to the velocity controller. The relative yaw angle is used to rotate the

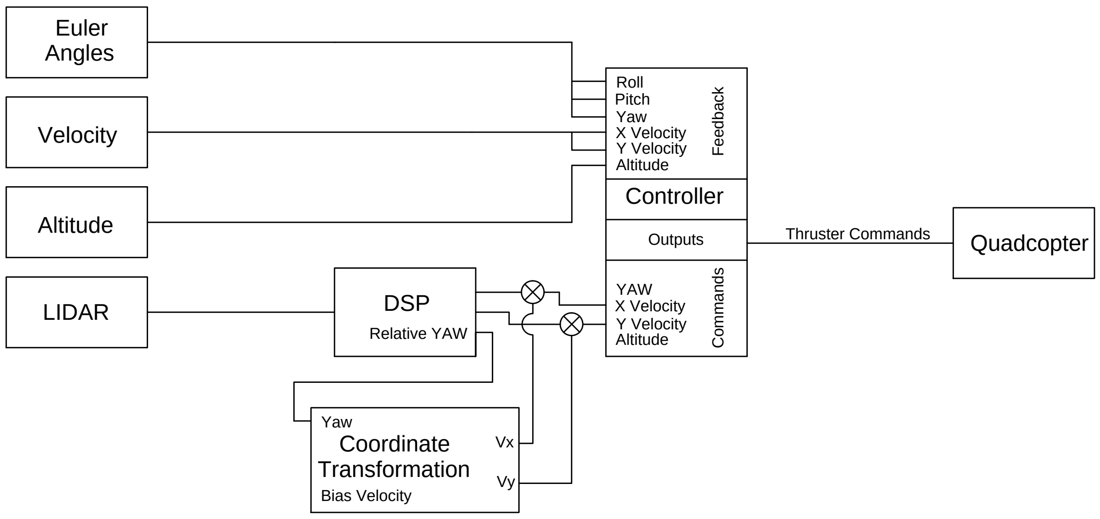

Figure 3.4: A schematic diagram of the quadcopter potential field controller.

bias velocity command to the inertial frame.

The rotation matrix used to transform between reference frames is shown in Equation 3.50 below.

$$\mathbf{R_z}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.50}$$

The reference velocity vector, $\mathbf{v}^* = \begin{bmatrix} v_x^* & v_y^* & 0 \end{bmatrix}^T$, is computed as shown below.

$$\mathbf{v}^* = \mathbf{R_z}(\psi_{rel}) \begin{bmatrix} v_{bias} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}, \tag{3.51}$$

where v_x and v_y are the reference velocities calculated using the potential field technique shown in section 3.2.

The above design provides the quadcopter with the ability to move any direction when using an ideal LIDAR scanner with a 360°scanning range. Though, since the LIDAR model used in this thesis has a limited scanning range of 239.77°, the controller design fails when the minimum range measurement falls outside of the scanning range of the LIDAR, in its blind spot. To compensate for the LIDAR's blind spot, referring to Figure 3.3, the next design depends on a controller to maintain the relative angle, ψ_{rel}, at zero. This means the horizontal velocity and yaw components of the output feedback controller will now be operating in the body frame. The new technique, shown in Figure 3.5, will use the same control architecture as shown in Equations 3.43 through 3.48, though modified as follows.

For the purposes of design in this thesis, it is assumed that motion along the wall is assumed slow

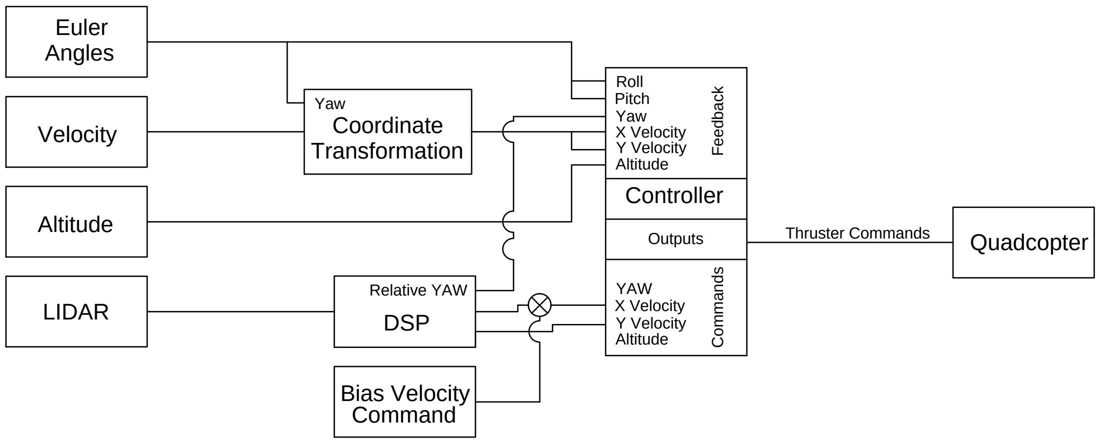

Figure 3.5: A schematic diagram of the quadcopter potential field controller.

enough to allow the surface to be instantaneously treated as flat with the x_i axis parallel to the surface. This means that ψ_{rel} has the same dynamics as ψ and may replace it, as shown below, without the need for any changes to the control design. Also with this assumption when transforming to the body coordinates, a linearization about a yaw angle of zero is used, which results in the same linear dynamics. This result means that states $\dot{x}$ and $\dot{y}$ can be replaced with $\dot{x}_b$ and $\dot{y}_b$, as shown below.

$$\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi_{rel} & \dot{\phi} & \dot{\theta} & \dot{\psi}_{rel} & \dot{x}_b & \dot{y}_b & z & \dot{z} \end{bmatrix}^T, \tag{3.52}$$

$$\mathbf{y_m} = \begin{bmatrix} \phi & \theta & \psi_{rel} & \dot{x}_b & \dot{y}_b & z \end{bmatrix}^T, \tag{3.53}$$

$$\mathbf{y} = \begin{bmatrix} \psi_{rel} & \dot{x}_b & \dot{y}_b & z \end{bmatrix}^T, \tag{3.54}$$

$$\mathbf{y_r} = \begin{bmatrix} \psi_r & v_x^* & v_y & h \end{bmatrix}^T, \tag{3.55}$$

where

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ 0 \end{bmatrix} = \mathbf{R_z}^{-1}(\psi) \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}, \tag{3.56}$$

$$v_x^* = v_{bias} + v_x. \tag{3.57}$$

3.3 Wall Following Control

The above control design allows the quadcopter to track a curved surface in any direction using a potential field control technique while requiring horizontal velocity measurements. In this section the potential field control is replaced with a wall following technique. In this design, shown in Figure 3.1, the controller directly regulates the quadcopters distance from the surface reducing the need for velocity measurements to only the velocity along the x_b axis.

Using the linearized dynamic model, shown in Equations 2.57 and 2.58, an output feedback controller can be developed using the same method shown in section 3.2. The linearized model is first reduced to remove x from the state vector, as shown below.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \hat{\mathbf{B}}\boldsymbol{\delta\tau}, \tag{3.58}$$

$$\mathbf{y_m} = \mathbf{C}\mathbf{x} + \hat{\mathbf{D}}\boldsymbol{\delta\tau}, \tag{3.59}$$

where

$$\mathbf{x} = \begin{bmatrix}\phi & \theta & \psi_{rel} & \dot{\phi} & \dot{\theta} & \dot{\psi}_{rel} & \dot{x}_b & d & \dot{d} & z & \dot{z}\end{bmatrix}^T, \tag{3.60}$$

$$\mathbf{A} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
-g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{3.61}$$

$$\hat{\mathbf{B}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{-L}{I_{xx}} & 0 & \frac{L}{I_{xx}} \\ \frac{-L}{I_{yy}} & 0 & \frac{L}{I_{yy}} & 0 \\ \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} & \frac{K_{tor}}{K_t I_{zz}} & \frac{-K_{tor}}{K_t I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix}, \tag{3.62}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.63}$$

$$\hat{\mathbf{D}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.64}$$

As discussed in section 3.2.1, since the surface is assumed to be flat and the y and y_b are aligned then the same linear dynamic can apply to both. Referring to section 3.2 an integral control design is completed resulting in the following output feedback controller.

$$\begin{bmatrix} \dot{\mathbf{z}} \\ \dot{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \mathbf{E} & \mathbf{F_p} \\ \mathbf{0_{4\times15}} & \mathbf{A_p} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{p} \end{bmatrix} + \begin{bmatrix} \mathbf{0_{15\times4}} & \mathbf{F_y} \\ \mathbf{B_p} & \mathbf{0_{4\times6}} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{y_m} \end{bmatrix}, \tag{3.65}$$

$$\boldsymbol{\delta\tau} = \begin{bmatrix} \mathbf{G} & \mathbf{H_p} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{p} \end{bmatrix} + \begin{bmatrix} \mathbf{0_{4\times4}} & \mathbf{H_y} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{y_m} \end{bmatrix}, \tag{3.66}$$

where

$$\mathbf{e} = \mathbf{y} - \mathbf{y_r}, \tag{3.67}$$

$$\mathbf{y} = \begin{bmatrix}\psi_{rel} & \dot{x}_b & d & z\end{bmatrix}^T, \tag{3.68}$$

$$\mathbf{y_r} = \begin{bmatrix}\psi_r & v_x^* & d_{ref} & h\end{bmatrix}^T, \tag{3.69}$$

$$\mathbf{y_m} = \begin{bmatrix}\phi & \theta & \psi_{rel} & \dot{x}_b & d & z\end{bmatrix}^T, \tag{3.70}$$

$\mathbf{y}$ is the output to be tracked, $\mathbf{y_r}$ is the desired reference, $\mathbf{y_m}$ are the measured outputs, d is the minimum distance to the surface, and d_{ref} is the desired distance between the quadcopter and the surface. In this design the outputs to be tracked are the relative yaw angle with respect to a surface, velocity along the x_b axis, the position with respect to a surface along the y_b axis, and the altitude. As discussed in section 3.2.1, since the surface is assumed to be flat and the y and y_b are aligned then the same linear dynamic can apply to both. This means d, which is aligned with the y axis in steady-state, can replace the y state, as shown in the Equations 3.60 through 3.70 above.

The reference signals for the controller are constructed using the following equations.

$$\phi_r = 0, \tag{3.71}$$

$$v_x^* = v_{bias}, \tag{3.72}$$

$$d = \frac{\Gamma_{min}}{1000}. \tag{3.73}$$

Chapter 4

Simulation

4.1 Simulation Software

The simulation and testing of the control schemes, developed in section 3, is performed using MATLAB and Simulink. The non-linear dynamic model, constructed in section 2.1, was first realized in a subsystem, as shown in Figure 4.1. The subsystem consists of an input thrust vector, $\mathbf{T} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 \end{bmatrix}^T$, and an output state vector, $\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$. The input thrusts are passed through a saturation block that limits the four individual thrusters to a minimum of 0 newtons and maximum of 6 newtons. The non-linear dynamic equations, from section 2.1, are computed using a function block, an integrator, and a vector of constants, to set the initial states. To configure the quadcopter model the parameters from [32] are used and shown in table 4.1. For all simulations shown in this thesis the quadcopter model is initialized with state $y = 3.5$ m, with all other states set to zero.

