Terrain Tracking Motion Planning and Control of Unmanned Aerial Vehicles

by

Nasser Ayidh AlQahtani

B.Sc., Qassim University, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

June 2017

The undersigned certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis entitled:

Terrain Tracking Motion Planning and Control of Unmanned Aerial Vehicles

submitted by Nasser Ayidh AlQahtani in partial fulfilment of the requirements of the degree of

Master of Applied Science.

Dr. Homayoun Najjaran, School of Engineering
Supervisor, Professor (please print name and faculty/school above the line)

Dr. Yang Cao, School of Engineering
Supervisory Committee Member, Professor (please print name and faculty/school in the line above)

Dr. Rudolf Seethaler, School of Engineering
Supervisory Committee Member, Professor (please print name and faculty/school in the line above)

Dr. Jahangir Hossain, School of Engineering
University Examiner, Professor (please print name and faculty/school in the line above)

External Examiner, Professor (please print name and university in the line above)

(Date Submitted to Grad Studies)

Additional Committee Members include:

(please print name and faculty/school in the line above)

(please print name and faculty/school in the line above)

Abstract

In a modern society, autonomous quadrotors can be used to perform tasks and collect data in dangerous and inaccessible environments where human involvement would traditionally be necessary. Unmanned Aerial Vehicles (UAVs), and especially the quadrotor, are still facing obstacles in terms of following a trajectory and flying autonomously in enclosed, complex or GPS denied areas. This work presents a novel framework for navigating a quadrotor over undulated terrains, so it can be of great importance for the use of UAVs in civilian applications such as monitoring of pipes, bridges and buildings. The proposed approach involves the use of a single-beam LiDAR to estimate the terrain profile under uncertainty. The LiDAR is installed at the base of a quadrotor and can be set at different angles to send information to the quadrotor about undulations of the terrain ahead. This strategy helps the quadrotor to build a smooth trajectory for the UAV and allows the controller to follow it closely. In turn, maneuverability of UAV over and around ground obstacles is improved in comparison to default autopilot controllers programmed to maintain a UAV at a given altitude. Through simulation, the result shows how this alternative technique with motion planning algorithms improves the performance of the quadrotor. In this thesis, the method used in the simulation was tested using a single-beam LiDAR reading the undulations of the terrain beneath and ahead of the quadrotor at a desired altitude. Then, motion planning algorithms, such as Gaussian filter and cubic spline, plan a smooth trajectory so that the quadrotor can avoid having to make any sudden turn or recoveries. Finally, this planned trajectory is provided to an appropriate controller, such as the model predictive control, linear quadratic regulator, proportional integral derivative control which help the quadrotor to not only follow the terrain exactly, but also to minimize energy wasted on sudden recoveries.

Preface

All work presented here was supported and funded by Qassim University and conducted in the advanced control and intelligent system (ACIS) laboratory at the School of Engineering within University of British Columbia under the supervision of Prof. Homayoun Najjaran.

The contributions of this thesis are highlighted as follows:

- The problem formalization and the proposed solution in this thesis were published as an abstract in the Canadian Society for Mechanical Engineering International Congress 2016 (CSME 2016) under the title" Simulation-Based Study of Terrain Following Using an Unmanned Aerial Vehicle".
- A version of Chapter 2, Chapter 3 and Chapter 4 were published and presented in the SAI Intelligent System Conference 2016 in London, UK. The title of the paper is ''Motion Control of a Terrain Following Unmanned Aerial Vehicle under Uncertainty". This paper has also been selected to be a chapter in a book titled ''Lecture Notes in Network and Systems'' under the Springer publisher.
- The work presented in Chapter 1, Chapter 2, Chapter 3 and Chapter 4 has been accepted in the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC2017). The title of the paper is ''Adaptive Motion Planning for Terrain Following Quadrotors".
- Lastly, a version of Chapter 1, Chapter 2, and Chapter 5 have been accepted in the Future Technologies Conference (FTC 2017) under the title ''Control System of a Terrain Following Quadrotor under Uncertainty and Input Constraints: a Review and Research Framework".

I, Nasser AlQahtani, was one of three authors for the first three papers and was lead investigator, responsible for conducting research, developing algorithms, programming, and

writing the papers. Bara Emran, PhD candidate, at the ACIS laboratory was the second author. He contributed to the manuscript edition as a mentor and was also involved in the early stages of the concept formation. The third author and supervisor of the project was Homayoun Najjaran, involved throughout the project as manuscript editor and by assisting with concept formation. I, Nasser Alqahtani, investigated the final paper with Prof Najjaran contributing to the manuscript and concept formation. This thesis includes the prepared conference material. Where figures and tables from the papers have been used, they have been cited. Texts from the papers have also been used and cited throughout this thesis.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Definition
UAVs	Unmanned Aerial Vehicles
LiDAR	Light Detection and Ranging
IMUs	Inertial Measurement Units
STARMAC	Testbed of Autonomous Rotorcraft for Multi Agent Control
OS4	Omnidirectional Stationary Flying Outstretched Robots
SLAM	Simultaneous Localization and Mapping
GPS	Global Positioning System
SPA	Sense Plan Act
PID	Proportional Integral Derivative
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control

List of Symbols

$X_i\ Y_i\ Z_i$	Position in earth coordinate frame
$X_b\ Y_b\ Z_b$	Position in body coordinate frame
$\mathbf{\Gamma^E}$	Vector of linear position in earth coordinate frame
$\mathbf{\Theta^E}$	Vector of angular position in earth coordinate frame
ϕ	Roll Angle
θ	Pitch Angle
ψ	Yaw Angle
$\mathbf{V^B}$	Vector of linear velocity in body coordinate frame
$\boldsymbol{\omega^B}$	Vector of angular velocity in body coordinate frame
ξ	Composed of the quadrotor linear and angular position
$\dot{\xi}$	Generalized velocity vector of a quadrotor with respect to earth frame
ν	Generalized velocity vector of a quadrotor with respect to body frame
J_Θ	Generalized matrix
R_Θ	Rotational matrix
T_Θ	Translational matrix
b	Thrust factors
d	Drag force
$\boldsymbol{F_i}$	Vertical forces of ith motor
$\boldsymbol{M_i}$	Moments
$\boldsymbol{\Omega}$	Propellers' speed vector
Ω_i	Propellers' speed of ith motor
U	Total body force
$\boldsymbol{\tau_\phi}$	Roll torque
$\boldsymbol{\tau_\theta}$	Pitch torque
$\boldsymbol{\tau_\psi}$	Yaw torque
m	Mass
$\boldsymbol{I}$	Inertia matrix
$I_x\ I_y\ I_z$	Mass moment of inertia of each axis
g	Gravity

L	Horizontal distance from the center of propeller to CoG
t	Time
t_0	Initial time
t_f	Final time
q_0	Initial position in configuration space
q_f	Final position in configuration space
v_0	Initial velocity
v_f	Final velocity
q_{min}& q_{max}	Limitation on the configuration space
$\dot{\theta}_{min}$& $\dot{\theta}_{max}$	Constraint on the rotational speed of the electric motors
$\boldsymbol{\tau}_{min}$& $\boldsymbol{\tau}_{max}$	Torque constraint
μ	Mean of Gaussian filter

Acknowledgements

As a scientist and an engineer, I crossed the ocean to Canada seeking knowledge that would bring me closer to Allah. All praises are due to Allah for helping me to complete this work and for giving me the ability and patience to overcome all the challenges of my graduate studies. O Allah, I ask You for knowledge that is beneficial, valuable, which can be used for the greater good of mankind, and deeds that will be accepted.

I would like to extend my gratitude to my supervisor Professor Homayoun Najjaran for believing in me. His supervision style opened my eyes and extended my horizon in the field of robotics and control. It was an honor and pleasure to work under him and I am looking forward to extending our work beyond this Master's degree. The acknowledgements would not be complete without mentioning my gratitude to and appreciation of my colleague and wonderful mentor Bara J. Emran for sharing my challenges and difficulties and for all our memorable academic discussions. A big word of appreciation goes to my committee members Dr. Yang Cao and Dr. Rudolf Seethaler for taking the time to read and comment on my thesis.

I would also like to extend a special acknowledgement to Qassim University, my sponsor, and to Saly Michael, my academic advisor at the Saudi Arabian Cultural Bureau, for their invaluable assistance.

To my friends in Kelowna, I would like to thank you for your support and encouragement. Special thanks goes to my friend Mohamed Hegazi for working long hours with me and sharing tea breaks. Also to my Arabic community, Mohammed Yifia, Aziz Alghamdi, Mohamed Alhashimi, Abdul Rahman Masoud, Fawaz Altamimi, and my office mate Anas Issa, I have enjoyed every moment we have shared – thanks for your support.

It is impossible for me to express how much gratitude I owe to my parents, Huda and Ayidh, for their incalculable love, prayers and support. I have never felt for one moment alone, knowing that they exist in the world. I would like for them to know that I understand and appreciate the sacrifices they have made for me along the way, and that I love them very much. To my brothers and sisters for their unfailing support, guidance and protection – a million times thank you. My family with me here in Canada - my wife and son – you honor me by sharing this journey and it would not have been nearly so enjoyable without you. Thank you for accepting and supporting my goals and for being there for me unconditionally.

To my parents, Huda and Ayidh,
for their enduring support, and
to my son Omar, my inspiration.

Chapter 1: Introduction

1.1 Motivation

It is indisputable that autonomous vehicles including unmanned aerial vehicles have become a part of modern society, and that they have the potential to shape the future in a positive way [1, 2]. In particular, unmanned aerial vehicles (UAVs) were originally developed and tested for military purposes though today they are one of the most popular types of unmanned vehicles used by both ordinary citizens as well as researchers and engineers [3]. Civilians generally use UAVs for aerial photography and videography, while scientists have found them extremely useful for applications that would not normally be possible without human involvement. Interest in UAVs as a research tool has grown in parallel with outstanding innovations in the sensor industry. Inventions such as wireless communication and single board computers have dramatically contributed to the capabilities of UAVs [4], and in recent years, the cost of such technologies has dropped dramatically while at the same time their efficacy has vastly improved. These improvements in cost and efficiency mean that today UAVs continue to offer the potential to fill gaps in both civilian and military applications.

There are two main types of UAV design: fixed wing and multi-rotor. On account of their affordability, reliability and flexibility, quadrotors, from the multi-rotor UAV category, have become a popular research option for scientists and engineers [3]. A quadrotor is a much more versatile and stable flying structure than the traditional helicopter thanks to its simple form: four rotors attached to each end of a cross shaped body. Aside from its stability, other quadrotor benefits are its ability to fly vertically, land in small spaces, and hover close to the ground or other objects. Also, unlike fixed wing structures, quadrotors can be made in miniature sizes that are safe to fly indoors. The quadrotor's versatility and stability when performing intricate maneuvers has encouraged researchers to use these UAVs for real-world

operations such as mapping difficult terrain, search and rescue, surveillance, and monitoring gas and oil pipelines [5]. A main disadvantage of this type of UAV, however, is the amount of energy it consumes powering four motors.

One of the biggest influencers on the evolution of UAVs has been the role of navigation sensors including inertial measurement units (IMUs) and global positioning system (GPS). These devices now make it possible for UAVs to fly autonomously while keeping track of their position. In parallel with the capabilities of the quadrotor itself, integrating of additional sensors into UAVs has advanced significantly. For example, the computer vision communities now have an excellent platform for testing their algorithms on tasks like tracking objects and avoiding obstacles [6]. However, despite these gains, some challenges remain. One of the biggest challenges currently facing researchers, in terms of quadrotor flight capabilities, is how to make precision navigation at low and high altitudes possible within environmental constraints [4]. The difficulty of this scenario lies in that it not only requires a sophisticated motion planning algorithm and a reliable control technique [7], but also sufficient data about the environment.

The motivation for this thesis, therefore, was to solve the obstacles presented by this quadrotor usage scenario or, more specifically, to develop a novel navigation framework which would allow the UAV to autonomously follow unknown terrain while maintaining a certain distance from it, within environmental and energy consumption restraints.

The objective of this thesis was to develop a software for a new quadrotor navigational method and then present the impact it has had on the quadrotor's performance. The method combines the addition of a forward facing LiDAR sensor with an analysis of the effectiveness of existing control systems on this type of mission. The proposed method

involves connecting a single-beam LiDAR sensor to the base of the quadrotor in order to retrieve reliable and detailed information about the undulations of the terrain ahead. The sensor then feeds back this information to the quadrotor so that its controller can create a suitable trajectory and ensure a smooth flight-path.

1.2 Literature Review

The purpose of this section is to provide a brief overview of existing quadrotor use scenarios in the relevant literature. It also aims to present topics and discussion pertaining to this thesis including sensing, motion planning algorithms, and control techniques.

1.2.1 Sensing and System Architecture

A quadrotor is generally equipped with various onboard sensors to enable it to fly autonomously. Quadrotor sensors can be categorized as proprioceptive or exteroceptive [8]. Proprioceptive sensors, (e.g., inertial measurement units, IMUs), provide the measurements or estimates necessary to make it possible for a quadrotor to fly when paired with human interaction. The issue with this type of sensor is that it does not give enough information to enable autonomous flight or long term state estimation [8]. Exteroceptive sensors, such as laser scanners and cameras, integrate with proprioceptive sensors to enhance the state estimation ability of the systems. In recent years, both military and civil applications have seen an uptake in the use of multisensory data fusion techniques [9]. These techniques provide more accurate and specific inferences, by merging data and related information from multiple sensors, than would be possible using data from just one source. Camera and laser rangefinder sensors work very well for modern applications and, therefore, have become popular in the field of UAVs. Consequently, autonomous quadrotors are categorized into those with a laser-based autonomous flight approach and those with a vision based approach.

Before presenting the usage of these approaches in the literature, it is necessary to point out the differences in the system architecture of these two types of quadrotors, and the impact this architecture has on their autonomous flight performance, or in other words, the state estimation capabilities of each quadrotor.