The LIDAR sensor is realized using a triggered subsystem, shown in Figure 4.2 with inputs x, y, and ψ and output vectors which contain the ranges and angles of each of the LIDAR measurements. When

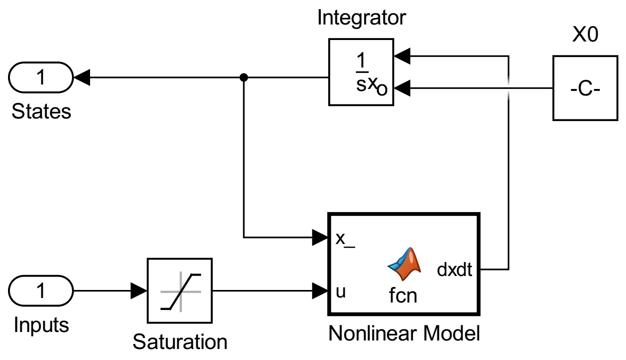

Figure 4.1: A Simulink subsystem containing the non-linear quadcopter model.

Table 4.1: Quadcopter parameter values for simulation

Parameter	Value	Unit
g	9.81	$\mathrm{m/s^2}$
m	0.468	kg
L	0.225	m
K_t	1.0×10^{-4}	$(\mathrm{N \cdot s^2})/(\mathrm{rad^2})$
K_{tor}	2.5×10^{-5}	$(\mathrm{N \cdot m \cdot s^2})/(\mathrm{rad^2})$
I_{xx}	4.856×10^{-3}	$\mathrm{kg \cdot m^2}$
I_{yy}	4.856×10^{-3}	$\mathrm{kg \cdot m^2}$
I_{zz}	8.801×10^{-3}	$\mathrm{kg \cdot m^2}$

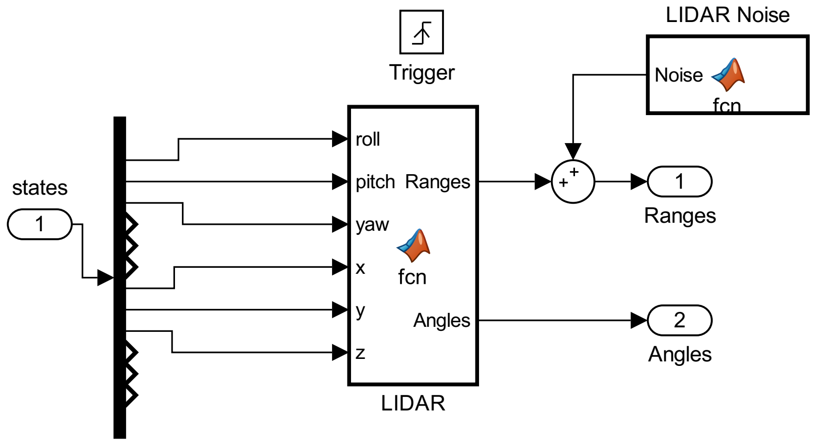

Figure 4.2: A Simulink subsystem containing the non-linear quadcopter model.

triggered by an external clock signal, with a period equal to the scan time of the sensor, the function block calculates the LIDAR measurements using the equations shown in section 2.3. For this thesis the simulated LIDAR sensor is modeled based on the URG-04LX-UG01 LIDAR sensor, using parameter values from Ref. [1], as shown in table 4.2.

Throughout the simulations shown in this chapter both the feedback measurements, obtained from the quadcopter model, and LIDAR measurements are corrupted by Gaussian noise with the corresponding standard deviation given in table 4.3. The following simulations are conducted using the 3D LIDAR model shown in Equations 2.67 through 2.74.

4.2 Potential Field Control

The first control technique that is to be examined is the potential field control design developed in section 3.2. Using the design procedure, proposed in section 3.2, an output feedback controller is designed. The LQR technique is used to calculate the control gains with design matrices $\mathbf{Q} = \mathbf{I_{13}}$ and $\mathbf{R} = \mathbf{I_4}$. The

Table 4.2: LIDAR parameter values for simulation

Parameter	Value	Unit
Resolution	1	mm
Accuracy	+/- 30	mm
Maximum Range Measurement	4000	mm
Scan Angle	Approx. 240	deg
Angular Resolution	Approx. 0.36 (360/1024)	deg
Scan Time	100	mSec/scan
Total Data Points	683	

Table 4.3: Noise parameter values for simulation

Measurement	Standard Deviation	Unit
ϕ	6.7×10^{-3}	rad
θ	6.7×10^{-3}	rad
ψ	6.7×10^{-3}	rad
z	166.7×10^{-3}	m
$\dot{x}$	33.3×10^{-3}	m/s
$\dot{y}$	33.3×10^{-3}	m/s
Γ_j	10.0×10^{-3}	m

pole placement technique is then used to find the observer gains by ensuring the observer poles are ten times greater than those of the controller. The designed controlled is then added to the simulation using a state space block, with the initial conditions of all the states set to zero.

Referring to Figure 3.2 the controller is provided with all the required feedback and reference signal as shown in Figure 4.3. The horizontal reference velocities are provided via potential field control which is calculated in the digital signal processing (DSP) function block. The DSP block receives the ranges and angles of the LIDAR measurements and desired following distance as input. The minimum range measurement is found, using a moving average filter, and used to construct a potential field with only a goal point located at the desired distance from the wall, as calculated in Equation 3.7. The gradient is then calculated and used as the output of the function block. The resulting gradients are too large and result in an undesirable oscillating response. To fine tune the response both gradients are scaled by multiplying them by a constant, K_{pot}, to obtain v_x and v_y. v_x is combined with a bias velocity before both velocity commands are connected to the reference velocity inputs of the controller. The altitude reference is set to 0 m throughout all simulations in this thesis. The thrust outputs of the controller are then added with $\boldsymbol{\tau_0}$, given in Equation 2.54, before being passed to the quadcopter model, as thrust inputs.

A comparison of the surface following performance of the controller, in the absence of noise, while using different values for K_{pot} is shown in Figure 4.4, where the following distance is stepped from 2 m to 2.5 m and 1 m at 15 s and 30 s, respectively. For this thesis 0.1 is selected as the ideal value since it results in a quick response, while still maintaining a small overshoot.

The surface following performance of the potential field controller is tested with and without the

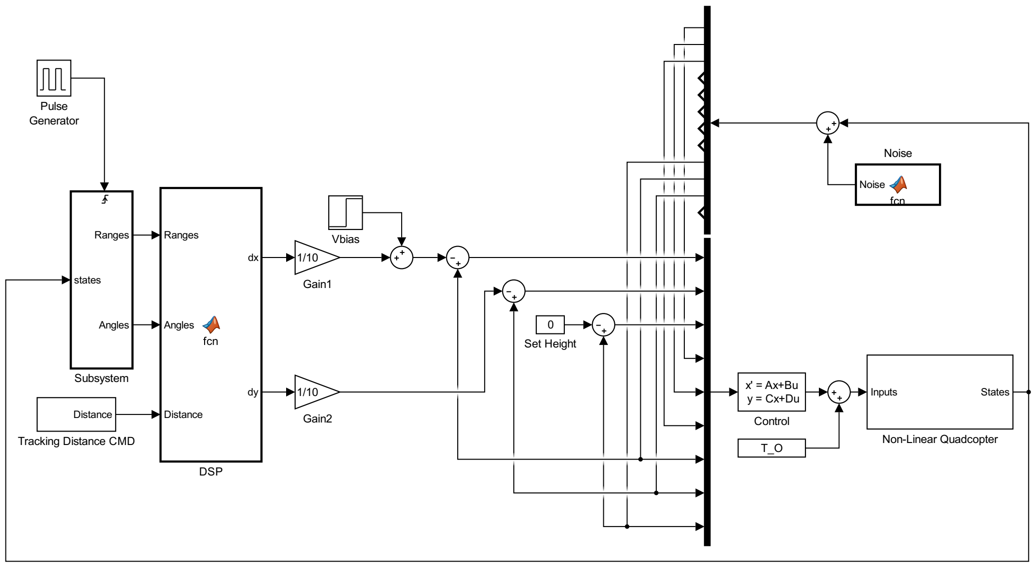

Figure 4.3: A Simulink model used to simulate the potential field control design.

moving average applied to the LIDAR measurements, as shown in Figure 4.5. For the purpose of testing the moving average filter, the standard deviation of the noise added to the LIDAR measurements is increased to 30×10^{-3} m. Various samples sizes are tested for the filter design, though, the best performance is seen when 5 range measurements are used for the moving average. The filter can be seen to nearly remove the steady-state tracking error in the minimum distance to the surface. From this point forward all simulations will use a 5 range measurement moving average on the LIDAR measurements.

The performance of the potential field control design while following a vertical surface, defined by the following function, can be seen in Figures 4.6 and 4.7.

$$f(x) = \begin{cases} 2 & \text{if } x < 70 \\ 1.5 & \text{if } 70 <= x <= 72 \\ 2 & \text{if } x > 72 \end{cases}, \tag{4.1}$$

where the units are in meters. The controller begins initially maintaining a 2 m distance from the surface before the following distance is stepped to 2.5 m and 1 m at 15 s and 30 s, respectively. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s. Referring to Figures 4.6 and 4.7 the proposed design is able to track a flat surface with no steady-state error. The controller handles step changes in following distance well, with a settling time of approximately 5 s. The controller also allows the quadcopter to pass the 2 m surface gap, seen between $x = 70$ m and $x = 72$ m, with relatively no disturbance to the trajectory of the quadcopter.

Next the performance of the controller is tested while maintaining a 2 m distance from a sinusoidal surface, defined by the following function, as shown in Figures 4.8 through 4.10.

$$f(x) = 2\sin\frac{x}{4} + 1, \tag{4.2}$$

where the units are in meters. Figure 4.8 shows simulation with v_{bias} stepped from 0 m/s to 1.6 m/s at 10 s, while in Figures 4.9 through 4.10 v_{bias} stepped from 0 m/s to 0.6 m/s at 10 s. Referring to Figure 4.8 the controller is shown to perform very poorly with the quadcopter crashing into the surface after traveling 27 m along the x_i axis. The cause of the crash is a result of the bias velocity pushing the quadcopter into the surface. Referring to Figures 4.9 through 4.10 the performance of the controller has improved with the lower velocity, though still remains very poor with a large following error of 1.4 m peak to peak.

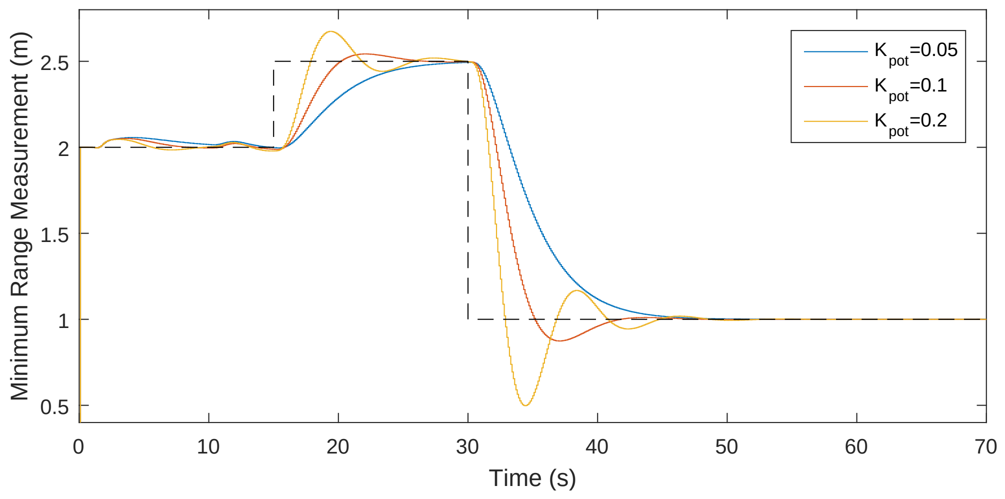

Figure 4.4: A comparison of the performance of the potential field controller while following a flat surface with various values for K_{pot}.

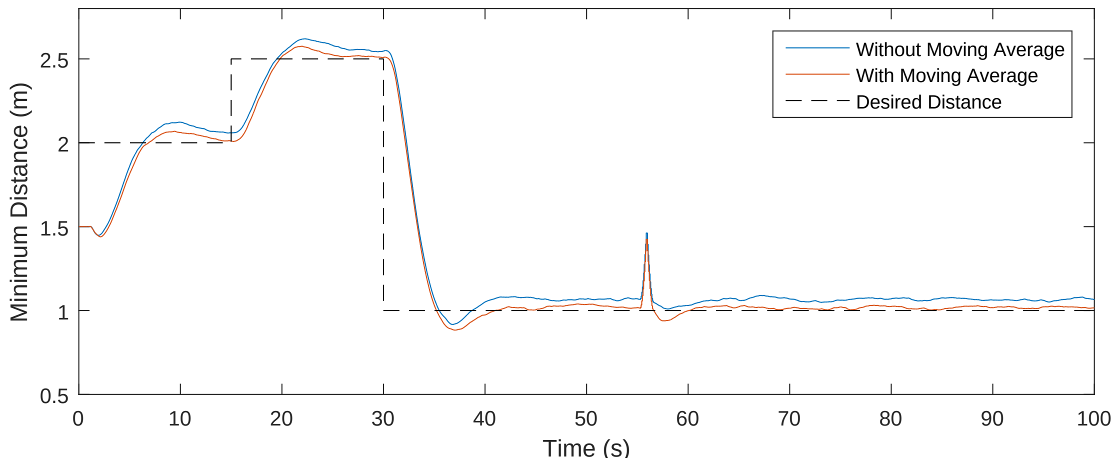

Figure 4.5: A comparison of the performance of the potential field controller response with and without a moving average applied to the LIDAR measurements.

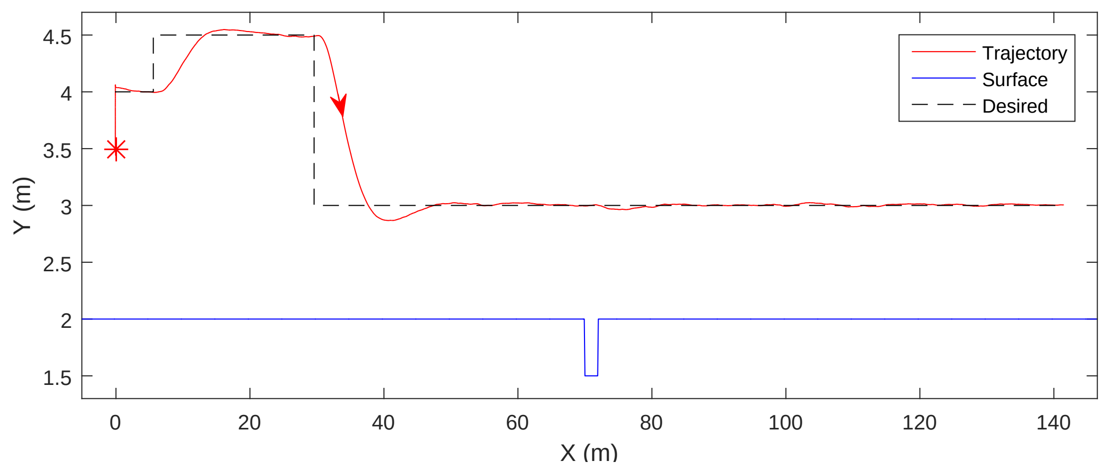

Figure 4.6: The trajectory followed by the quadcopter while initially maintaining a 2 m distance from a vertical surface. The desired following distance is stepped to 2.5 m and 1 m at 15 s and 30 s, respectively.

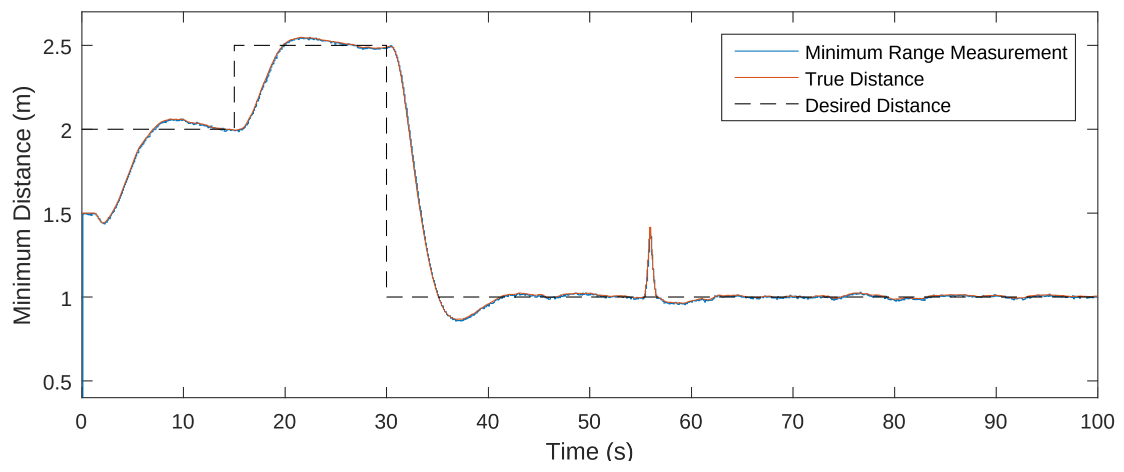

Figure 4.7: The minimum distance to a vertical surface.

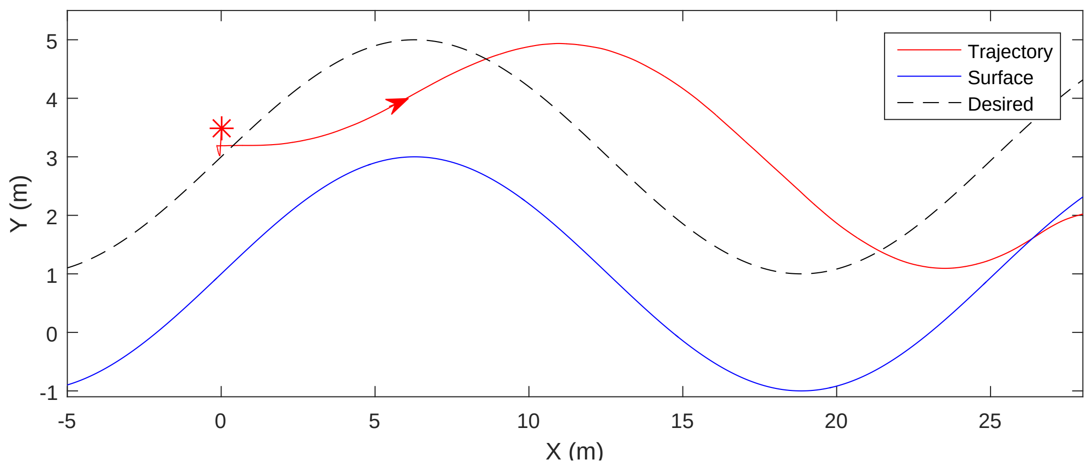

Figure 4.8: The trajectory followed by the quadcopter while maintaining a 2 m distance from a sinusoidal vertical surface, at a velocity of 1.6 m/s along the x_i axis. The quadcopter crashes into the surface after traveling 26 m.

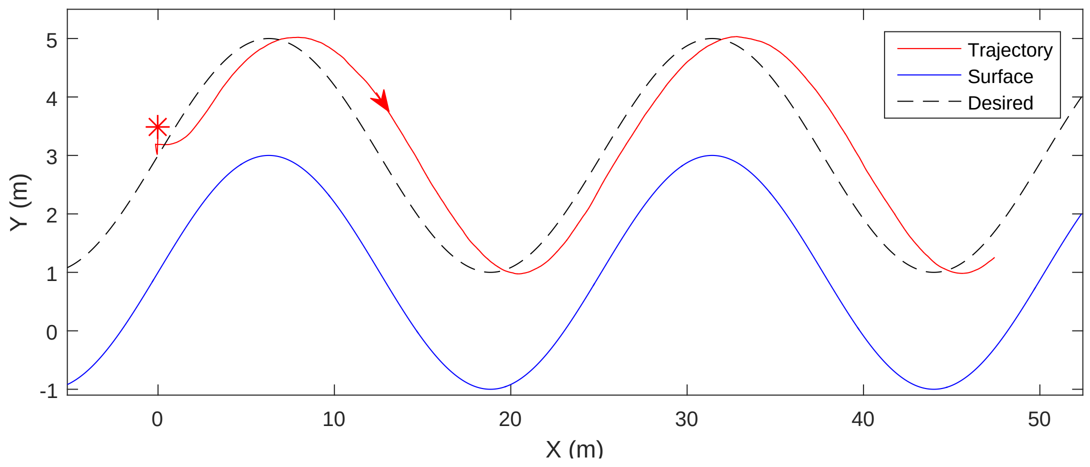

Figure 4.9: The trajectory followed by the quadcopter while maintaining a 2 m distance from a sinusoidal vertical surface, at a velocity of 0.6 m/s along the x_i axis.

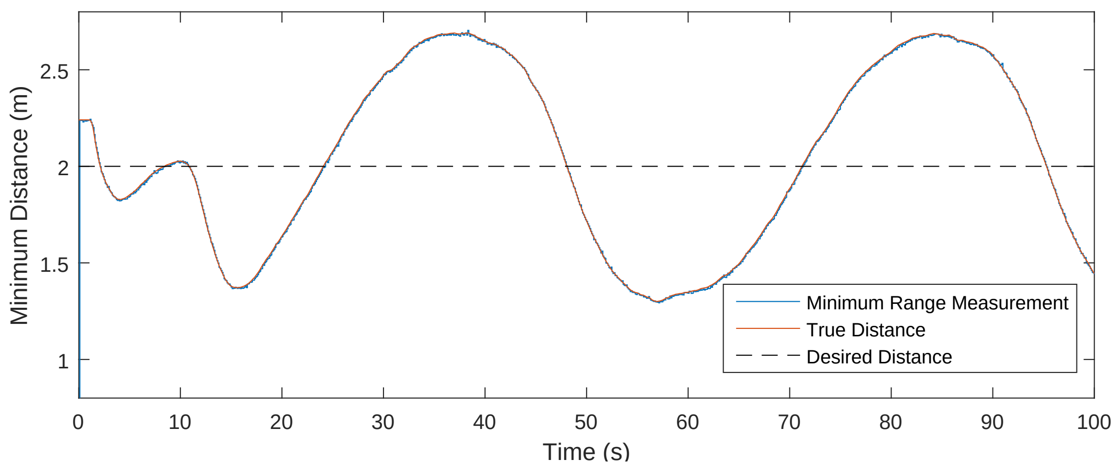

Figure 4.10: The minimum distance to a sinusoidal vertical surface.

4.2.1 Curved Surface following

Building on the potential field design shown above, referring to Figure 3.5, this design adds a frame transformation to transform the velocity measurements from the inertial to the body frame, using Equation 3.50. Using the same procedure as for the previous design a modified output feedback controller is designed with the addition of integral yaw control, using Equations 3.48 through 3.43. The control gains are again calculated using the LQR technique with design matrices $\mathbf{Q} = \mathbf{I_{14}}$ and $\mathbf{R} = \mathbf{I_4}$.