A quadrotor is based on a simple mechanism, but what makes it an outstanding flying vehicle is its ability to perform extreme maneuvers and accommodate onboard devices. Depending on the device variations, quadrotors can range from basic electronic components like the Parrot AR Drone 2.0 to fully developed flying vehicles such as the STARMAC test bed at Stanford University [10]. The basic overview of the quadrotor structure, as shown in Figure 1.1, includes a mechanical frame, a microcontroller, actuators, and sensors. Using these primary components, the quadrotor is able to perform certain tasks. However, this basic structure can be modified based on the user's goal. The Parrot AR Drone, as an example, is one of the cheapest drones on the market. It is equipped with limited actuator and sensor features. Its microcontroller has also been designed for non-professional users in that the pilot does not require any background in operating flying vehicles. The reason for this low-budget and user-friendly approach is that the main use of the Parrot AR Drone 2.0 is videotaping and photographing, and sometimes the prototyping of engineering courses. The drawback of this machine is that it is not designed for developing algorithms such as trajectory planning. In terms of research platforms, the Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC) is an example of how the quadrotor structure can be modified for research goals. STARMAC is basically the same structure as the Parrot AR Drone 2.0, but its components have distinct features. The power of its actuators can lift heavy payloads and increase flight time, usually one of the main drawbacks

of quadrotors. STARMAC is also equipped with sensors that allow it to avoid obstacles and follow an object. More importantly, its microcontroller is designed for developing algorithms such as those for control and trajectory planning. Hence, as a result of the increasing interest in unmanned aerial vehicles, the structure of the quadrotor is gradually becoming one of the most sophisticated embedded systems available today.

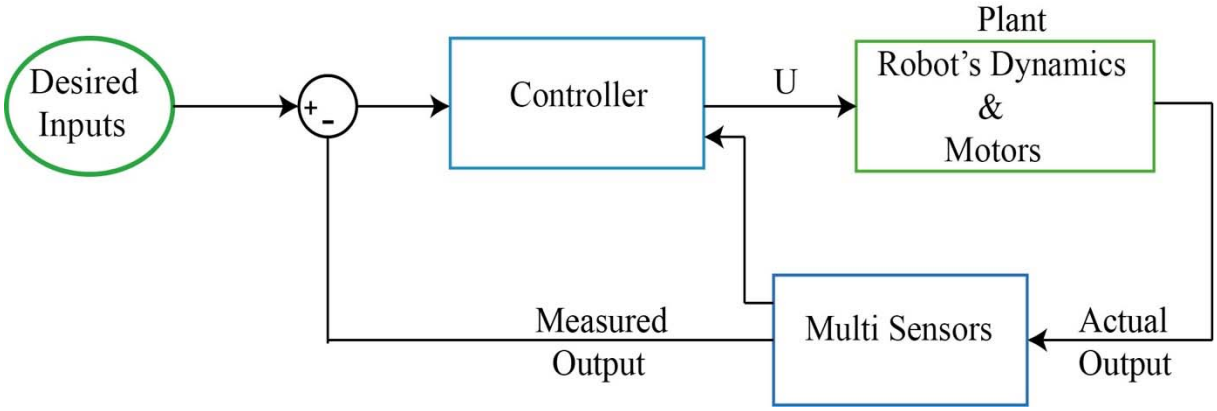

Figure 1.1: The general overview of quadrotor structure.

UAVs' autonomous flight capabilities in various environments have recently been studied extensively and divided into indoor and outdoor applications. Readers interested in ongoing research on the autonomous navigation approach for UAV platforms are referred to a survey in [11]. An early example of flying a quadrotor autonomously by utilizing a visual feedback system as a primary source of estimation was conducted by Professor Erdine Altuge and his group in GRASP Lab [12, 13]. They used a visual system, a camera on the ground, to estimate the position and orientation of the quadrotor. The offboard controller was responsible for gathering data and processing the images, before setting and sending goals to the quadrotor's onboard controller. Their primary goal was to enhance state estimation and apply feedback linearization and backstepping control techniques. From a navigational perspective, it is worthwhile to note its limitations: Erdine's work used a nearby processing unit, which meant that in the case of a lost connection, even for a short time, there was

potential for the quadrotor to crash. However, his work did encourage the UAV community to build quadrotors fully equipped with expensive onboard measurement instruments such as Omnidirectional Stationary Flying Outstretched Robots (OS4) [14]. Today, the number one obstacle faced by onboard sensors is their ability to supply the quadrotors with high-fidelity, real-time environmental data [15].

One of the challenges for UAVs in both indoor and remote outdoor locations with full onboard sensing is that Global Positioning System (GPS) and other positioning systems, such as motion capture, are often unavailable, therefore limiting the UAVs' capabilities when operating in these areas. UAVs flying in remote or environmentally complicated areas are able to use dead reckoning for positioning, although these measurements over time are not precise. Simultaneous localization and mapping (SLAM), on the other hand, projects a map of the terrain while at the same time estimating the vehicle's position on it. While SLAM algorithms have seen powerful improvements in terms of accuracy and drift-free measurements for these complicated environments, the algorithms have concentrated mainly on autonomous underwater and ground vehicles. The bids to utilize similar algorithms for UAVs have not been as successful because of both the unstable structure of quadrotors and their limited payload capabilities for carrying sensing and computing equipment [16]. Because of the variety of sensors available and the different possible uses for quadrotors, the research community has reached a point where the design of the quadrotor comes to depend on its application or goal and, therefore, the mission needs to be clear even before starting work on the quadrotor's design. In the next section, we aim to present work in the area of navigation for GPS denied environments.

1.2.1.1 Navigation on GPS Denied Environments

Despite visible progress in the literature of modern UAV functionality, authors in [17] emphasize that there would be even greater potential for contemporary applications if UAVs had the ability to navigate autonomously, without using a GPS system for outdoor or motion caption for indoor environments. Currently, most of the research on UAVs or quadrotors is based on estimations of the vehicle's position and orientation taken mainly from GPS or off board units. Thus, in the last ten years, the issue of GPS-free navigation has become a hot topic within the UAV community, especially for those involved with building Micro Aerial Vehicles (MAVs).

One of the early research projects on autonomous quadrotor flight in GPS denied environments was by a group at MIT [6, 16]. They proposed using laser as a primary sensor in conjunction with SLAM and an obstacle avoidance technique. Their experiment was conducted in indoor and outdoor areas. Similar to this work, authors in [17-19] utilized laser scanners as a primary sensor. In [18], authors were able to improve the 3D sensing by using mechanized planning laser scanners. While all these projects made notable progress in fully autonomous flight in GPS denied environments by combining the data from the laser scanner with IMUs, their work relied on having a *prior* knowledge of the surroundings uploaded to onboard sensing systems. The weight of the 2D or 3D laser scanner also posed a problem in terms of limiting the payload capacity of the vehicles: using a heavy scanner added constraints to the agility of the quadrotor [20]. Another study carried out by the computer vision community recommended foregoing the laser scanner and instead using an onboard camera as the primary sensor by reason of its capacity to generate large amounts of data. In other experiments, authors in [21] utilized an optical flow–based velocity estimator while

researchers in [22] used stereo vision based state estimators. A monocular SLAM framework also became the UAV community's preferred technique when using a camera as a primary sensor [23]. However, even though the use of the camera as a sensor has developed at a faster rate than the use of laser, the success of this tool is based on assumptions of a slowly changing environment rather than a completely unknown environment [8].

In the studies mentioned above, it is worthwhile to note that the position of the laser scanners used was horizontal, as in [18], while the cameras were horizontal or downward facing as in [24]. When positioning sensors horizontally, as all the previous research examples have done, the primary mission is to avoid obstacles; whereas using a different position, such as forward-down facing, would allow information about real-time changes in unknown environments to be collected. None of the previous research projects proposed positioning the laser scanner or LiDAR sensor at the bottom of the quadrotor at a forward facing angle to allow information about the unknown region ahead of the quadrotor to be gathered. In this thesis, we claim that positioning the LiDAR sensor in precisely this way helps a quadrotor to navigate unknown terrain at a low altitude without requiring any *prior* knowledge of the environment.

1.2.2 Motion Planning

Directing a robot to navigate autonomously around a space without colliding with anything is what is defined as motion planning. The original formation of this planning method was called the piano mover's problem and it outlined an imaginary scenario where a complicated piece of furniture could be moved unimpeded through a cluttered house. This programming of robots to do geometrical reasoning about their environments, create plans from the information gathered, and then execute the plans autonomously has been a recurring theme in

robotics over the last few decades. A short history of motion planning saw it clearly defined in the 1970s with some solutions perfected for specific situations in the 1980s, while the 1990s offered modern industrial problems some inelegant but practical motion-planning solutions. Since the turn of the century, robotics and automation have found many uses for motion-planning algorithms and these have often been used in applications far beyond what was possible in the 1990s, including in the virtual prototyping [25]. More on the background of motion planning algorithms, including information on many of the basic concepts, can be read in Latombe's textbook [26]. More recent motion planning algorithms and techniques are covered in The "Planning Algorithms" book by LaValle [27]. The more recent surveys by authors in [28-30] offer information on existing algorithms for deterministic and uncertain environments [31].

Motion planning techniques developed effectively for ground vehicle applications have not seen the same success with UAV applications because of their unstable systems [10, 16]. Researchers have been challenged many times by motion planning and trajectory generation problems when using UAVs for specific situations such as [6], [10], [16], and the resulting innovations have led to some progressive solutions for UAVs including those enabling autonomous flight. One breakthrough innovation made use of a GPS system which told the quadrotor where and when to arrive at set points as part of a responsive time-scheduled map [32]. As good as this method was for mapping and monitoring, however, it still did not provide a comprehensive solution for applications where the environment was not easily accessed, i.e. for remote or indoor tasks. Another innovation improved the autonomy of the quadrotor by adding a sensor to the UAV; for example, a camera, and creating an algorithm to generate a trajectory based on the information from the sensor. Two new issues surfaced in

this case nevertheless: it exposed both payload limitations and long computational processes [16]. Recently, a common method as stated in [10, 33] is to identify the robot configuration as a point in a potential field which incorporates attraction to the objective and repulsion to the obstacles, resulting in a trajectory or path. The advantage of this method is that it produces a trajectory without any complicated computation. Figure 1.2 shows the general structure of motion planning with considering a *prior* knowledge when generating the trajectory. In this thesis, it has been assumed the robot as a point in the configuration space as a result of reducing the computing, and the same structure of motion planning has been used without considering environment model.

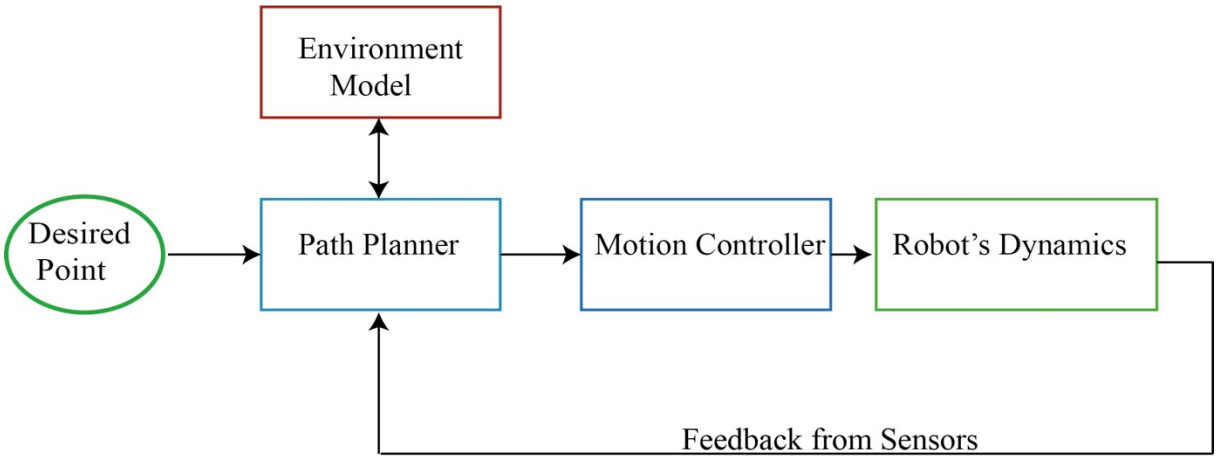

Figure 1.2: The general overview of motion planning structure.

Generally speaking, there are obvious deficiencies in basic path planning when looking at how robotic systems typically use computed paths. Effective autonomous systems must act accordingly on data which is continually received, as seen in the sense–plan–act (SPA) paradigm. Figure 1.3 presents the general overview of computing plan [34].

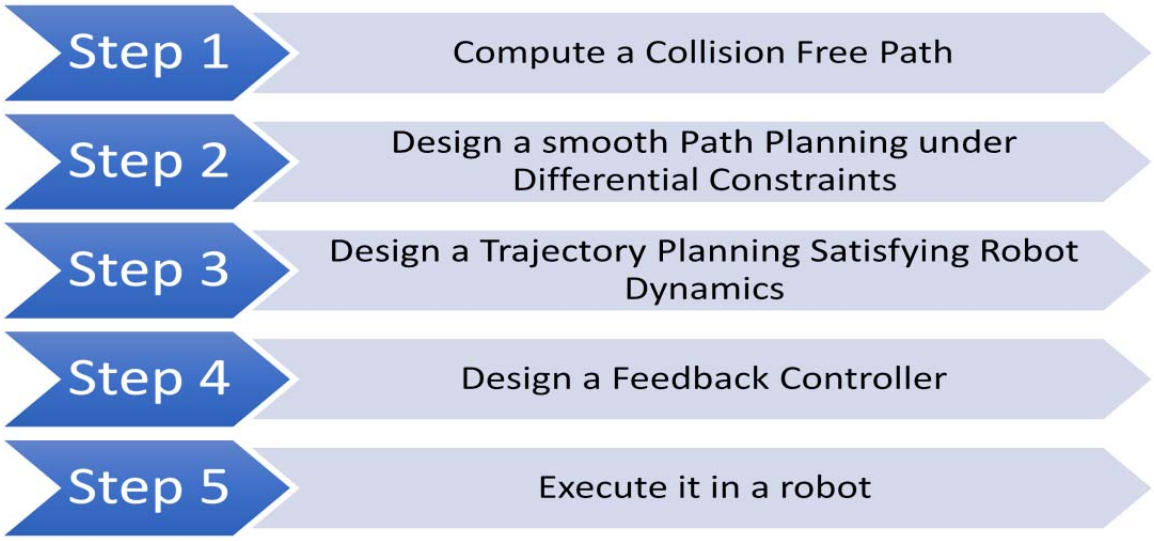

Figure 1.3: Steps of computing a plan for a robot.