Referring to the Simulink model shown in Figure 4.11 the DSP block now also provides the angle of the minimum range measurement which is multiplied by a gain of -1 before being used as the yaw measurement for the output feedback controller. The new design also includes the addition of a frame transform block which takes the quadcopter horizontal inertial velocities and yaw angle and provides the body frame velocities as output, which are used as velocity measurements for the output feedback controller.

The performance of the improved potential field control design while following a vertical surface, given by Equation 4.1, can be seen in Figures 4.6 and 4.7. The controller begins initially maintaining a 2 m distance from the surface before the following distance is stepped to 2.5 m and 1 m at 15 s and 30 s, respectively. v_{bias} is stepped from 0 m/s to 1.6 m/s and at 10 s. The controller appears to respond to step changes in following distance well with a settling time of approximately 5 s, though, the controller appears to have poor noise rejection with significant error seen in minimum distance when following a flat surface. The controller also suffers from a relatively large tracking error of 0.4 m when passing the surface gap, between $x = 70$ m and $x = 72$ m.

Next the performance of the controller is tested while maintaining a 2 m distance from a sinusoidal surface, given by Equation 4.2, as shown in Figures 4.15 through 4.18. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s. The controller is shown to track the sinusoidal surface relatively well with a peak tracking error of 0.5 m.

Finally the performance of the controller is tested while navigating around the outside and inside of a corner while maintaining a 2 m distance from the surface, defined by the following function, as shown in Figures 4.19 through 4.21.

$$f(x) = \begin{cases} x & \text{if } x < 30 \\ |x - 60| & \text{if } x >= 30 \end{cases}, \tag{4.3}$$

where the units are in meters. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s. The controller performs well while navigating the outside of the corner, at 35 s, with a 0.8 m peak to peak tracking error seen in Figure 4.20. The controller performance while navigating the inside of the corner, at 62 s, is less desirable with a peak to peak tracking error of 1.4 m. The error while following a flat surface is also seen here.

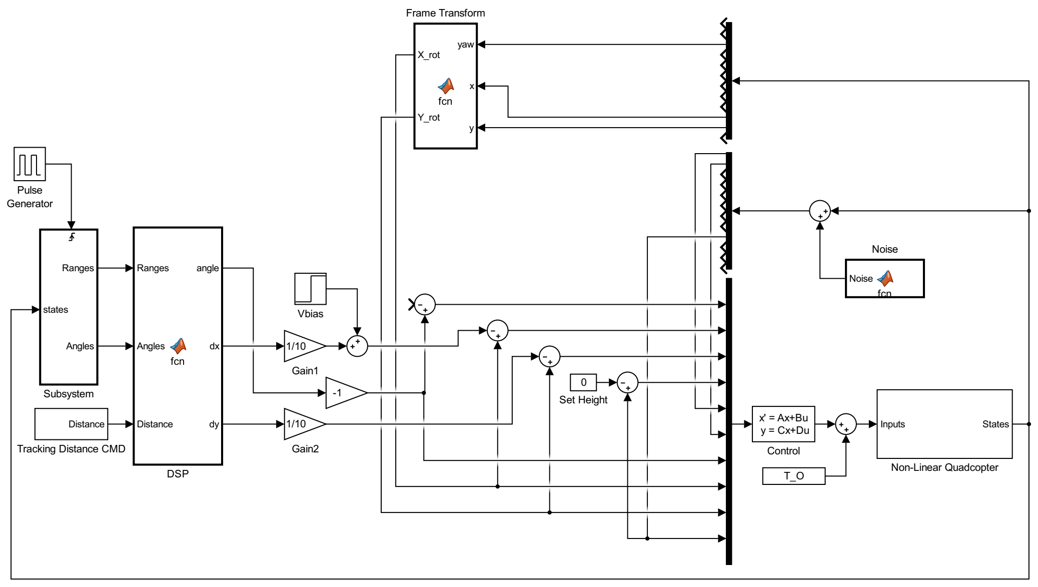

Figure 4.11: A Simulink model used to simulate the potential field control design.

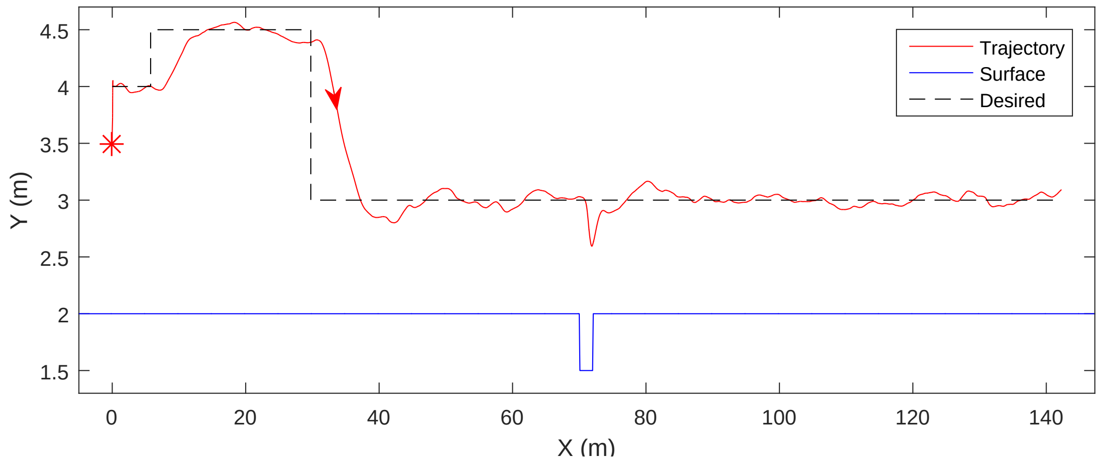

Figure 4.12: The trajectory followed by the quadcopter while initially maintaining a 2 m distance from a vertical surface.

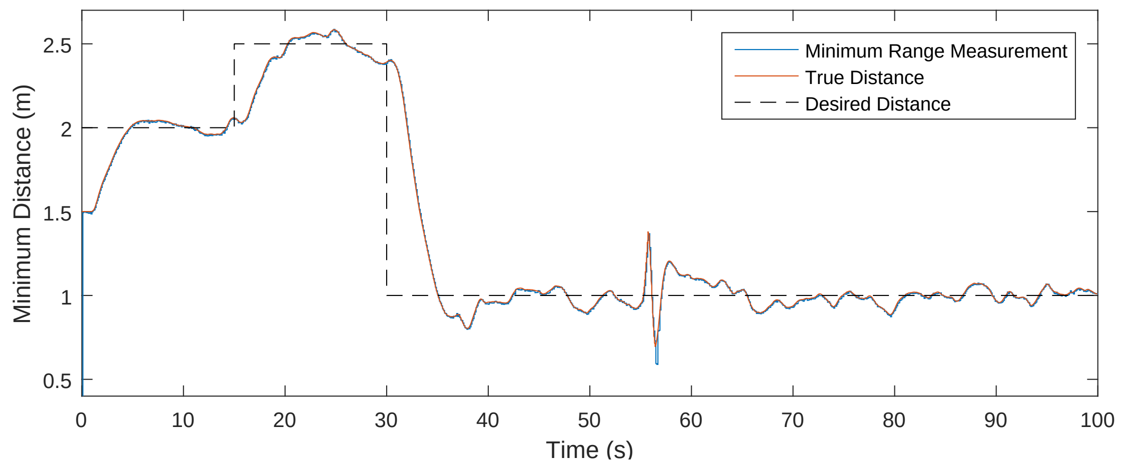

Figure 4.13: The minimum distance to a flat vertical surface.

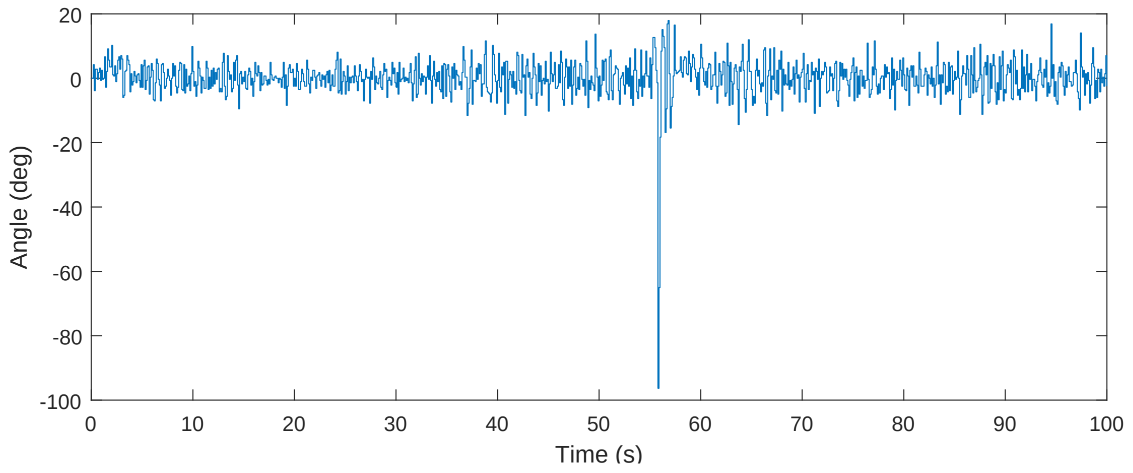

Figure 4.14: The relative yaw angle of the quadcopter while following a vertical surface.

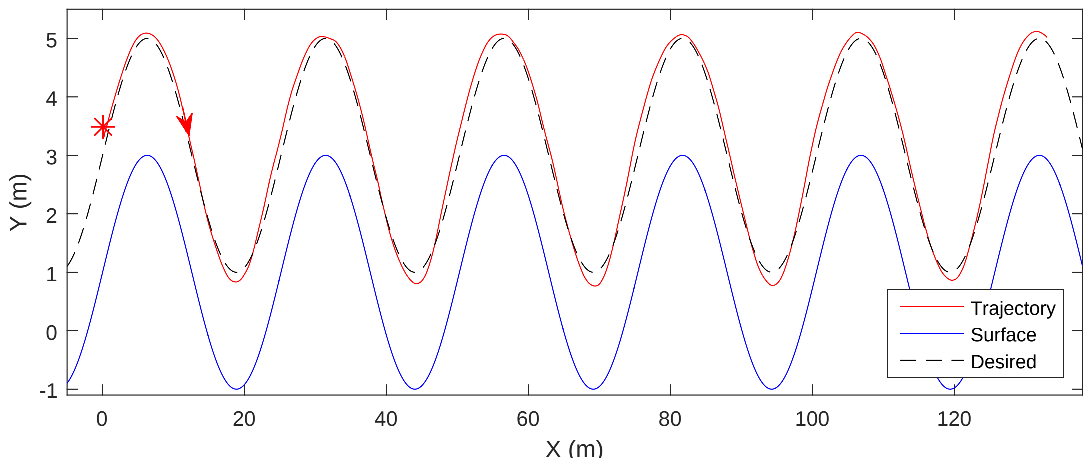

Figure 4.15: The trajectory followed by the quadcopter while maintaining a 2 m distance from a sinusoidal vertical surface, at a velocity of 1.6 m/s along the x_b axis.