1.2.3 Control

Developing a cost-efficient and precise control technique was the main challenge facing researchers wanting to use quadrotors as a research platform. Although the first stage of development was modeling and controlling the dynamics of the rigid body and motors separately [35], researchers have now achieved a full nonlinear control system within the quadrotor's limitations. For example, Holger Voos published a paper on nonlinear control of a quadrotor using feedback linearization, and the result was successful, but limited [36]. To illustrate this point, the maximum control input is limited to the maximum thrust of the quadrotor's motors, which means that having a reliable control system responsive to specific tasks is a critical step in the process. Hence, the development of a suitable control technique that allows for precision navigation at low and high altitudes is currently a very interesting research topic.

1.3 Statement of Contributions

This work focuses on developing a navigation system that allows the quadrotor to autonomously follow unknown terrain while maintaining a certain distance from it, within environmental and energy consumption restraints. More information about our contribution is listed below:

- We introduced a new framework for a quadrotor to enhance its state estimation while navigating in unknown terrain, at a certain distance from it and under differential constraints. Using this new method, it is possible for a quadrotor to follow unknown terrain without utilizing the existing technique of uploading *prior* knowledge about the terrain or environment to the quadrotor's controller. The new framework includes: (i) the integration of a LiDAR with the quadrotor's system architecture in a way that feeds the quadrotor information about the undulation of the terrain ahead. (ii) the introduction of a Gaussian filter as a path planning method which uses the new information and tests the validity of the framework. (iii) the design of a Proportional Integral Derivative (PID) quadrotor controller to track the Gaussian trajectory and validate the whole framework.

- We introduced alternative motion planning algorithms. These alternative algorithms take into account the differential constraints. In this thesis, we assume that a quadrotor flies a certain distance from the terrain and that the velocity of the quadrotor is constant. Thus, the success of the new quadrotor framework relies on the smoothness of the motion planning that is based on those differential constraints. This smooth motion planning will help the quadrotor to have a collision-free and long flight as a result of knowing the terrain.

- Alternative optimal control techniques are introduced. These alternative algorithms take into account the quadrotor's limitations, especially the quadrotor inputs. The aim of the new framework is that it helps the quadrotor to follow the terrain precisely while avoiding any sudden recoveries and more importantly minimizing the control input energy. These criteria are solutions of one of the main drawbacks of a quadrotor which is the short flight time.

- We developed a simulation package both to examine the performance of the new framework and to offer an educational tool for control and robotic engineering students. The simulation package is an interactive tool designed to help students understand motion control and optimal control concepts.

1.4 Thesis Organizations

This thesis is organized into six chapters as follows:

- Chapter 1 presents the motivations of this thesis and a statement of contribution, after giving a literature review.

- Chapter 2 includes the fundamentals of the quadrotor system relating to this thesis. This chapter focuses on the dynamic behavior of a quadrotor, then presents the equation of the motion. It concludes with the problem that the thesis is trying to solve.

- Chapter 3 includes the development of two methods of motion planning: splines and Gaussian filters. The former solves the motion planning problem while considering the differential constraints, while the latter solves the problem without considering the differential constraints. This chapters concludes with a presentation of the result of these two methods acting on a Step terrain profile.

- Chapter 4 includes three control techniques: PID, Linear Quadratic Regulator (LQR), and Model Predictive Control (MPC). The main aim of this chapter is to present the optimal control techniques to minimize the quadrotor's energy consumption by avoiding any sudden recoveries.
- Chapter 5 validates the idea of the thesis in five scenarios. It also presents the simulation programme that has been developed for this research.
- Chapter 6 includes the conclusion of this work and comments on future work.

Chapter 2: A Quadrotor System

2.1 Overview

The work presented here focuses on an autonomous navigation method for a quadrotor to follow a terrain in different scenarios where there is a lack of information about the environment. The task becomes even more challenging if a long flight time is required or if the quadrotor need to maintain its altitude a certain distance from the outdoor terrain without relying on GPS data. To make it possible for a quadrotor to do such a task, an understanding of quadrotor dynamics and system architecture is required. Next, utilizing these concepts and equations of motions, control and motion planning algorithms can be developed to lead the flying vehicle across unknown terrain without requiring any major corrections. This chapter briefly discusses three main topics: the dynamic model of a quadrotor, its system architecture, and the problem statement.

This chapter starts with the dynamic model of a quadrotor. As a result, it contains most of the mathematical equations that describe the quadrotor movements. The development of the control algorithms that will be discussed in Chapter 4 and motion planning algorithms that will be discussed in Chapter 3 will also be based on these equations. However, it is important to point out that the main point of this thesis is not to develop the modeling of the quadrotor. Consequently, the readers interested in the kinematics of a quadrotor, presented in Section 2.2.1 and equation of motions discussed in Section 2.2.4, should refer to [2, 37]. The best literature on the inputs of the quadrotor presented in Section 2.2.2 and linearization of the nonlinearity of the quadrotor discussed in Section 2.2.4 is by Professor Vijay Kumar's in his online course in Coursera and in [1, 38, 39].

The rest of this chapter will mainly tackle the terrain following problem. It begins by describing the system architecture of the quadrotor as discussed in Section 2.3, and then elaborates on the problem in terms of constraints and difficulties as presented in Section 2.4.

2.2 Dynamic Model of a Quadrotor

The modeling of the quadrotor is an important initial step in the control design. Understanding the dynamics of the quadrotor and the principle of the quadrotor's movements allows the designer to utilize this information to achieve their goal. A quadrotor is a perpendicular cross configuration with motors mounted at the end of each arm. These motors have a fixed pitch angle and can be controlled individually [40]. The differences between rotor speeds allow the six degrees of the freedom system to move in translational and rotational motion [41]. The vertical up and down movements can be achieved by equally increasing or decreasing the power from every single motor at the same time. Yaw movement can be achieved by generating opposite difference torques from each pair. The translational movements are generated by altering pitch and roll angles. The former can be achieved by varying the speeds of the first and third motors while the latter is generated by having a difference in speed between the second and fourth rotors [36]. Hovering is a special movement that keeps the quadrotor level at a specific point in the Z direction. However, describing the behavior of a quadrotor mathematically requires many robotic concepts. The quadrotor's system is explained in detail from its coordinate system to its linearization in the following model information [37].

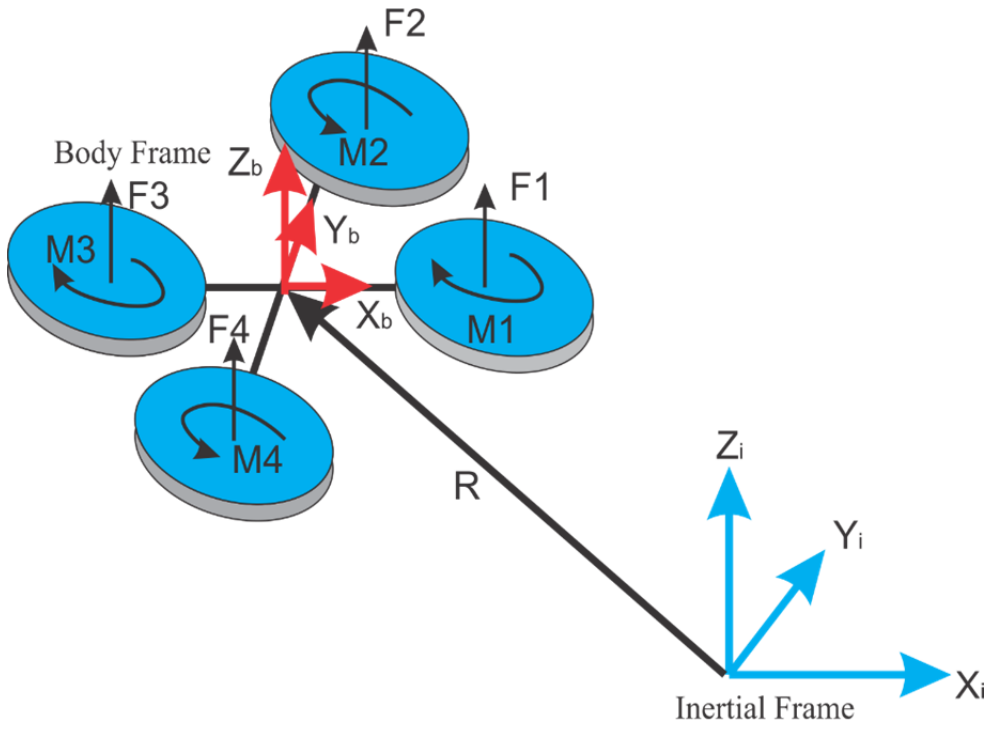

Figure 2.1: A quadrotor configuration.

2.2.1 Kinematics of a Quadrotor

The derivation of the dynamics of a quadrotor begins with establishing two reference frames: an Inertial or Earth frame and a body frame as shown in Figure 2.1. The Earth frame is a reference point used to describe the linear position of a quadrotor in three axes [X Y Z]. The linear position of the quadrotor is a vector and is denoted as $\mathbf{\Gamma^E}$ [m]. The Earth frame is also used as a local coordinator for Euler angles $\mathbf{\Theta^E}$ [rad] which are roll (ϕ), pitch (θ), yaw (ψ) angles [1]. The body frame is also a moving reference point attached to the center of gravity of a quadrotor to describe the linear $\mathbf{V^B}$ [m s^{-1}] and angular $\boldsymbol{\omega^B}$ [rad s^{-1}] velocities. The full mathematical description of the Earth and Body frame is shown below [1, 37]:

$$\xi = [\,\boldsymbol{\Gamma}^{\boldsymbol{E}}\ \boldsymbol{\Theta}^{\boldsymbol{E}}]^{T} \tag{2.1}$$

$$\nu = [\mathbf{V}^{\boldsymbol{B}}\ \boldsymbol{\omega}^{\boldsymbol{B}}]^{\,T} \tag{2.2}$$

From a kinematics perspective, the description of a generic six degrees of freedom quadrotor is shown below [37]:

$$\dot{\xi} = J_{\Theta}\,\nu \tag{2.3}$$

Where $\dot{\xi}$ $[\dot{\boldsymbol{\Gamma}}^{\boldsymbol{E}}\ \dot{\boldsymbol{\Theta}}^{\boldsymbol{E}}]$ and ν $[\mathbf{V}^{\mathrm{B}}\ \boldsymbol{\omega}^{\mathrm{B}}]$ are velocity vectors applied in an inertial frame and a body frame respectively. J_{Θ} is a matrix that contains the rotational $[R_{\Theta}]$ and translational$[T_{\Theta}]$ matrices through which the forces and torques of a quadrotor in a body frame can be expressed with respect to an inertial frame and vice versa [37]. It is important to mention that the rotation model used here is Z-X-Y Euler angles. It means that the sequence of the rotations will start with rotating the Z axes by the yaw (ψ). Then, it will end the rotation by rotating Y via pitch (θ) after rotating X axes by the roll angle (ϕ) [2, 37]. These rotational and translational matrices are defined below (where C, S are cosine and sine respectively):

$$R_{\Theta} = \begin{bmatrix} \mathrm{C}\,\psi\,\mathrm{C}\,\theta - \mathrm{S}\,\phi\,\mathrm{S}\,\psi\,\mathrm{S}\,\theta & -\mathrm{C}\,\phi\,\mathrm{S}\,\psi & \mathrm{C}\,\psi\,\mathrm{S}\,\theta + \mathrm{C}\,\theta\,\mathrm{S}\,\phi\,\mathrm{S}\,\psi \\ \mathrm{C}\,\theta\,\mathrm{S}\,\psi + \mathrm{C}\,\psi\,\mathrm{S}\,\phi\,\mathrm{S}\,\theta & \mathrm{C}\,\phi\,\mathrm{S}\,\psi & \mathrm{S}\,\psi\,\mathrm{S}\,\theta - \mathrm{C}\,\psi\,\mathrm{C}\,\theta\,\mathrm{S}\,\phi \\ -\mathrm{C}\,\phi\,\mathrm{S}\,\theta & S\,\phi & \mathrm{C}\,\phi\,\mathrm{C}\,\theta \end{bmatrix} \tag{2.4}$$

And

$$T_{\Theta} = \begin{bmatrix} 1 & S\phi\mathrm{t}\theta & C\,\phi\mathrm{t}\theta \\ 0 & \mathrm{C}\,\phi & -S\,\phi \\ 0 & S\,\phi/\mathrm{C}\theta & C\phi/\mathrm{C}\theta \end{bmatrix} \tag{2.5}$$

2.2.2 Inputs of a Quadrotor

As mentioned above, the propeller's speed is the input of the quadrotor. The rotor speed not only generates the vertical forces, but also produces moments that are associated with thrust factors (b) and drag force (d), respectively as shown below [42-44]:

$$\boldsymbol{F_i} = b\,\Omega_i^{\,2} \tag{2.6}$$

$$\boldsymbol{M_i} = d\,\Omega_i^{\,2} \tag{2.7}$$

To have a full description of the quadrotor's inputs, there are a few points that should be clarified. First, the propellers that are attached to the motors should be rigid. Second, the direction of the pair of rotors located on the same arm should rotate in the opposite direction to the other pair, meaning that rotor 1 and 3, as shown in Figure 2.1, rotate clockwise, which is the opposite direction of rotor 2 and 4. From a mathematical perspective, the overall description of the rotor speed that allows the quadrotor to generate thrust forces is presented in Equation (2.8) [37].

$$\boldsymbol{\Omega} = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \tag{2.8}$$

At the current time, the quadrotor's control input, generated by the motors, can be expressed in terms of a total body force **[U]** and three torques $[\boldsymbol{\tau_\phi \tau_\theta \tau_\psi}]$ as presented from Equation (2.9) to Equation (2.12) [12, 44]. In addition, these equations are mathematical descriptions of the quadrotor's movements that are described above. To illustrate this point, we have said that Yaw movement can be achieved by generating opposite difference torques from each pair. It is clear from Equation (2.12) that the direction of rotor 1 and 3 is different from the other rotors.

$$\boldsymbol{U} = b\,(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{2.9}$$

$$\boldsymbol{\tau}_{\phi} = b\,(-\Omega_2^2 + \Omega_4^2) \tag{2.10}$$

$$\boldsymbol{\tau}_{\theta} = b\,(\Omega_1^2 - \Omega_3^2) \tag{2.11}$$

$$\boldsymbol{\tau}_{\psi} = d\,(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{2.12}$$

Finally, it is necessary to mention that the performance of the quadrotor is relying on these inputs which are rotor speeds. The functions of the propeller, motor controller, and motor dynamic combine to produce the rotor speed [2]. Therefore, a high performance is achieved when the actual rotor speed is the same as the commanded rotor speed. In this study, we neglect the motor dynamic as it has only a minor effect on the performance of the quadrotor.

2.2.3 Equations of Motions

When determining the equations of motion, it is important to consider some ground points [45]. Firstly, the quadrotor's structure, and in particular the cross arms, should be symmetrical. Secondly, the entire structure, including the propellers, should be rigid. Thirdly, the quadrotor's center of gravity and fixed frame origin should be at the same point. Finally, the thrust and drag forces need to be proportional to the propellers' speeds [45]. The dynamics of the quadrotor takes into account the mass m [kg] of the body frame and its inertia matrix $\boldsymbol{I}$ [N m s^2]. The translational and rotational dynamic equations of a quadrotor with respect to the body frame are described as follows [14, 37]:

$$\begin{bmatrix} \boldsymbol{F}^B \\ \boldsymbol{\tau}^B \end{bmatrix} = \begin{bmatrix} m\,\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}}^B \\ \dot{\boldsymbol{\omega}}^B \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}^B \times (m\,\mathbf{V}^B) \\ \boldsymbol{\omega}^B \times (m\,\boldsymbol{\omega}^B) \end{bmatrix} \tag{2.13}$$

After taking all of the above into account, the Newton- Euler formalization can be applied to derive the equations of the motions of the quadrotor [14].

$$\ddot{x} = (\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi)\frac{U}{m} \tag{2.14}$$

$$\ddot{y} = (\sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi)\frac{U}{m} \tag{2.15}$$

$$\ddot{z} = (-g + \cos\theta \cos\phi)\frac{U}{m} \tag{2.16}$$

$$\ddot{\phi} = \left(\frac{I_y - I_z}{I_x}\right)\dot{\theta}\,\dot{\psi} + \left(\frac{L}{I_x}\right)\boldsymbol{\tau}_{\phi} \tag{2.17}$$

$$\ddot{\theta} = \left(\frac{I_z - I_x}{I_y}\right)\dot{\phi}\,\dot{\psi} + \left(\frac{L}{I_y}\right)\boldsymbol{\tau}_{\theta} \tag{2.18}$$

$$\ddot{\psi} = \left(\frac{I_x - I_y}{I_z}\right)\dot{\phi}\,\dot{\theta} + \left(\frac{1}{I_z}\right)\boldsymbol{\tau}_{\psi} \tag{2.19}$$

Because gyroscope effects had such a minor impact on the results, they were considered unimportant and left out of the dynamics equations [3]. The other parameters are acceleration due to the gravity (g), and the mass moment of inertia ***I*** of each axis [I_x I_y I_z]. Also, the length L [m] is the horizontal distance from the center of propeller center to the center of gravity of the vehicle.

2.2.4 Linearization

The quadrotor's dynamics, from a control perspective, is nonlinear. Any equilibrium configuration equations of motion must therefore be linearized if using a nonlinear system with a linear controller. The hover mechanism, set at any arbitrary position, is the equilibrium configuration of this study. The related thrust force required for hovering at this configuration is exactly mg and the moment is zero. The hover configuration variables' values are the roll and pitch angles zeros ($\theta = \phi = 0$) except yaw angle ($\psi = \psi_0$), and the linear position vector **Γ** is equal to $\boldsymbol{\Gamma_0}$**.** However, the linear velocity and angular velocities are zeros.

In order to simplify the dynamic equation, we need to linearize it by taking all non-linear functions of the states which exist at the quadrotor's hovering status and replacing them with

their first order Taylor approximation, where the roll and pitch are insignificant ($\cos\theta \approx 1, \cos\phi \approx 1, \sin\theta \approx \theta, \sin\phi \approx \phi$). In this study, the linear equations presented below will be used in developing the control algorithms especially the PID controller as will be discussed in Chapter 4 [3].

$$\ddot{x} = -g\theta \tag{2.20}$$

$$\ddot{y} = g\phi \tag{2.21}$$

$$\ddot{z} = -g + \frac{U}{m} \tag{2.22}$$

$$\ddot{\phi} = \left(\frac{L}{I_x}\right)\boldsymbol{\tau}_{\phi} \tag{2.23}$$

$$\ddot{\theta} = \left(\frac{L}{I_y}\right)\boldsymbol{\tau}_{\theta} \tag{2.24}$$

$$\ddot{\psi} = \left(\frac{1}{I_z}\right)\boldsymbol{\tau}_{\psi} \tag{2.25}$$

2.3 System Architecture

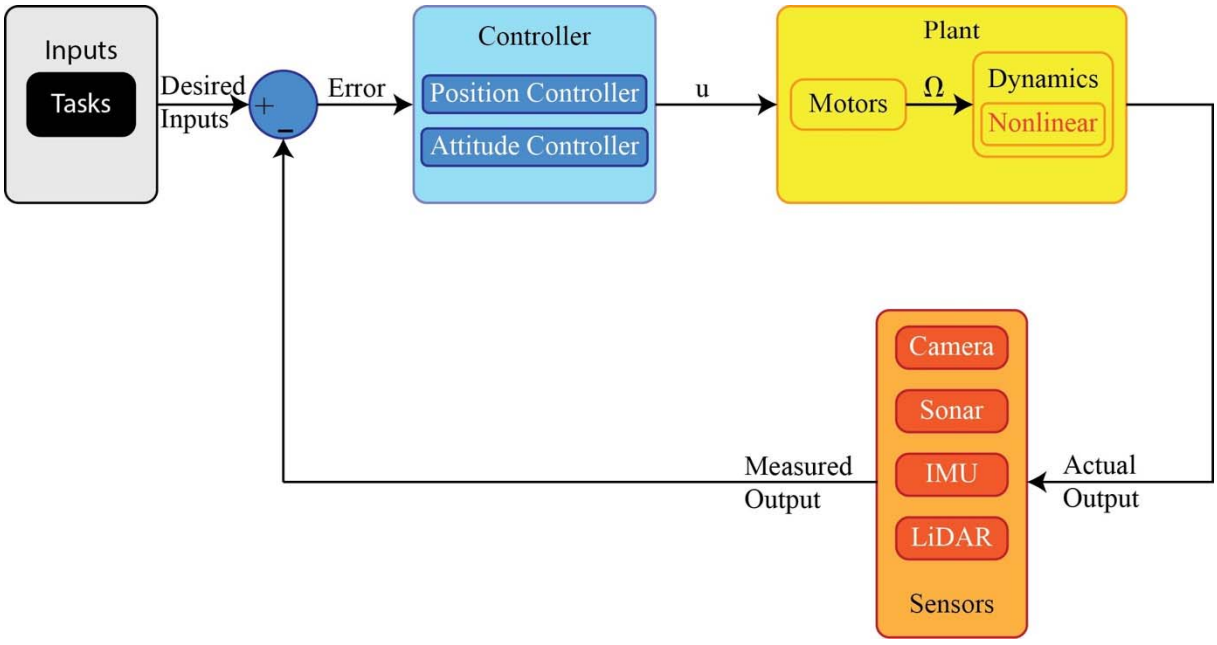

Figure 2.2: The general overview of quadrotor structure.

The system architecture used in this work is presented in Figure 2.2. Its first block represents one goal of this work: following the terrain with a certain altitude in two dimensions. More precisely, the inputs of this study are the changes of the Z-axes, and the velocity of the quadrotor in X axes. Since the velocity of the quadrotor in X axes is affected by pitch angle as shown in Equation (2.20), the third input of this study is the pitch angle. The controller block is the main component of this work because all the computational algorithms occur in this block. It can be divided into two parts: high level controller or position controller, and low level controller or attitude controller. The developments of this block are presented in Chapter 3 and 4.

Measuring the output of the quadrotor will complete the circle of this work. Sensors, which are the fourth block, are playing undreamed of roles in evolving autonomous vehicles, especially in applications that would not normally be possible without human involvement such as detecting mines. Choosing the right sensors for this work requires making a comparison between the sensors. Here is a brief description of each sensor shown in Figure 2.2. A camera is not explicitly a measurement sensor. Cameras allow the quadrotor to photograph people and objects on the ground. These photos are then used to create trajectory planning using computer vision strategies. The drawback of this type of sensor is a long computation and not a direct measurement like other sensors.

An Inertial Measurement Unit (IMU) is an electronic device made up of accelerometers, magnetometers, and gyroscopes, which measures information about the quadrotor's altitude and direction [37]. Accelerometers basically measure the acceleration of the quadrotor through which the orientation of the system can be determined. It provides the controller with the position of the quadrotor in three axes (X,Y,Z). Gyroscopes also measure the rate of

angular velocity through which the controller gets information about the roll-pitch-yaw angles. Magnetometers measure the magnetic field around the quadrotor and provide the controller with the head of the quadrotror. Based on this information, we can see that IMUs are a necessary component of any flight machine. A sonar is a sound navigation and ranging technique device which uses sound propagation to determine how far a flying vehicle is from the ground. An ultrasonic rangefinder, a type of sonar sensor, is made up of a receiver and a transmitter. These two features work in tandem, the transmitter to send and the receiver to collect waves returned from obstacles in the environment. Ultrasonic rangefinders are not complicated or expensive and their beams are reflected back from almost any kind of obstacle, unlike those of a laser [37]. It also remains unaffected by background light, dust, moisture or radiation. There are, however, limitations to ultrasonic sensors which make them unsuitable for certain tasks. One of these limitations is the occurrence of specular reflection when the angle of incident sound is below a specific angle. This can be a problem because the reflected waves of the ultrasonic beams will not be reflected back if they hit the obstacle below the critical angle. Therefore, the only beams reflected back are those from surfaces almost perpendicular to the ultrasonic beams [9].

The laser rangefinder is similar to the ultrasonic sensor; the only difference between the two sensors is that the former measures the travel time of a laser pulse, whereas the latter measures the time of flight of an ultrasonic wave. Laser rangefinders are more relevant for use with outdoor robots because they can perform at long distances [9], at an accuracy of up to 2% of the range measured and at a high sample frequency of 20-50 KHz. More importantly, they are more flexible than ultrasonic rangefinders in terms of positioning. Laser rangefinders do not require vertical and horizontal positions on the robot to offer the

optimum detection, even if the surface of the obstacle is below the critical angle for ultrasonic rangefinders. However, the lasers are affected by dust or other particles, and they do not manage to measure the range of a point of transparent, sharp or shiny surfaces as well as ultrasonic rangefinders do [9].

2.4 Problem Statements

Today, quadrotors are frequently used in many civilian applications, and especially in industrial projects. One such applications is where a camera or specific sensor attached to the bottom of the quadrotor visually monitors a plant or measures the level of fugitive gases present in the air surrounding a gas pipeline. Nowadays, there are many remote controlled quadrotors performing these tasks perfectly; however, recently, some industrial projects have intended to utilize these applications with an autonomous quadrotor. This slightly different goal creates difficulties in terms of implementation as a result of two main issues. First, aerial monitoring sensors such as gas sensors attached to the bottom of the quadrotor are required to be at a certain distance, for example from the source of the gas, to operate properly. Second, one of the major issues with quadrotors and especially with this application is that the quadrotor has to be very intuitively navigated, especially when dealing with low altitudes, within the environmental boundaries of often varied and changing terrains [4]. Third, the quadrotor structure in the market today is not sufficient for this type of application. Looking closely at the challenges and finding a solution for this way of operating a quadrotor is the main contribution of this thesis.

To solve this problem, we started with an analysis of quadrotors in the market today. We found that many of the quadrotors contain basic electronic units such as sonar sensors and IMU. These sensors perform tasks adequately on manual flights, but are insufficient for

maintaining a fixed distance from the terrain during automatic flights as a result of lack of data about the elevation and undulations of the environment. Figure 2.3 shows how the quadrotor flies when the basic instrumental devices are attached in the traditional way. It is clear that the quadrotor relies on the ultrasonic rangefinders to identify the altitude of the quadrotor. This technique helps the quadrotor to follow the ideal path as long as the changes in the undulations of the terrain are not too dramatic; however, this technique is not sufficient if the quadrotor operates as shown in Figure 2.3. The flight path will actually look like that of the blue quadrotor. In this case, we can clearly identify the insufficiency of traditional mode of operation in following the terrain while autonomously maintaining a certain distance from it as the main problem. To solve this problem, the following steps should be taken. First, the quadrotor requires an additional sensor to feed the microcontroller about the undulations of the environment or terrain. Second, the range data should be processed in a way that helps the quadrotor to build its trajectory ahead. Finally, a controller needs to be designed that can help the quadrotor to follow the terrain precisely and minimize energy wasted as a result of sudden trajectory changes or recovery. The second and third issues will be discussed in detail in Chapter 3 and 4, respectively.