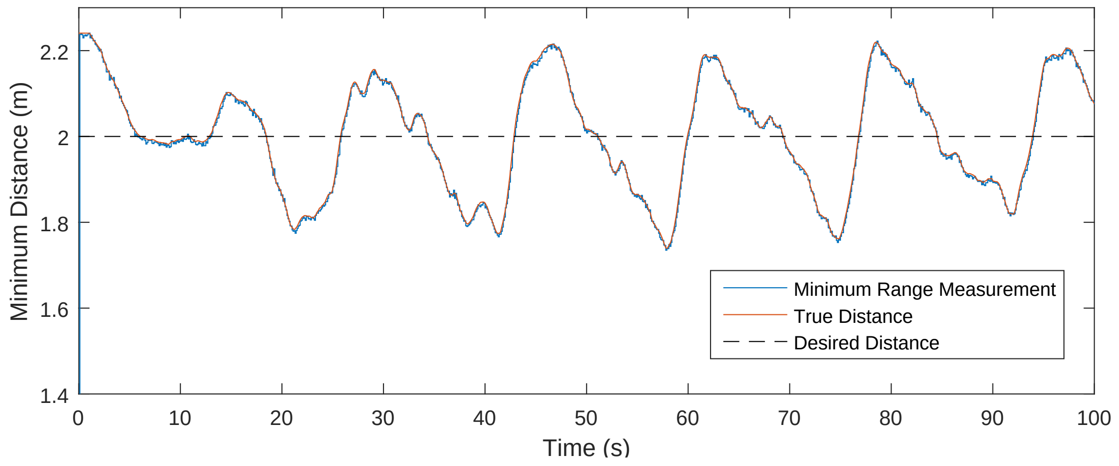

Figure 4.16: The minimum distance to a sinusoidal vertical surface.

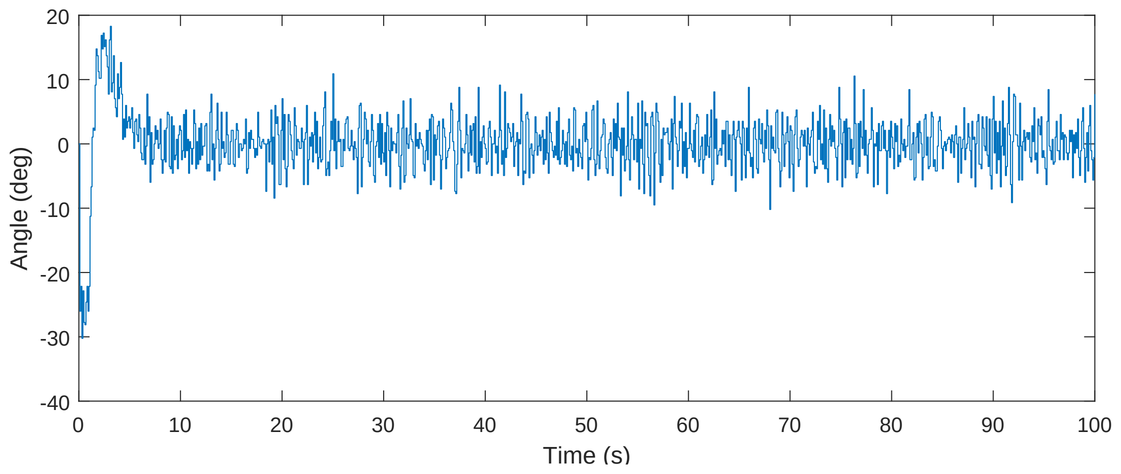

Figure 4.17: The relative yaw angle of the quadcopter while following a sinusoidal vertical surface.

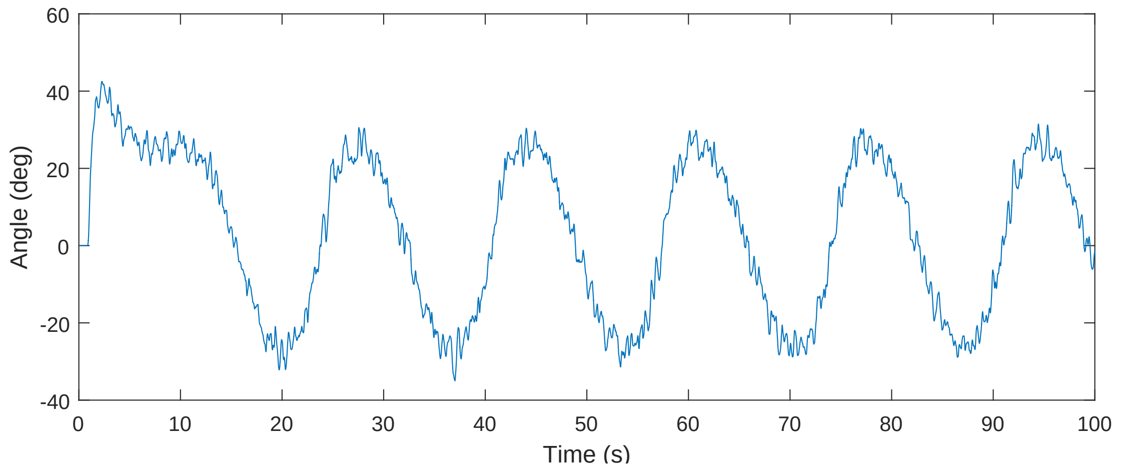

Figure 4.18: The yaw angle of the quadcopter while following a sinusoidal vertical surface.

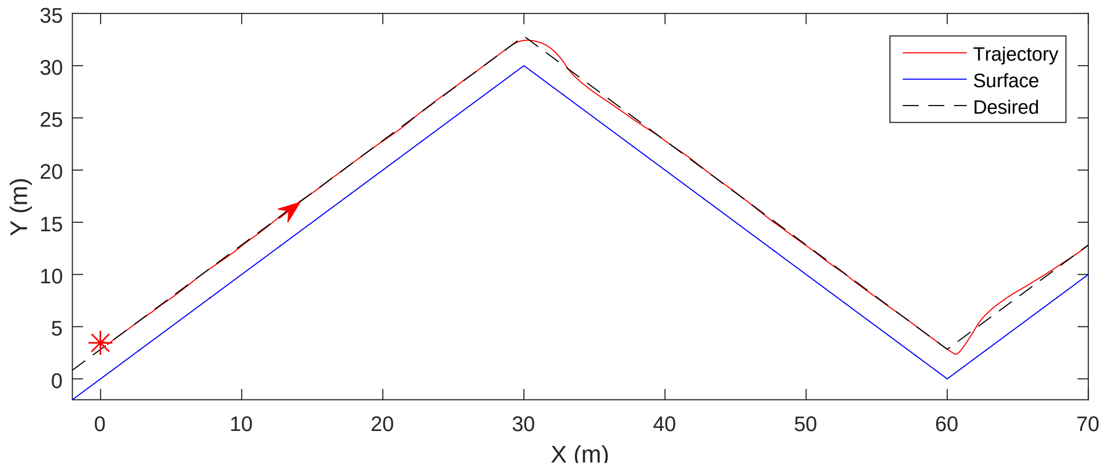

Figure 4.19: The trajectory followed by the quadcopter while maintaining a 2 m distance from a vertical surface while navigating outside and inside corners, at a velocity of 1.6 m/s along the x_b axis.

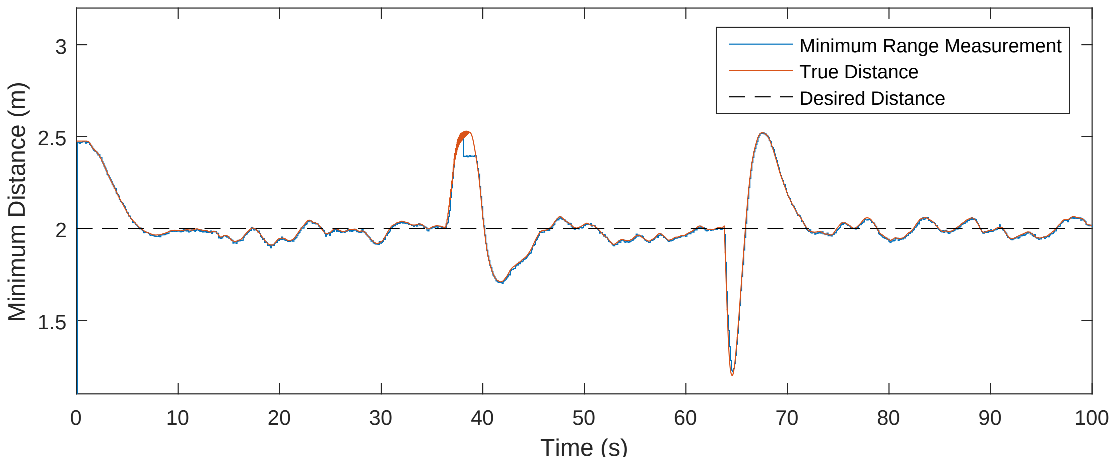

Figure 4.20: The minimum distance to a vertical surface while navigating outside and inside corners, at 35s and 65 s, respectively.

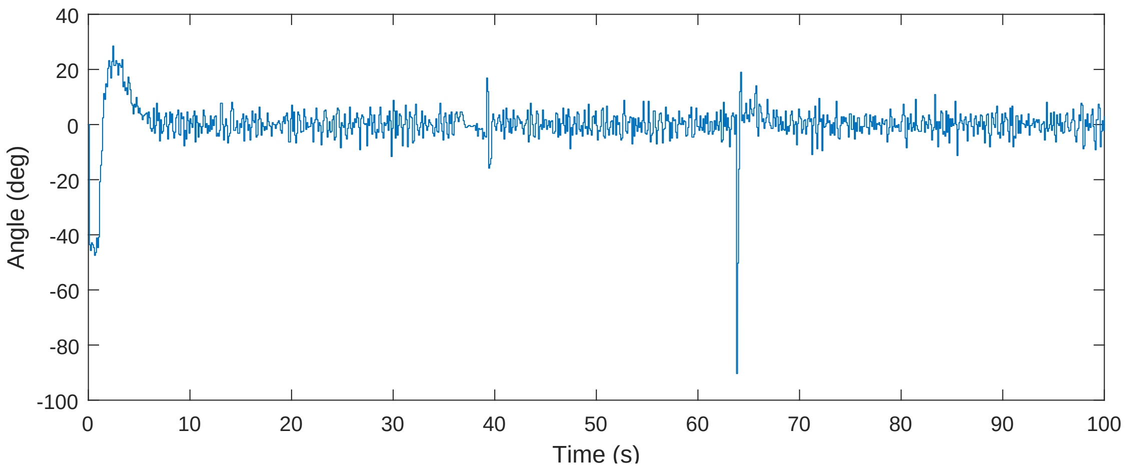

Figure 4.21: The relative yaw angle of the quadcopter while navigating outside and inside corners.

4.3 Wall Following Control

The wall following control design, from section 3.3 and shown in Figure 3.1, is simulated using the Simulink model shown in Figure 4.22. Using the same procedure, shown in the previous section, an output feedback controller is designed using Equations 3.58 through 3.70. Through testing design matrices $\mathbf{Q} = diag(100\mathbf{I_3}, \mathbf{I_{16}})$ and $\mathbf{R} = 50\mathbf{I_4}$ are selected to provide the best surface following performance.

In this design the DSP block has the LIDAR measurement along with a bias velocity as inputs. The LIDAR measurements are processed with the moving average filter to determine the range and angle of the minimum LIDAR measurement, which are provided as outputs. The LIDAR measurements are also used to determine if any obstacles, such as walls, are directly in front of or behind the quadcopter. In the event of an obstacle, the output speed command will reduce to 0.4 m/s, otherwise the speed command will be equal to the bias velocity.

Since the LIDAR provides measurements in millimeters, the minimum range measurement is passed through a gain block with a value of 1/1000 before it is passed to the controller as the distance measurement. The angle of the minimum range measurement is passed though a gain block with a value of -1 before it is passed to the controller as the yaw measurement. To improve initial transients response of the controller, 2 m is subtracted from both the reference distance, d_{ref}, and measurement, d, as shown in Figure 4.22.