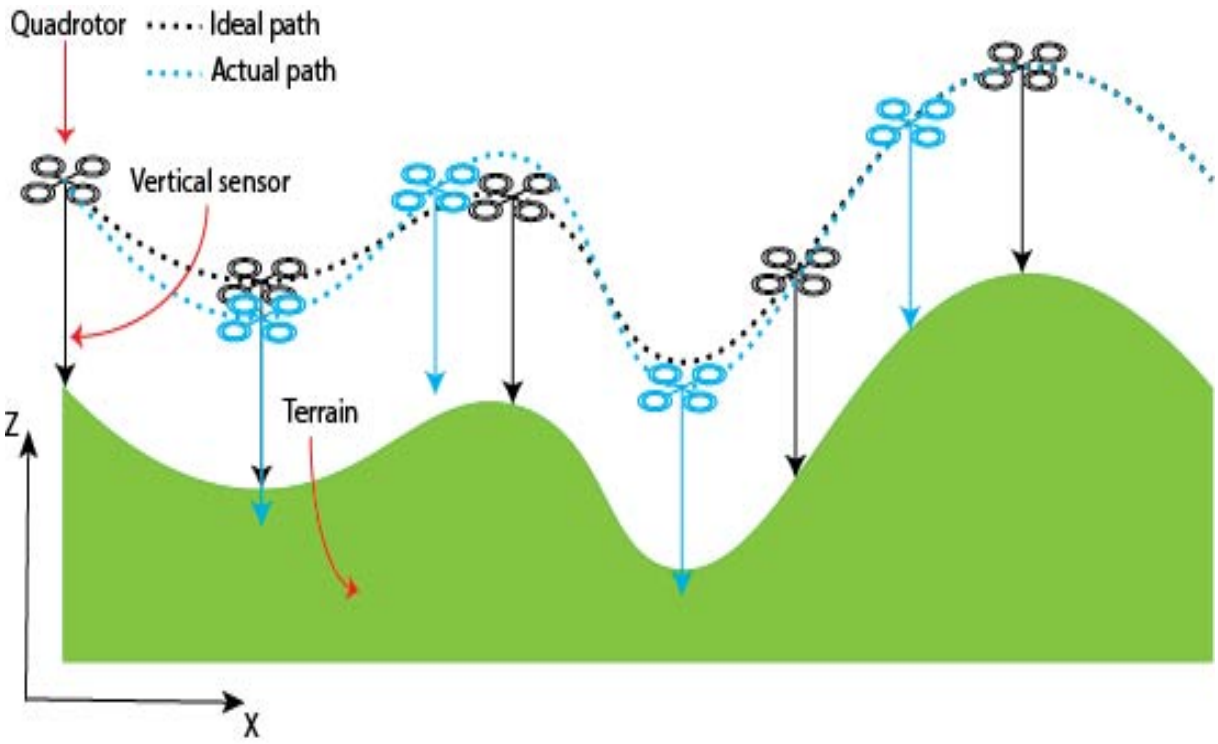

Figure 2.3: Following the terrain by a quadrotor.

The proposed and new navigational method in this thesis involves attaching a single-beam LiDAR to the base of a quadrotor and then setting the sensor at different sets of angles to send information back to the quadrotor about undulations in the terrain beneath and ahead. Utilizing this laser feedback, the motion planning algorithms create a smooth trajectory which enables the quadrotor controller to track and follow the terrain precisely. To fix the distance between the quadrotor and terrain autonomously, the new navigation mode has three steps as illustrated in Figure 2.4. First, the single-beam LiDAR measures the terrain undulations beneath and ahead of the quadrotor at the desired altitude. Then, motion planning algorithms, such as Gaussian filter and cubic spline, create a smooth trajectory plan following which the quadrotor can avoid having to make a sudden recovery. Finally, this trajectory planning is provided to an adequate controller (namely a model predictive control

(MPC)), which not only helps the quadrotor to follow the terrain exactly, but also to minimize energy wasted on sudden recoveries.

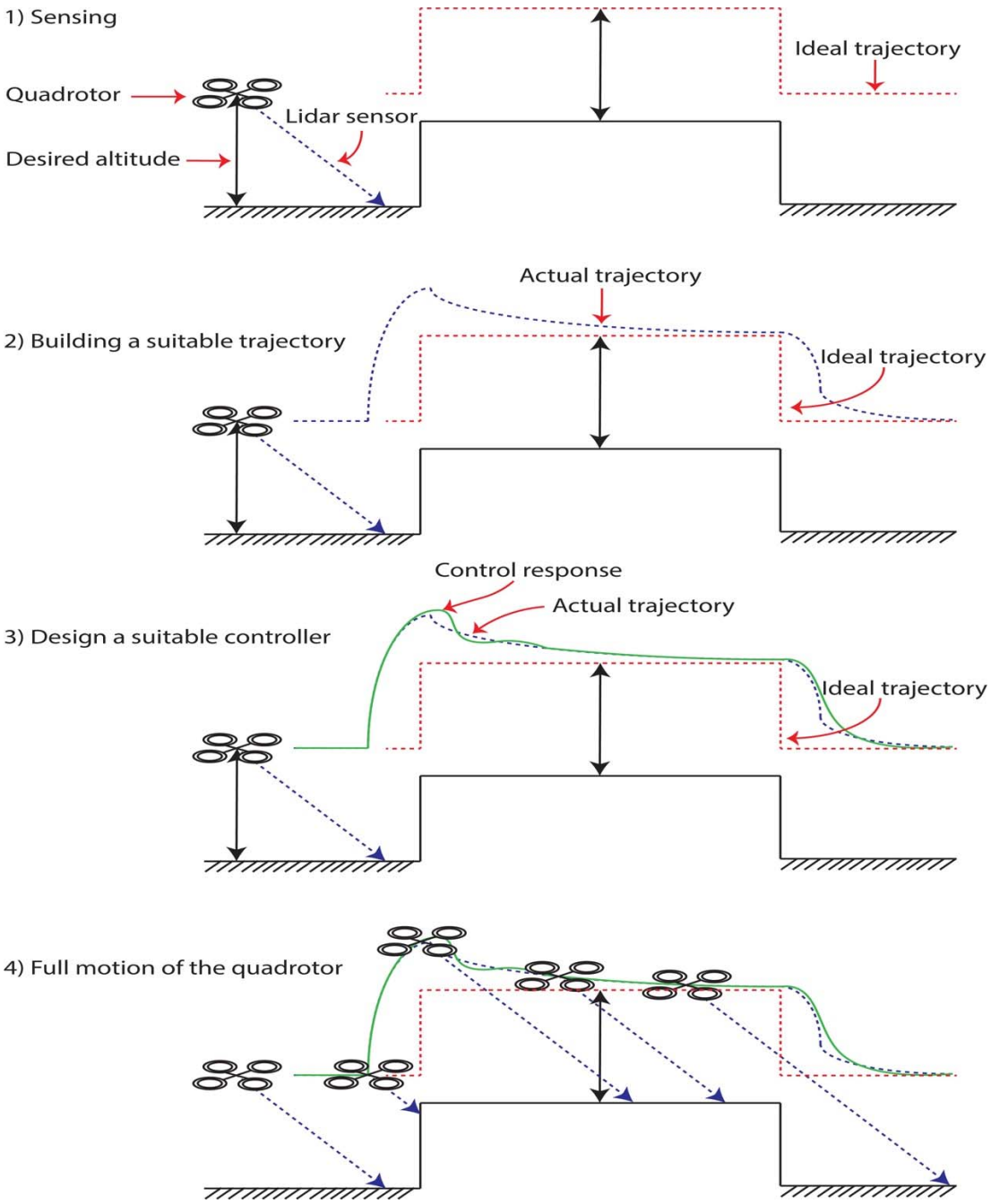

Figure 2.4: Full steps of maintaining the altitude by a quadrotor with the new navigational approach.

Chapter 3: Motion Planning

3.1 Overview

In the previous chapter, we established the problem quadrotors have in following unknown terrain, and developed a solution: a method of gathering information about the undulations of the terrain. This chapter addresses a way of utilizing the laser feedback about the terrain's undulations, both beneath and ahead of the quadrotor. More precisely, it provides the methodology, result, and discussion regarding motion planning algorithms with which we can analyze the performance of a quadrotor using the proposed solution outlined in the previous chapter. The performance will be analyzed based on the quadrotor's ability to avoid sudden recoveries, minimize wasted energy, and offer a long flight time. Generally speaking, we introduced three different trajectory planning algorithms based on a linear spline, cubic spline, and Gaussian filter. We then compared them with the existing method. Ultimately, analyzing whether or not helping the quadrotor to gather information about the undulations ahead of it is effective depends on the goal of using the quadrotor. This means that the choice of a suitable trajectory planning algorithm and the angle of the pointing laser also depends on the purpose of using the quadrotor. The development of this approach starts by illustrating the overall idea in Section 3.2. Next, the linear and cubic spline algorithms are described in Section 3.3. Gaussian Filter is presented as an alternative method of trajectory planning in Section 3.4. Finally, the simulation results are shown in the simulation Section 3.6, after a brief introduction about low pass filter in Section 3.5.

3.2 Problem Statement

Positioning the LiDAR sensor at an angle feeds the quadrotor information about imminent changes in the terrain, or more precisely, the Z axes. The size of the window and hence the amount of data gathered about the upcoming terrain depends on the set angle of the LiDAR

sensor. We can assume that this data becomes the waypoints of a quadrotor within the configuration space. The quadrotor is required to travel collision-free based on these waypoints starting from an initial point q_0 ending at a final point q_f. The next challenge is to design a trajectory that connects these waypoints while satisfying specific constraints such as velocity and acceleration constraints at each point. In practice, all constraints are defined mathematically as shown below [46]:

$$q(t_0) = q_0 \;\&\; q(t_f) = q_f \tag{3.1}$$

$$\dot{q}(t_0) = v_0 \;\&\; \dot{q}(t_f) = v_f \tag{3.2}$$

$$q_{min} \leq\; q(t) \;\leq q_{max} \tag{3.3}$$

$$\dot{\theta}_{min} \leq\; \dot{\theta}(t) \;\leq \dot{\theta}_{max} \tag{3.4}$$

$$\boldsymbol{\tau}_{min} \leq\; \boldsymbol{\tau}(t) \;\leq \boldsymbol{\tau}_{max} \tag{3.5}$$

Equation (3.1) and (3.2) represent the quadrotor's goal. The former illustrates the initial and final configuration of a quadrotor with initial and final required flight time while the latter represents the velocity at each configuration. The limitation on the configuration space in which the quadrotor will move is illustrated in Equation (3.3). Equation (3.4) illustrates the constraint on the rotational speed of the electric motors powering the quadrotor. It also includes the constraint on the direction of the rotation of these motors. Equation (3.5) illustrates the torque constraint that arises from the limited power of these motors. Both Equations (3.4) and (3.5) are practical constraints that depend on the motor specifications. In this thesis, we intended to design a trajectory for a quadrotor that flies at an altitude of 40m above the terrain keeping a safe distance from the obstacles at all times and with a constant velocity of 10 m/s in X direction. These specifications are used as examples throughout the research to validate the concepts of the application.

3.3 Linear and Cubic Splines

The main advantage of utilizing a spline in generating trajectory for a robot is the spline's ability to take constraints into account while building the trajectory. The basic idea behind linear and cubic splines is that they generate a series of feasible waypoints by utilizing high level planning algorithms. Trajectory planning is then built based on passing through these points under certain smoothness criteria [47]. The smoothness criteria are the constraints at the start and end velocities. Linear Segments with Parabolic Blends and cubic splines are represented in Equations (3.6) and (3.7) respectively. Moreover, these equations are a function of time, which helps users of splines to determine the time required for the robot to go from its starting point to its destination.

$$q(t) = a_o + a_1 t + a_2 t^2 \tag{3.6}$$

$$q(t) = a_o + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.7}$$

Where the coefficients are smooth criteria. In other words, these coefficients represent the unique solution of all constraints at each feasible point. Thus, determining these coefficients is a crucial step in helping the quadrotor to generate a smooth trajectory. In our case, it is required that the quadrotor maintain an altitude of 40m, which means that the velocity and acceleration of the quadrotor in the Z axes are zeros. As a result, the general required cubic polynomial function that satisfies zero velocities in both points is represented in Equation (3.7) [33]. The only element not defined is time as shown in Equation (3.8). Determining the time depends on the purpose of using the quadrotor. For instance, if we want the quadrotor to do the task in 1 second, the Equation (3.9) represents the required polynomial equation, or more precisely the unique solution of the cubic spline that generates trajectory while

considering all constraints. The reader interested in deriving linear spline is referred to these references [33, 47].

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f^2 & 3t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{3.8}$$

$$q(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3 \tag{3.9}$$

Figure 3.1 shows the full steps of generating cubic spline during the quadrotor flight. This example shows how a quadrotor's trajectory is generated while moving over a flat terrain before encountering a 5m vertical object. The changing landscape of the terrain has already been fed to the quadrotor via the LiDAR sensor which detected the object before it was reached by the quadrotor. Figure 3.1 illustrates that even though the quadrotor has the information about the sharp change in elevation ahead, the spline did not generate a trajectory until the quadrotor reached the action point. The action point is a point that is calculated to allow the quadrotor to move up and down as smoothly and efficiently as possible. Choosing a suitable action point is one of main factors in helping the quadrotor to have a smooth trajectory. In Figure 3.1, this action point appears 10 m before the sharp object. As well as offering a smooth trajectory, the figure indicates that the proposed solution with cubic spline as a trajectory technique also assists the quadrotor to move as efficiently as possible: it elevates gradually so as not to use a lot of energy in avoiding the sharp object.

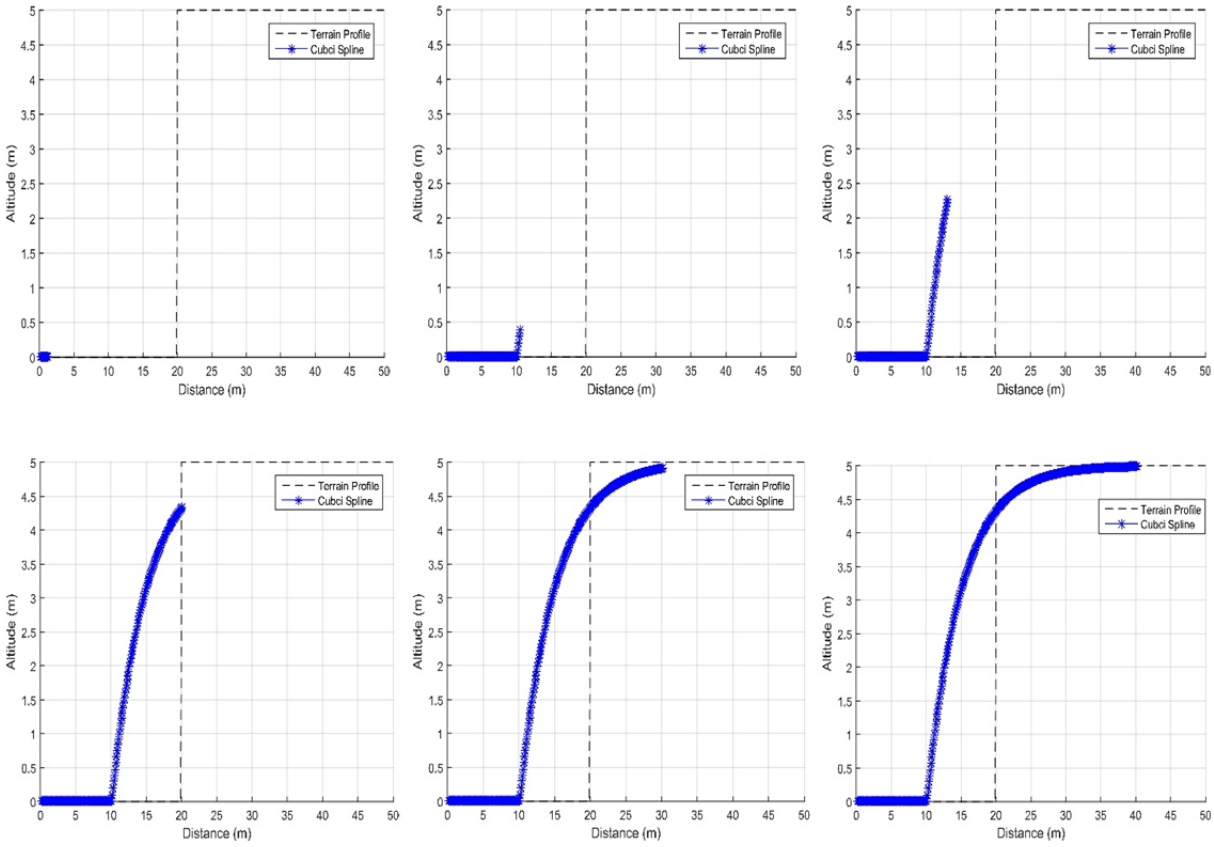

Figure 3.1: The cubic spline trajectory generating while the quadrotor flies.

3.4 Gaussian Filter

In this section, it is shown that there is a method of generating trajectory without considering velocity and acceleration constraints at each feasible point. Gaussian filter is a popular technique used in robotics, and especially in terrain modeling, due to the fact that it is not a parametric tool [48, 49]. The concept of building a smooth quadrotor trajectory by using a Gaussian filter revolves around distributing the terrain data on a Gaussian filter of the mean (μ) zero, as shown in Figure 3.2. The effectiveness of a trajectory will rely on the width of the Gaussian filtering window [48]. The way we used Gaussian filter to building the trajectory was by distributing 11 data points of the undulations of the terrain at each step, and the varying the width of the Gaussian filtering window from 25 to 5 as the quadrotor ascends

and descends, respectively. After that, we built the trajectory based on the filtered data of the left side of the Gaussian distribution, as illustrated in Figure 3.2. The reason for choosing the left side is that we give a higher weight to the points of the terrain closer to the quadrotor.

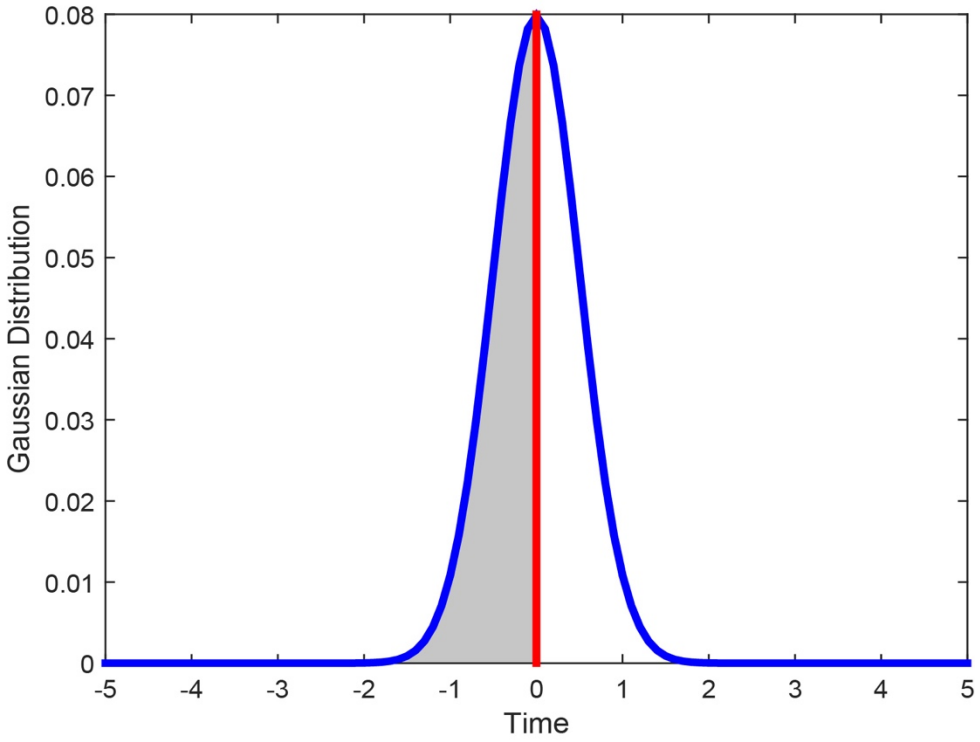

Figure 3.2: Trajectory planning via Gaussian filter.

3.5 Filtering

It is assumed when we build the trajectory that the information we get from the sensors is accurate; however, this is not the case in real life applications. Navigation in unknown terrain with a system like a quadrotor, integrated with several measuring instruments, is expected to be difficult due to the existence of uncertainties. Measurements data are not only fully associated with uncertainties, but are also corrupted by measurement noise. Moreover, the system model itself is a source of uncertainties as a result of the linearization and modeling errors. Hence, it is necessary to introduce a filtering technique.

Filtering is an estimation method whose function is to reduce the impact error on the measurements. The basic idea behind filtering is that it uses existing information about the system, as well as redundant measurements, to achieve a clearer state estimate than would be available directly from the measurements themselves. This technique of filtering assists the quadrotor to have a better estimation of the states, if our concern is about the estimation of the states. Our concern for this application pertains to environmental uncertainties. The difficulty with environmental uncertainties is that there is no *prior* knowledge of the environment to draw from, nor is there another sensor that can be used alongside the LiDAR sensor, to estimate, in particular, small changes in the environment.

Due to the presence of small terrain alterations, the model uses a low pass filter technique to reduce uncertainty in the measurements of the LiDAR sensor. Since the LiDAR sensor has a high resolution, the range of uncertainty has been assumed to span from + 0.5 down to - 0.5 meters. That range has been added to the laser measurements as random noise in the simulations.

3.6 Results

The proposed solution presented in this chapter has been validated numerically in Matlab, and the parameters used are presented in Table 3.1[50]. As mentioned above, the effectiveness of utilizing the reading of the undulations of the terrain underneath and ahead of a quadrotor depends on the method that is used in the trajectory planning algorithms and the control algorithms. To highlight the utlitiy of the proposed solution with regards to the trajectory planning techniques only, we used in this section only one type of control technique: the PID controller. Of course, the result will be better if we use a more advanced controller that is the scope of next chapter. It is also important to mention that when we

compare between trajectory planning methods, we are looking at how these methods perform on the baseline. The baseline is an imaginary and identical profile created for this study and set at a defined distance from the terrain. The terrain profile used in this section is a step profile, thus it has a sharp vertical incline at the beginning and a sharp vertical decline at the end, making it difficult for the UAV to follow. For easy comprehension, this section splits into two sections: trajectory planning and motion planning using a Gaussian Filter.

Table 3.1 : The parameters of a quadrotor used in the Matlab.

#	**Parameters of a Quadrotor**		
	Parameter	***Value***	***Unit***
1	Mass	1.2	kg
2	Mass Moment of Inertia of X and Y Axes	0.002	Kgm^2
3	Mass Moment of Inertia of Z Axis	0.005	Kgm^2

3.6.1 Case Study: Generating Trajectory Planning

Generating a trajectory suitable for a step terrain profile relies on two main factors. The first factor is the set angle of the laser beam: if the system has a wide angle set, it will be able to gather more feasible data about the terrain. The second factor is the constraints on the inputs of the quadrotor, which come from the motor's limitations. Besides these general influences, every method used in this section has its own pros and cons. To make the results easier to understand, the results are divided into two main categories: i) finding the best trajectory planning algorithm and ii) finding the best motion planning method. The first study compares the proposed trajectory planning algorithms in terms of the smoothness of the trajectory, and how closely it replicates the terrain. The second study compares the proposed trajectory planning algorithms in terms of the performance of a quadrotor by utilizing the trajectory planning produced. Figure 3.3 shows the trajectories of a quadrotor, using a Gaussian filter and cubic and linear splines with an angle of 45 degrees. It also shows the trajectory of a

quadrotor based on a vertical sensor. The results show that the Gaussian filter outperforms the cubic and linear splines in terms of replicating the terrain. However, the results also show that the cubic and linear splines are better than the Gaussian filter in terms of predicting missing information such as sharp-ended objects. There are three noteworthy points on this figure. First, it shows how the extra information about the terrain helps the quadrotor to avoid collisions in comparison with the quadrotor's traditional response on reaching an obstacle. Second, Figure 3.3 indicates how the controller's constraints affect the generation of a trajectory. It is clear that the trajectory without constraints, Figure 3.3 (b), is better even though it consumes a lot of energy as we will see in Chapter 4. Finally, it is important to mention that it is sometimes difficult to say which method is the best because it depends on the terrain profile itself. For the step profile, the most difficult compared to the others, the Gaussian method is the best. However, sometimes the task requires more flexibility when building the trajectory, and in that case, splines are far better methods to use than the Gaussian method.

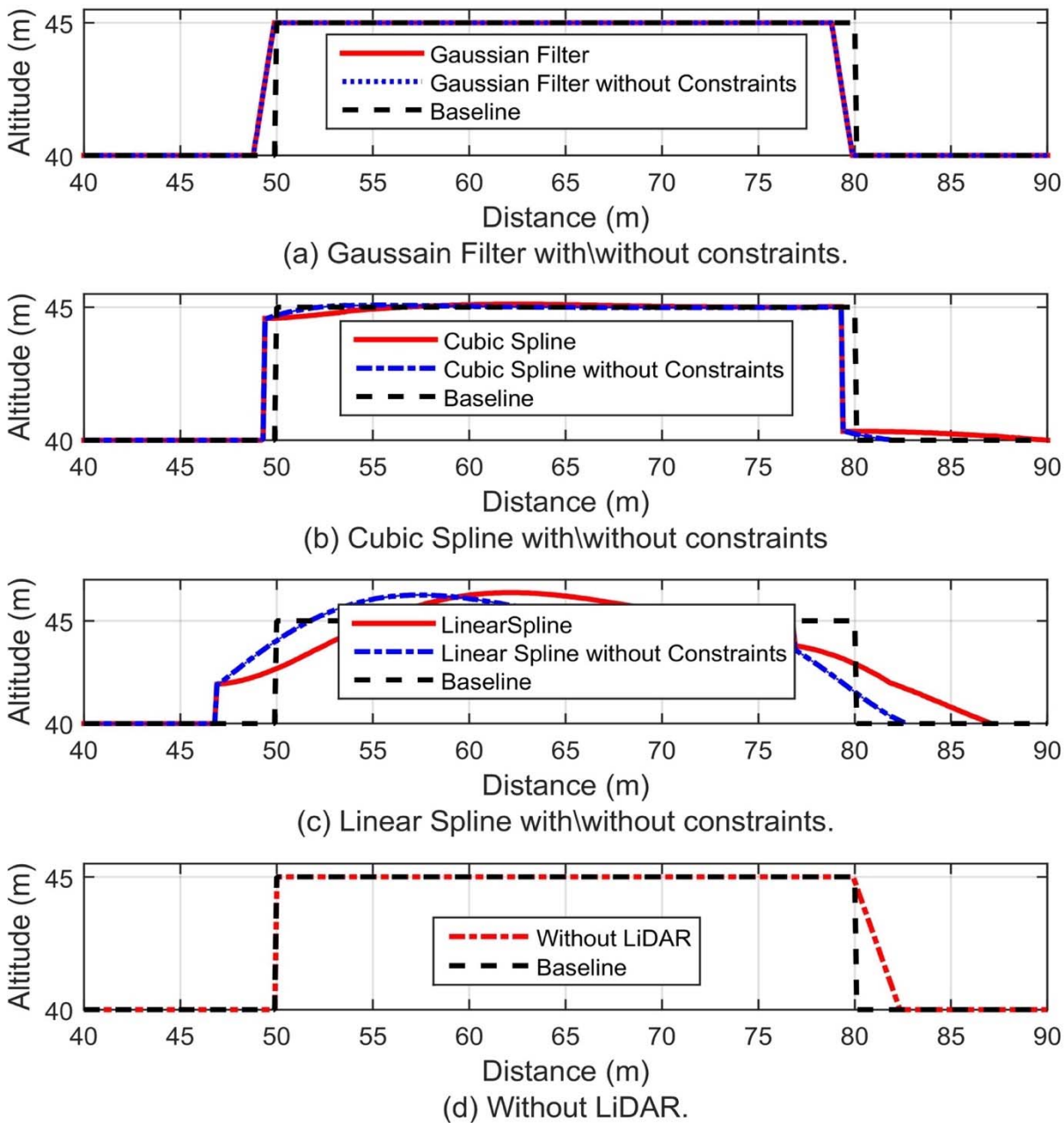

Figure 3.3: Trajectory of step profile in all techniques.