The performance of the wall following controller design while following a vertical surface, given by Equation 4.1, is shown in Figures 4.23 through 4.25. The controller begins initially maintaining a 2 m distance from the surface before the following distance is stepped to 2.5 m and 1 m at 15 s and 30 s, respectively. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s. After the initial transients the controller performs very well with no overshoot or undershoot resulting from step changes in the desired following distance. This behavior is very desirable when following a surface in close proximity to lower the risk of a collision with the surface. The controller, however, suffers from a relatively large tracking error of 0.7 m peak to peak when passing the surface gap, between $x = 70$ m and $x = 72$ m.

The performance of the wall following controller design while maintaining a 2 m distance from a sinusoidal vertical surface, given by Equation 4.2, can be seen in Figures 4.26 through 4.29. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s. After the initial transients the controller performs very well with only a small error of 0.4 m peak to peak is seen between the desired and measured distances.

The performance of the controller while navigating around the outside and inside of a corner while maintaining a 2 m distance from the surface, given by Equation 4.3, can be seen in Figures 4.30 through 4.32. v_{bias} is stepped from 0 m/s to 1.6 m/s at 10 s and the quadcopter begins navigating the outside and inside corners at 35 s and 65 s, respectively. After the initial transients the controller performs well with tracking errors of 1.2 m and 1.1 m peak to peak while navigating the outside and inside of the corner, respectively.

Finally Figure 4.33 shows a comparison of the performance of the wall following controller while navigating the inside of a corner at a distance of 2 m with and without speed reduction. When a surface is detected in front or behind of the quadcopter, the speed command is reduced to 0.4 m/s. The speed reduction proves to have a significant impact on the performance of the controller reducing the tracking

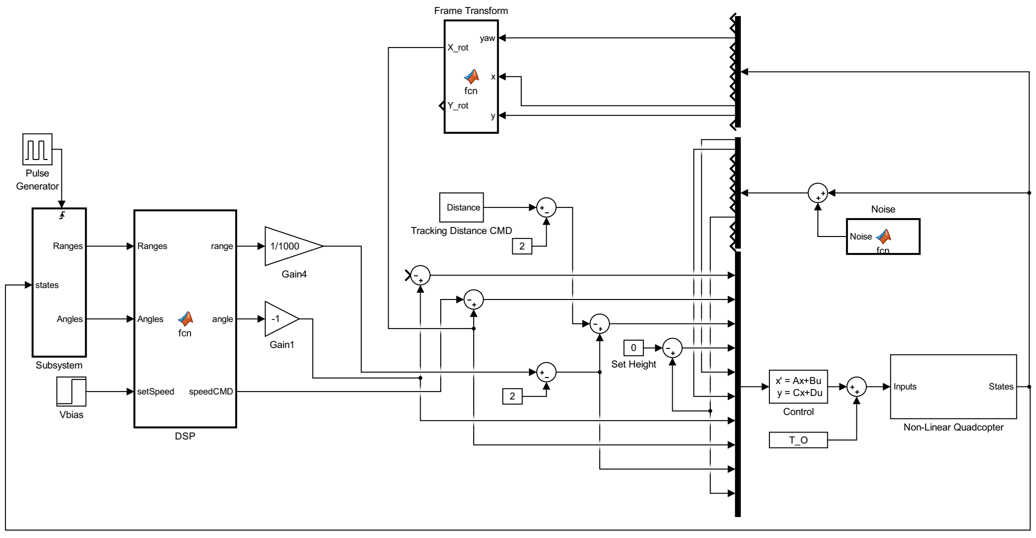

Figure 4.22: A Simulink model used in the simulation of the wall following controller.

error from 1.7 m peak to peak without speed reduction to 1.1 m peak to peak with speed reduction.

Comparing these results with those of the two earlier designs, it is clear that this design provides better robustness to steady-state error in the surface tacking distance. The wall following controller also appears to have better robustness to noise, with a very small error seen when following a flat surface. The wall following control design has slightly better performance to step changes in the surface following distance, with a quicker setting time and no overshoot. The wall following controller results in the smallest tracking error when following a sinusoidal surface. Comparing the controllers behavior when navigating the inside and outside of corners, the wall following control design results in a most desirable performance, with the smallest peak to peak tracking errors in the following distance.

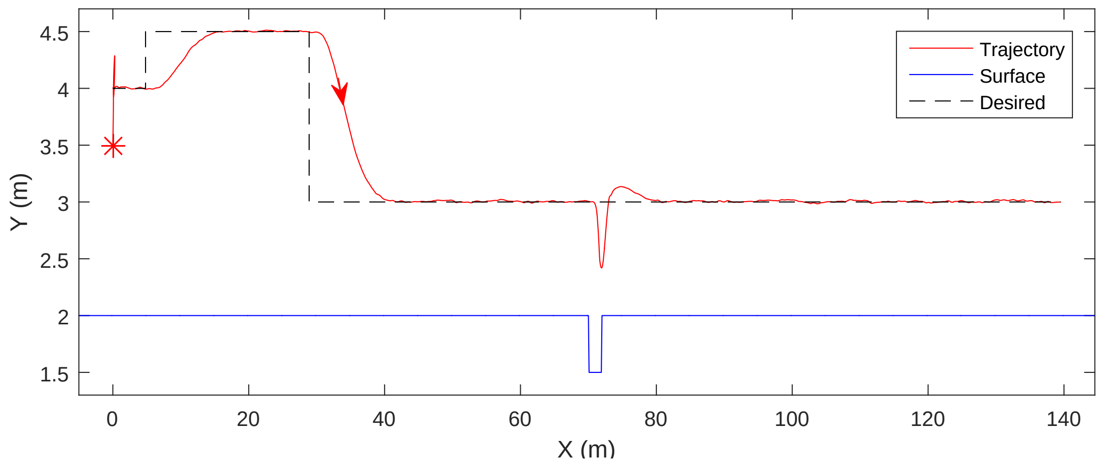

Figure 4.23: The trajectory followed by the quadcopter while initially maintaining a 2.5 m distance from a vertical surface. The desired following distance is stepped to 2.5 m and 1 m at 15 s and 30 s, respectively.

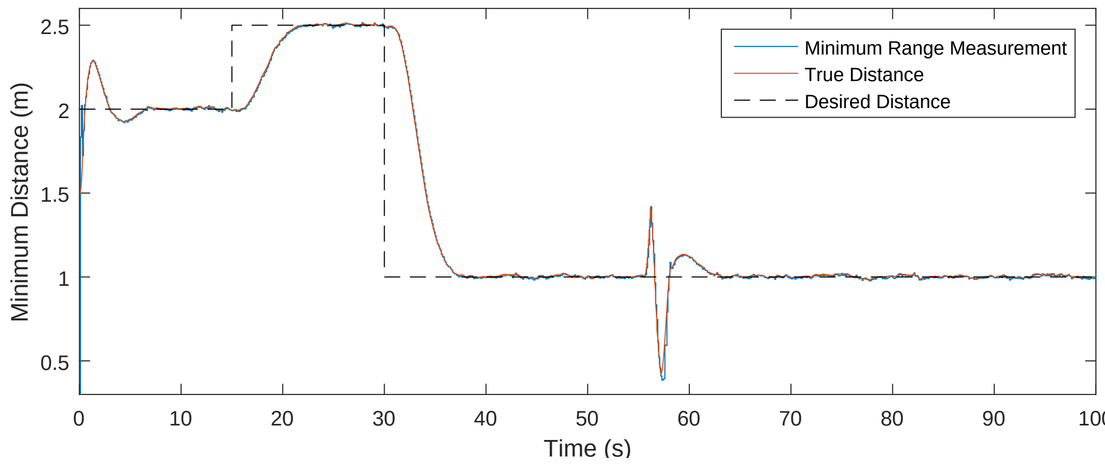

Figure 4.24: The minimum distance to a vertical surface.

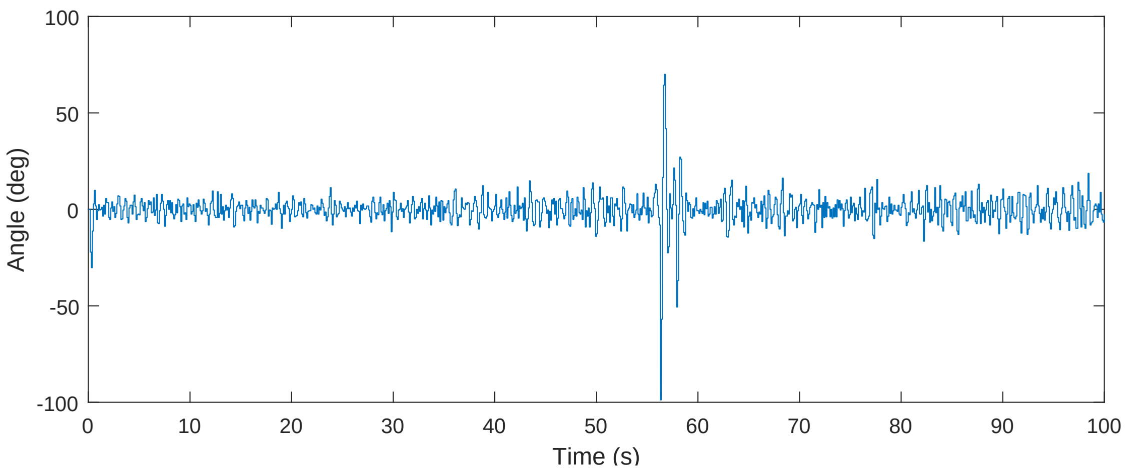

Figure 4.25: The relative yaw angle of the quadcopter while following a vertical surface.

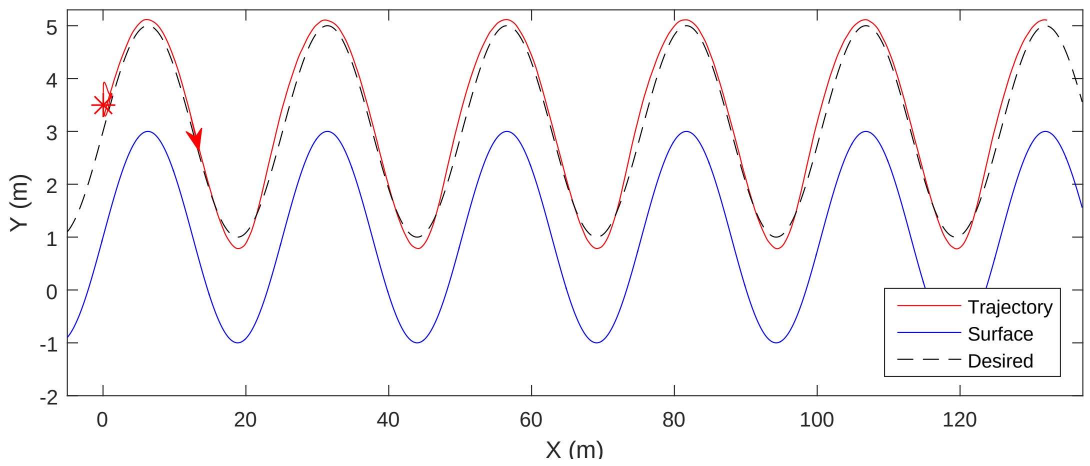

Figure 4.26: The trajectory followed by the quadcopter while maintaining a 2 m distance from a sinusoidal vertical surface, at a velocity of 1.6 m/s along the x_b axis.

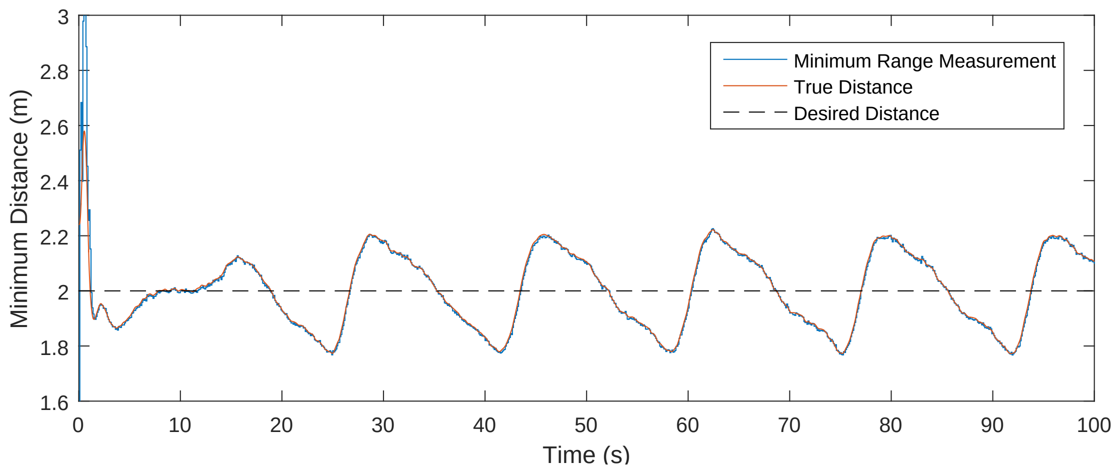

Figure 4.27: The minimum distance to a sinusoidal vertical surface.

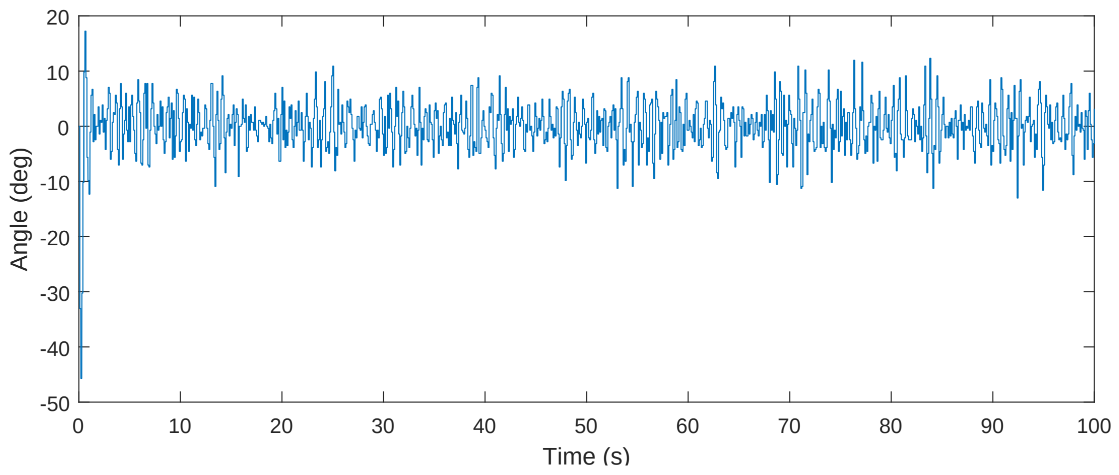

Figure 4.28: The relative yaw angle of the quadcopter while following a sinusoidal vertical surface.

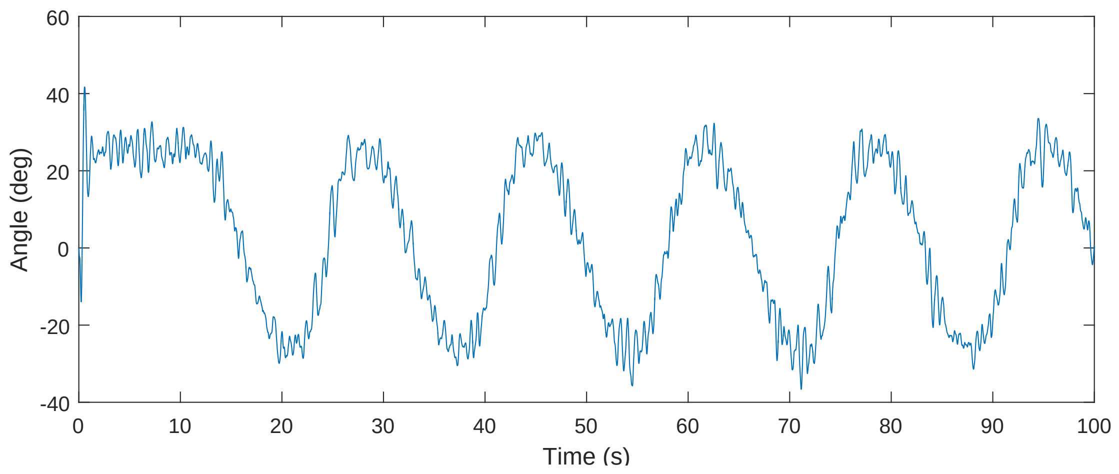

Figure 4.29: The yaw angle of the quadcopter while following a sinusoidal vertical surface.

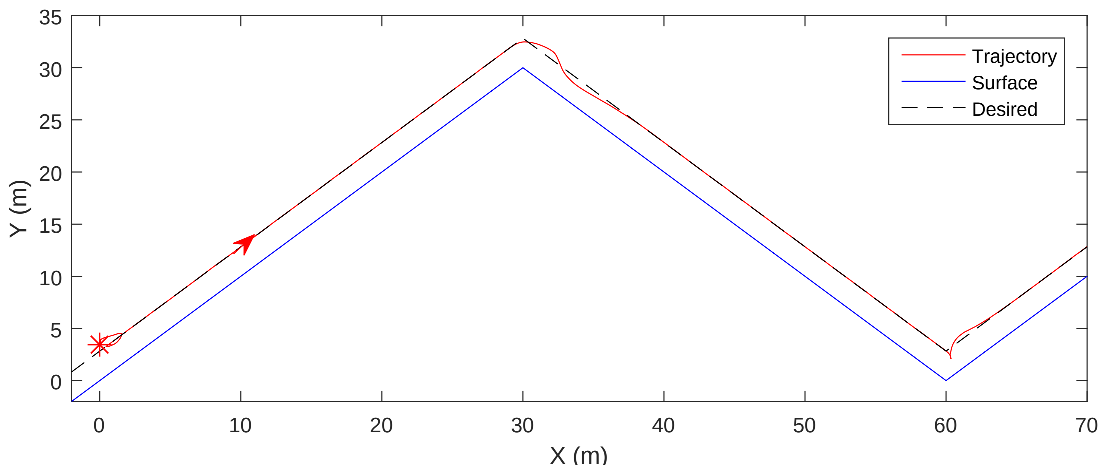

Figure 4.30: The trajectory followed by the quadcopter while maintaining a 2 m distance from a vertical surface while navigating outside and inside a corners, at a velocity of 1.6 m/s along the x_b axis.

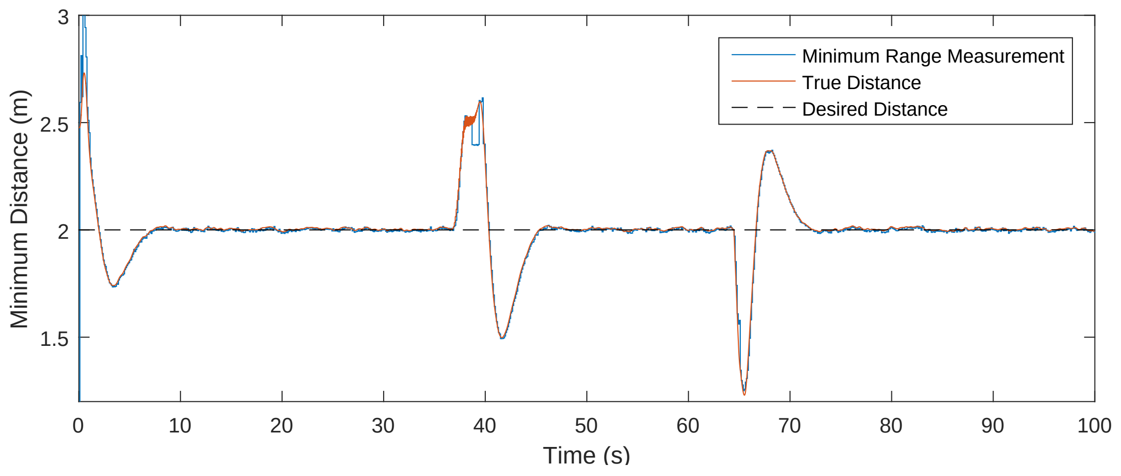

Figure 4.31: The minimum range measurement to a vertical surface while navigating outside and inside corners, at 35 s and 65 s, respectively.

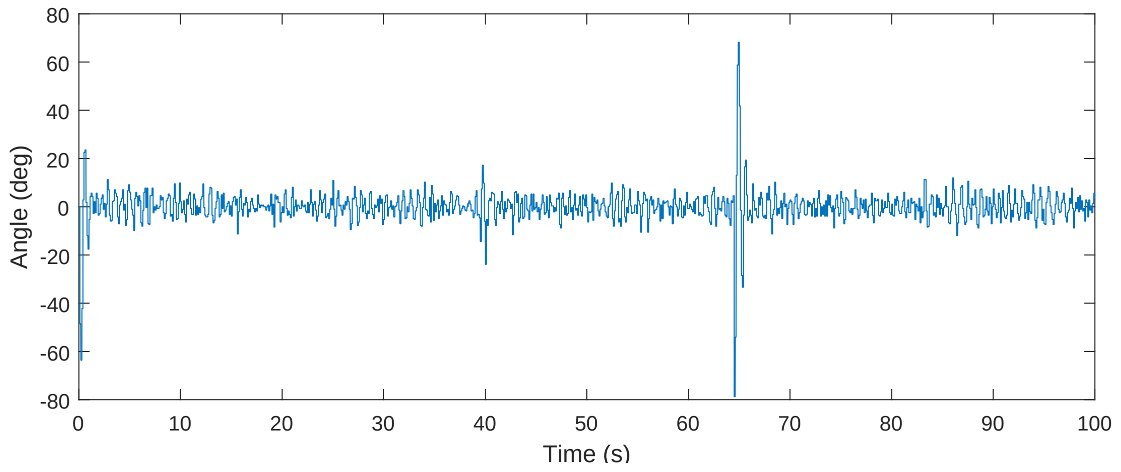

Figure 4.32: The relative yaw angle of the quadcopter while navigating outside and inside corners.