Table 3.2 and Table 3.3 present the error of each trajectory planning method with and without constraints acting on the controller of the quadrotor. These tables associate the error with the angle used by the laser beam. The first table presents the ideal case, for which there are no constraints on the controller. In this case, the Root Mean Square (RMS) of following the terrain without a proposed solution is 15.1 for a flight of 110 meters, while the average

RMS for the Gaussian method is around 11.7. However, splines are the worst case. The reason for this is that they consider the velocity and acceleration of the initial point and desired destination. The latter table presents as the first one when considering the constraints on the inputs. Despite their flaws, all the presented techniques improve on the traditional method of sensing vertical altitude. The extra information provided by the LiDAR sensor and either Gaussian or spline techniques leads the quadrotor to plan the trajectory based on its ability while avoiding any sudden recoveries. The quadrotor therefore becomes better able to follow the terrain precisely. General observations of both tables suggest that the relationship between decreasing errors in replicating the terrain is proportional to the increasing angle set of the laser beam. Having a large angle set on the sensor, therefore, decreases errors when replicating the terrain.

Table 3.2: Root mean square deviation of the different trajectory planning techniques without different angle constraints.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	29.1286	17.7300	11.7220	15.1301
45	28.8875	17.5955	11.7181	
35	29.1414	17.7300	11.7265	
25	29.1756	17.7267	11.7172	
15	29.0862	17.6691	11.7215	
10	29.1092	17.6835	11.7016	

Table 3.3: Root mean square deviation of the different trajectory planning techniques with different angle constraints.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	24.2864	19.3898	22.7622	28.5927
45	24.1113	20.5056	22.7751	
35	24.3892	19.5008	22.7477	
25	25.7166	19.3002	22.7636	
15	25.5396	19.5407	22.7636	
10	24.3385	19.5725	22.8492	

Before discussing the motion of a quadrotor using various trajectory techniques, we need to mention important criteria to be considered when choosing the trajectory algorithms. The way the trajectory algorithms build a path, in terms of ascending sharply in cases like the step profile, and where profile information such as sharp edges are missing, can be seen in Figure 3.4 and Figure 3.5, where the trajectory of all ascension and descension quadrotor methods is shown. It is obvious that the traditional method of beginning the ascent as the quadrotor reaches the sharp object is likely to end in a collision if the sharp object's height is above the altitude of the quadrotor. In contrast, all the techniques which can offer extra information about the terrain undulations allow the quadrotor to react before reaching the sharp object, as we can see from Figure 3.4. The same is true of situations where the quadrotor needs to descend sharply to maintain its distance from the terrain, as shown in Figure 3.5. Without extra information, the trajectory algorithm instructs the quadrotor to descend before it reaches the cliff, which is also likely to end in collision. In the case of missing information about sharp edge, splines allow for a better trajectory than the Gaussian method. As we can see, linear and cubic splines fill in the missing information and generate a smooth trajectory, whereas the Gaussian method is likely to allow the quadrotor to make contact with the sharp edge. To fix this problem with the Gaussian method, it should have a small sigma. The full discussion of this issue will be in the next case study.

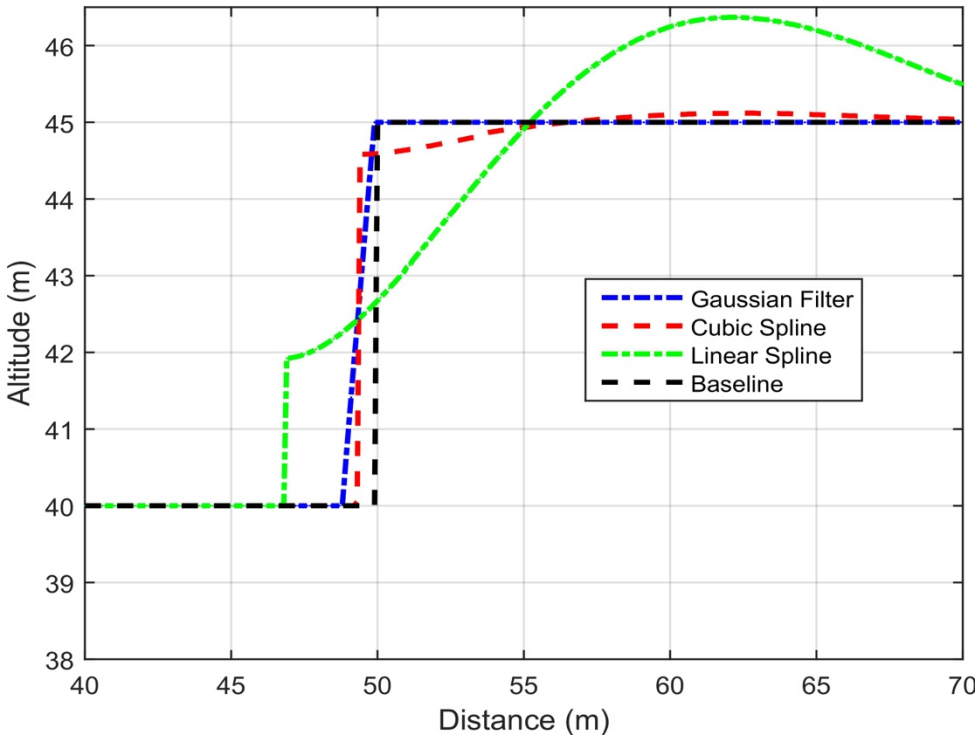

Figure 3.4: Trajectory planning of different techniques with constrains of the ascending case.

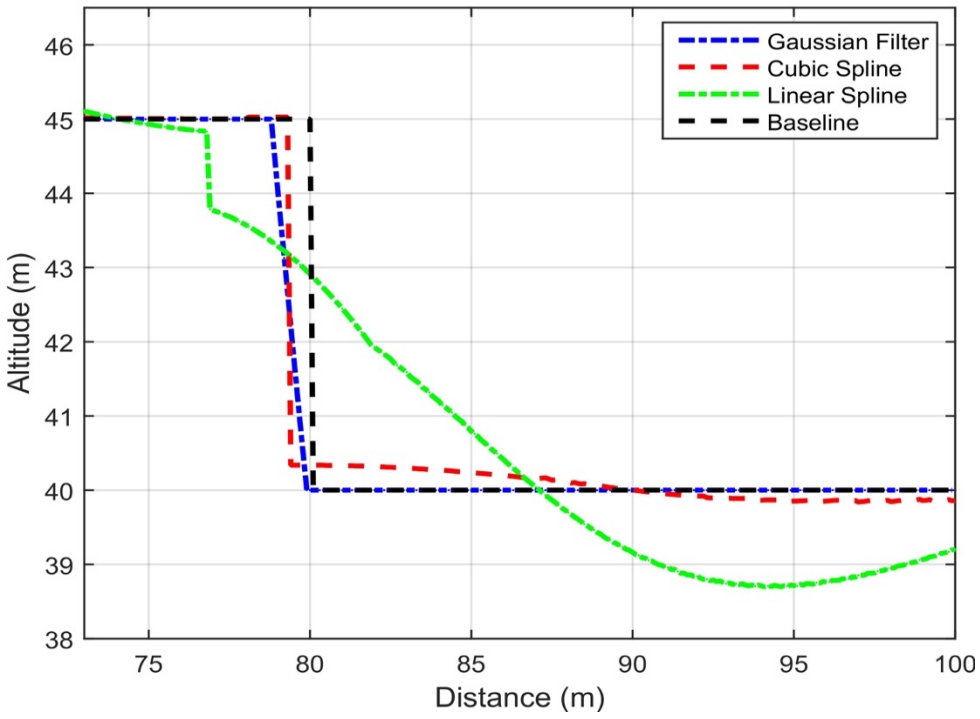

Figure 3.5: Trajectory planning of different techniques with constrains of the descending case.

The performance of the quadrotor based on these trajectory techniques is shown in Figure 3.6. It is clear that the motion of a quadrotor based on the Gaussian trajectory recovered better when it is required to follow the terrain at a certain distance from it and avoiding a sharp object. It is clear that the trajectories created using linear and cubic splines do not follow the baseline as well because the performance of the PID is not enough to help the quadrotor to closely follow the terrain. Using the traditional method, the quadrotor is able to closely follow the terrain but not in an energy-efficient manner, which then reduces the length of its flight time capabilities.

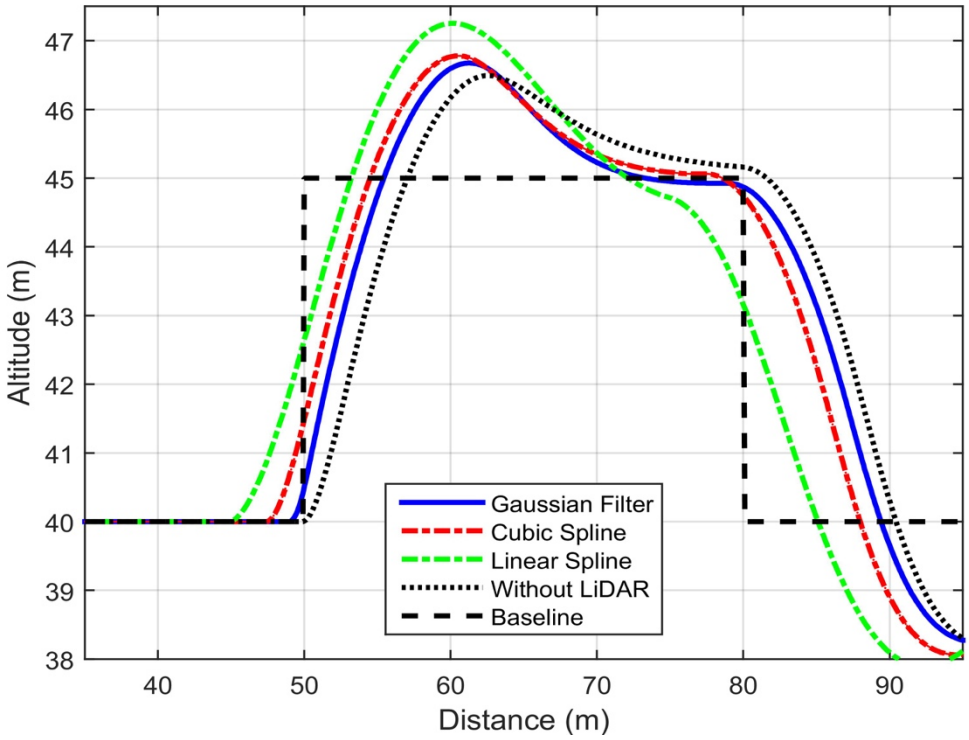

Figure 3.6: Motion planning of a quadrotor utilizing different trajectory planning techniques.

3.6.2 Case Study: Motion Planning with Gaussian Filter

It is clear that using a Gaussian filter, and in particular one set at an angle of 45 degrees, assists the motion planning algorithms to closely replicate the terrain. The combination of the

proposed trajectory planning method with the Gaussian filter means that the trajectory planning results are affected by two criteria: i) the Gaussian filter window's width (Sigma) and ii) the laser beam's angles. The simulation consisted of three scenarios. The first scenario's objective was to decide on the optimal standard deviation of the Gaussian distribution. The standard deviation was determined randomly via a number generator while the angle of the laser beam was kept the same throughout the case study. The outcome, shown in Figure 3.7, shows that the sensor's margin of error decreases in parallel with the standard deviation of the Gaussian distribution. However, the problem with this outcome is that a very subtle standard deviation results in a particularly smooth trajectory which would not take sharply protruding obstacles into account. The second case study, to determine the angle of the laser beam, was carried out in order to avoid quadrotor collisions with such obstacles. In this example, shown in Figure 3.8, the angles of the laser beams were selected at random while the filtering window was kept at a constant 15 measurements. The results suggest that quadrotors guided by a large laser beam angle coupled with a small Gaussian distribution standard deviation tend to have fewer inaccuracies in following the terrain when compared with quadrotors guided by vertical sensors. It also found that by fitting a forward-pointing laser beam, the quadrotor's navigation control cut down on energy wasted as a result of sudden trajectory corrections.

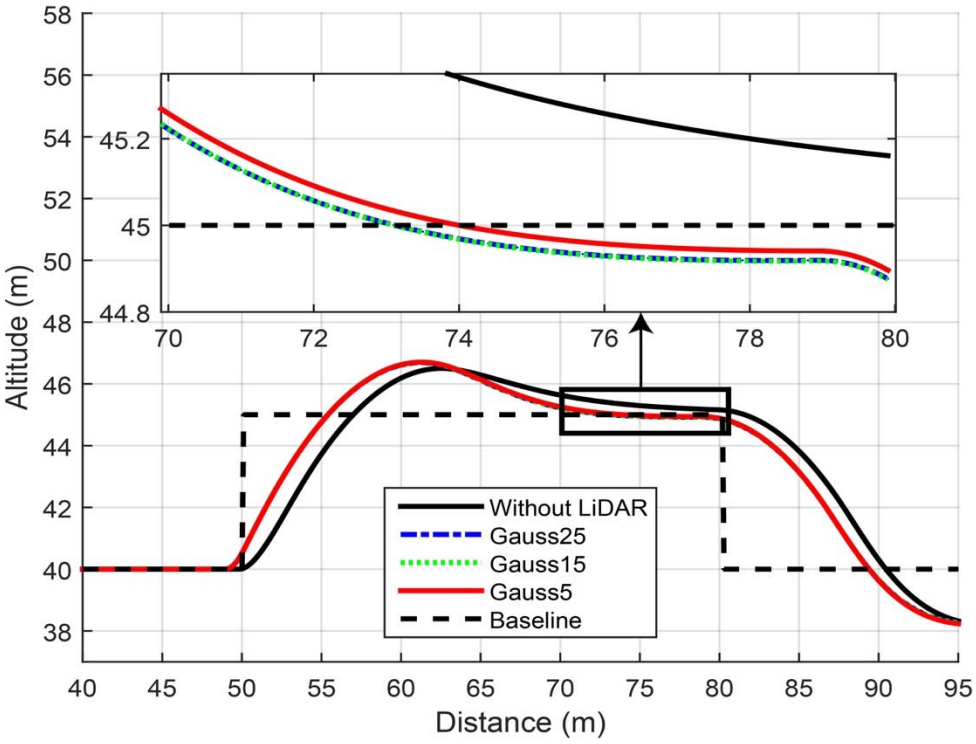

Figure 3.7: Trajectory of a fixed angle with different standard deviations.