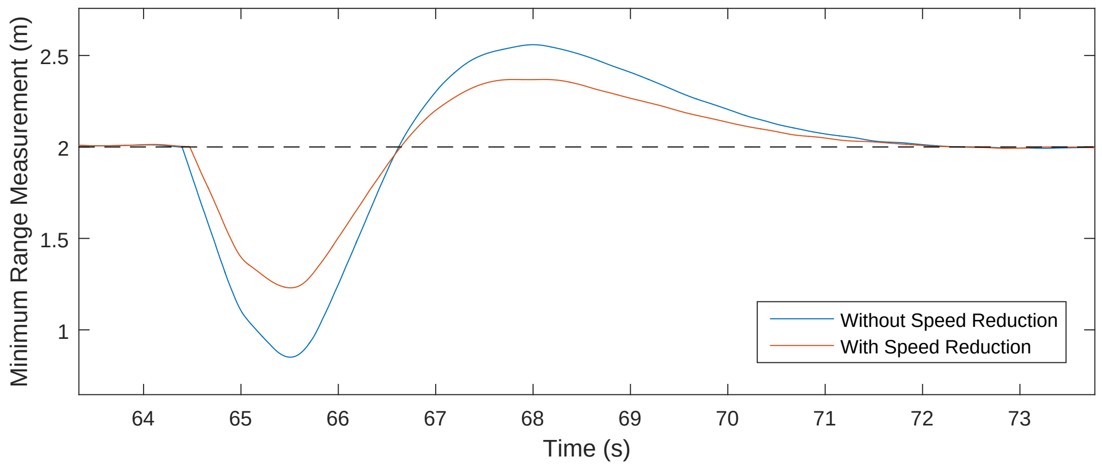

Figure 4.33: A comparison of the performance of the wall following controller while navigating the inside of a corner with and without speed reduction.

Chapter 5

Conclusion

5.1 Research Conclusion

The objective of this thesis is to design a controller to allow a quadcopter to navigate along and maintain a set distance from an unknown vertical surface. With only small deviations from the system's hovering equilibrium assumed, linear control design was considered sufficient.

The design process began with the linearization of the quadcopter dynamic model about the hovering equilibrium point. The design proceeded with the examination of potential field control as a possible solution to the control problem. An initial design was constructed to allow the quadcopter to follow a flat surface. The controller proved to be able to successfully track a flat surface, though when following an irregular shape such as a sinusoidal surface, significant following error appeared due to the controllers inability to adjust the bias command velocity to follow the surface. The potential field control design was then modified to maintain the relative yaw angle between the quadcopter and the surface at zero. This design is able to track a sinusoidal surface and even navigate around corners quite well, though a tracking error can be seen in the following distance when the quadcopter is moving along a flat surface. The error seems to be due to the poor noise rejection of the potential field design.

A wall following controller was designed in an attempt to remove some of the dependence on velocity measurement in the surface following process and improve following performance. The wall following controller performed very well when following a flat surface resulting in no steady-state error. The wall following control design results in relatively large tracking errors when subjected to surface gaps. These errors are to be expected with the current designs and could be avoided with extra processing of the LIDAR data to identify surface gaps. A small tracking error, which varies with speed, is seen when following a sinusoidal surface. This tracking error is expected due to the irregular shape of the surface, which the controller is not designed to account for. A relatively large tracking error is seen when navigating corners. Since corners appear as large disturbances to the controller, these tracking errors are reasonable. The speed reduction technique greatly improves the performance of the wall following controller when navigating the inside of a corner. With the limited scanning range of the LIDAR, the corner can not be detected early enough to fully reduce the speed of the quadcopter. This means with an increase in scanning range, an increase in performance of the controller would occur.

When comparing the three designs, the wall following control design provides the most desirable results. With its robustness to steady-state error and small tracking errors it proves to be the superior surface following control technique.

5.2 Future Work

This thesis does not proved a complete control control scheme ready for deployment to hardware, but rather a simulation study of a surface following control technique. The following are a list of recommendations to be completed in the future before using the control design on real hardware.

1. Develop a scan matching algorithm and state estimation to determine position and velocity measurements from only the LIDAR measurements.

2. Discretize the controller and deploy it to a quadcoter for testing.

References

[1] Hokuyo Automatic. Scanning laser range finder urg-04lx-ug01 specifications. `http://robotshop.com/media/files/pdf/hokuyo-urg-04lx-ug01-specifications.pdf`, 2009.

[2] Reinhard Braunstingl, Jokin Mujika, and Juan Pedro Uribe. A wall following robot with a fuzzy logic controller optimized by a genetic algorithm. In *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE Int*, volume 5, pages 77–82. IEEE, 1995.

[3] Koray Çelik and Arun K Somani. Monocular vision slam for indoor aerial vehicles. *Journal of electrical and computer engineering*, 2013:4–1573, 2013.

[4] Yong-bo Chen, Guan-chen Luo, Yue-song Mei, Jian-qiao Yu, and Xiao-long Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016.

[5] Yongbo Chen, Jianqiao Yu, Yuesong Mei, Yafei Wang, and Xiaolong Su. Modified central force optimization (mcfo) algorithm for 3d uav path planning. *Neurocomputing*, 171:878–888, 2016.

[6] Youngwan Cho and Jae-young Hwang. A study on ekf-slam simulation of autonomous flight control of quadcopter. *International Journal of Software Engineering and Its Applications*, 9(9):269–282, 2015.

[7] Tan Lam Chung, Trong Hieu Bui, Sang Bong Kim, Myung Suck Oh, and Tan Tien Nguyen. Wall-following control of a two-wheeled mobile robot. *KSME International Journal*, 18(8):1288–1296, 2004.

[8] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[9] Albert Diosi and Lindsay Kleeman. Fast laser scan matching using polar coordinates. *The International Journal of Robotics Research*, 26(10):1125–1153, 2007.

[10] Allen Ferrick, Jesse Fish, Edward Venator, and Gregory S Lee. Uav obstacle avoidance using image processing techniques. In *Technologies for Practical Robot Applications (TePRA), 2012 IEEE International Conference on*, pages 73–78. IEEE, 2012.

[11] Yanbin Gao, Shifei Liu, Mohamed M Atia, and Aboelmagd Noureldin. Ins/gps/lidar integrated navigation system for urban and indoor environments using hybrid scan matching algorithm. *Sensors*, 15(9):23286–23302, 2015.

[12] Tanja Hebecker, Robert Buchholz, and Frank Ortmeier. Model-based local path planning for uavs. *Journal of Intelligent & Robotic Systems*, 78(1):127–142, 2015.

[13] Md Didarul Islam, SM Taslim Reza, Jia Uddin, and Emmanuel Oyekanlu. Laser scan matching by fast cvsac in dynamic environment. *International Journal of Intelligent Systems and Applications*, 5(11):11, 2013.

[14] Ankyda Ji and Kamran Turkoglu. Development of a low-cost experimental quadcopter testbed using an arduino controller and software. *arXiv preprint arXiv:1508.04886*, 2015.

[15] Chia-Feng Juang and Chia-Hung Hsu. Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control. *IEEE Transactions on Industrial Electronics*, 56(10):3931–3940, 2009.

[16] Jinhyun Kim, Min-Sung Kang, and Sangdeok Park. Accurate modeling and robust hovering control for a quad-rotor vtol aircraft. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 9–26. Springer, 2009.

[17] Bor-Woei Kuo, Hsun-Hao Chang, Yung-Chang Chen, and Shi-Yu Huang. A light-and-fast slam algorithm for robots in indoor environments using line segment map. *Journal of Robotics*, 2011, 2011.

[18] Frank L Lewis, Chaouki T Abdallah, and Darren M Dawson. *Control of robot manipulators*, volume 236. Macmillan New York, 1993.

[19] Mee-Seub Lim, Joonhong Lim, and Sang-Rok Oh. High-speed wall following and obstacle avoidance of wheeled mobile robots using hybrid behavior specifications. In *Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on*, pages 143–148. IEEE, 2001.

[20] Emanuele Menegatti, Alberto Pretto, Alberto Scarpa, and Enrico Pagello. Omnidirectional vision scan matching for robot localization in dynamic environments. *IEEE transactions on robotics*, 22(3):523–535, 2006.

[21] Qing-hao Meng, Da-wei Liu, Ming-lu Zhang, and Yi-cai Sun. Wall-following by an autonomously guided vehicle (agv) using a new fuzzy-i (integration) controller. *Robotica*, 17(01):79–86, 1999.

[22] János Mészarós. Aerial surveying uav based on open-source hardware and software. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37:1, 2011.

[23] Alireza Nemati, Mohammad Sarim, Mehdi Hashemi, and Manish Kumar. Autonomous wall-following based navigation of unmanned aerial vehicles in indoor environments. In *AIAA Infotech@ Aerospace*, page 0989. 2015.

[24] Pedro Núñez, R Vazquez-Martin, Antonio Bandera, and F Sandoval. Fast laser scan matching approach based on adaptive curvature estimation for mobile robots. *Robotica*, 27(03):469–479, 2009.

[25] M.I. Ribeiro. Obstacle avoidance. Technical report, Institute for Systems and Robotics, 11 2005.

[26] Daniel L Richards. *Open source UAV platform development for aerial photography.* CALIFORNIA STATE UNIVERSITY, LONG BEACH, 2015.

[27] Bartek Roszak and Edward J Davison. The multivariable servomechanism problem for positive lti systems. *IEEE Transactions on Automatic Control*, 55(9):2204–2209, 2010.

[28] Piotr Rudol and Patrick Doherty. Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In *Aerospace Conference, 2008 IEEE*, pages 1–8. IEEE, 2008.

[29] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches.* John Wiley & Sons, Hoboken, New Jersey, 2006.

[30] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[31] PB Sujit and Randy Beard. Multiple uav path planning using anytime algorithms. In *American Control Conference, 2009. ACC'09.*, pages 2978–2983. IEEE, 2009.

[32] A Tayebi and S McGilvray. Attitude stabilization of a four-rotor aerial robot. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1216–1221. Ieee, 2004.

[33] Sebastian Thrun, Daphne Koller, Zoubin Ghahramani, Hugh Durrant-Whyte, and Andrew Y Ng. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In *Algorithmic Foundations of Robotics V*, pages 363–380. Springer, 2004.

[34] Hendri Himawan Triharminto, Oyas Wahyunggoro, Teguh Bharata Adji, and Adha Imam Cahyadi. An integrated artificial potential field path planning with kinematic control for nonholonomic mobile robot. *International Journal on Advanced Science, Engineering and Information Technology*, 6(4):410–418, 2016.

[35] Fredy Tungadi, Wen Lik Dennis Lui, Lindsay Kleeman, and Ray Jarvis. Robust online map merging system using laser scan matching and omnidirectional vision. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 7–14. IEEE, 2010.

[36] Daobin Wang, Huawei Liang, Tao Mei, Hui Zhu, Jing Fu, and Xiang Tao. Lidar scan matching ekf-slam using the differential model of vehicle motion. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 908–912. IEEE, 2013.

[37] Er-yong Wu, Gong-yan Li, Zhi-yu Xiang, and Ji-lin Liu. Stereo vision based slam using rao-blackwellised particle filter. *Journal of Zhejiang University-Science A*, 9(4):500–509, 2008.

[38] Teruko Yata, Lindsay Kleeman, and Shin'ichi Yuta. Wall following using angle information measured by a single ultrasonic transducer. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1590–1596. IEEE, 1998.

[39] Xiaoping Yun and Ko-Cheng Tan. A wall-following method for escaping local minima in potential field based motion planning. In *Advanced Robotics, 1997. ICAR'97. Proceedings., 8th International Conference on*, pages 421–426. IEEE, 1997.

[40] Erik Zamora and Wen Yu. Recent advances on simultaneous localization and mapping for mobile robots. *IETE Technical Review*, 30(6):490–496, 2013.

[41] Baochang Zhang, Zhili Mao, Wanquan Liu, and Jianzhuang Liu. Geometric reinforcement learning for path planning of uavs. *Journal of Intelligent & Robotic Systems*, 77(2):391–409, 2015.

[42] Xiong Zhiyong, Yang Xiuxia, Zhang Yi, Hua Wei, and Zhou Weiei. Study on dynamic guidance obstacle avoidance for uav. *International Journal of Future Generation Communication and Networking*, 9(8):293–306, 2016.