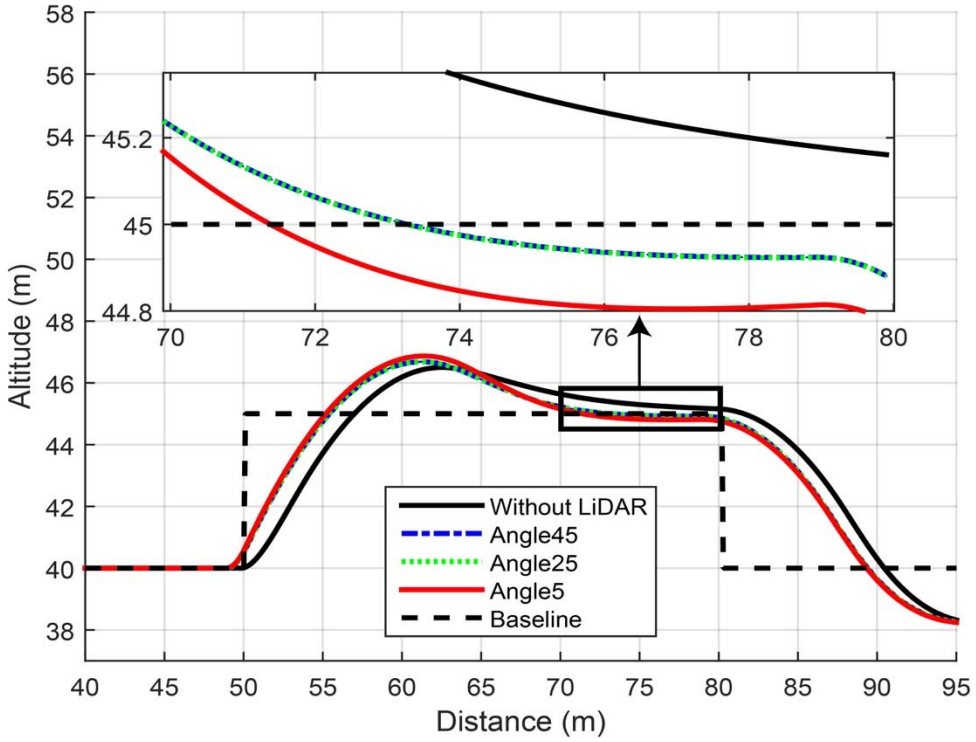

Figure 3.8: Trajectory of different angles with fixed standard deviation.

The third case study found that descent became an issue when the navigation system used a small filter window with a wide laser beam angle. In this instance, if part of the terrain became obstructed by a protruding object, the navigation system's inability to measure the full range of the environment meant that the quadrotor became in danger of descending too close to these obstructed features. To resolve this problem, the filtering window size should change depending on whether the quadrotor is ascending or descending, and in the case where sharper trajectories are expected, smaller filtering windows are used. Figure 3.9 shows the outcome of having different sized filtering windows for ascent (25 measurements) and descent (5 measurements). This modification to the proposed method rules out any chance the quadrotor will collide with a sharply protruding object and also offers the best outcome in terms of errors, compared with the two prior case studies.

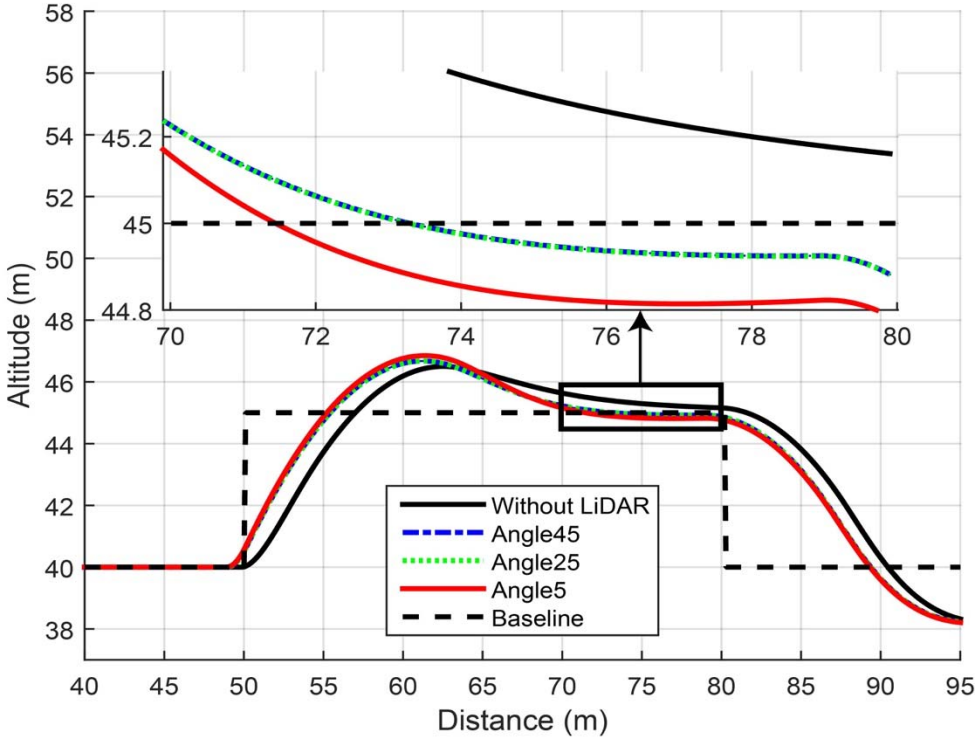

Figure 3.9: Trajectory of different angles with different standard deviations.

Chapter 4: Control Systems

4.1 Overview

This chapter aims to continue to demonstrate the effectiveness of the proposed solution while following the terrain at a maintained distance. Chapter 4 also highlights how the control system can be a factor in making the proposed solution successful. The goal of this aspect of the research was to firstly design a control system that is not only able to stabilize the inherently unstable system, but which can also generate and track a desired trajectory. In order to optimize the performance of the proposed solution, the second goal of the control system's design focused on assisting the quadrotor to maximize its flight time. This chapter focuses on three different kinds of control techniques: proportional- derivative- integral control (PID), linear quadratic regulator (LQR) with an integrator feedback, and model predictive control (MPC). Although these control techniques have already attracted many researchers interested in UAVs and their following terrain problems [51], controlling a quadrotor with actuator limitations and generating a smooth trajectory is still an active research topic [5]. In order to identify the most appropriate controller for our scenario, this chapter focuses on the design of the quadrotor controller and is presented as follows. The chapter starts with a problem statement in Section 4.2 and presents the issue of quadrotor limitations from the control perspective of the control system. Next, the general notes and equations of motions that are used in the control design are presented in Section 4.3. Section 4.3.1 presents the concept of PID and it is broken into altitude control and horizontal motion control presented in Subsections 4.3.1.1 and 4.3.1.2, respectively. Optimal control techniques are presented in two sections as follow. First, LQR is presented in Section 4.3.2 to investigate the system in terms of controllability and observability. Second, MPC is presented in Section

4.3.3 which also shows the system in terms of the digital domain. Finally, the results are presented in Section 4.4.

4.2 Problem Statement

In the last chapter, it was mentioned that the PID controller was trialled with all the trajectory techniques in order to show the effectiveness of the proposed solution in terms of trajectory planning. While the PID controller works perfectly with those trajectory planning techniques, some restrictions on the control inputs of the systems were not considered. In practice, this means that the proposed solution might not work in complex environments where control inputs may exceed the hardware limits. In addition, Figure 4.1 shows the energy consumption of a trajectory that is based on Gaussian filter with a 45 degree laser beam angle. It is clear that utilizing a lot of energy to track the desired trajectory means that the flight length, and therefore one of the thesis goals, is compromised. It also shows that the quadrotor needs to give a negative thrust to be able to quickly descend; however, the quadrotors typically does not generate a negative thrust and instead relies on gravity to descend. Although the proposed solution is effective, as demonstrated in the last chapter, it is nevertheless necessary to design a controller that takes into the account the quadrotor abilities and such practical aspects. It is also necessary to design a controller that assists the quadrotor to operate in different and complex environments.

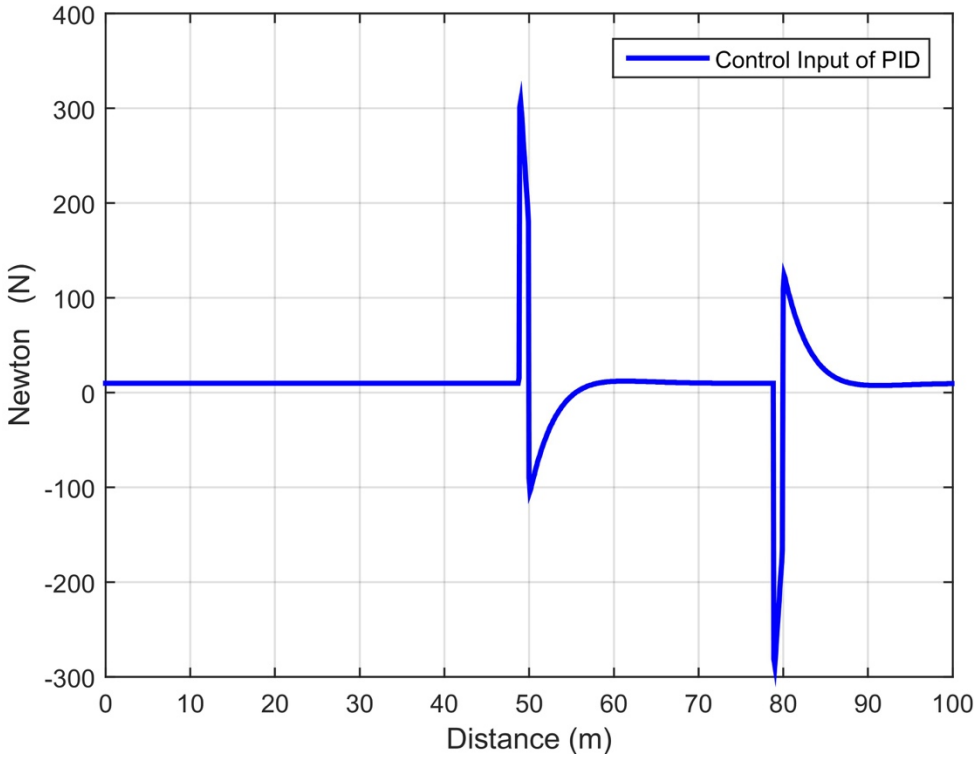

Figure 4.1: The energy consumption used by the quadrotor to follow the terrain at a certain distance.

4.3 Control Design

Although the primary goal of quadrotor research during the last decade has been to develop control laws [51], tracking smooth trajectories is still not an easy task [4]. The reason for this is that the reliable data that can feed a controller is still lacking. Moreover, designing a suitable control system depends primarily on two factors. First, as different control methodologies offer varying capabilities, identifying what is required from the controller is an initial critical step towards achieving a reasonable control performance. Second, having a proper mathematical representation of a quadrotor leads the designer to not only understand the dynamic patterns of the quadrotor but also facilitates designing the appropriate control system. In this thesis, the two criteria that the comparison of the control systems will be based on are: 1) the controller's ability to track the desired trajectory under the influence of real-world limitations and restrictions on control inputs, and 2) the contoller's ability to

maximise the quadrotor's flight time. One significant limitation affecting the quadrotor's performance is that the motor is unable to generate negative thrust. In terms of restrictions, the maximum control input is 100 N and the minimum control input is 0 N as taken from the quadrotor's specifications. Furthermore, the equations of motions that fulfill the task of maintaining a fixed distance between a quadrotor and the terrain are represented in Equation (4.1) to Equation (4.3). Equation (4.1) and Equation (4.3) are responsible for the horizontal motion of the quadrotor, while Equation (4.2) is responsible for the altitude of the quadrotor. To make the tracking error formulations more comprehensible, the control of the quadrotor's altitude is separated from the horizontal motion control.

$$\dot{V}_x = -g\theta \tag{4.1}$$

$$\ddot{z} = -g + \frac{U}{m} \tag{4.2}$$

$$\ddot{\theta} = \left(\frac{L}{I_y}\right)\boldsymbol{\tau}_\theta \tag{4.3}$$

4.3.1 PID

PID is a classic control technique that has been used in many industry projects. The basic idea of this controller is that each elements of the PID controller has its own property. Understanding these properties and finding a perfect combination of the PID parameters leads to a reliable response and a stable system. Proportional gains add energy to the system, which allows it to respond quickly, but having high proportional gains also lead the system to oscillate. Derivative gains absorb energy from the system, which leads to a slow response, while the advantage of integral gains is that they eliminate steady state errors. However, despite the simplifications of the concept portrayed here, tuning the PID parameters doesn't offer a suitable response because it requires experience with the controller and a

comprehensive knowledge of the system. The PID control technique is commonly used for tracking a trajectory [40]. Generally, the formula of PID is as shown below:

$$u\,(t)=PID\,(e(t)) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{d}{dt} e(t) \quad (4.4)$$

Where u(t) is the control signal and e(t) is the error in which the difference between the reference signal r(t) and the current state of the system y(t) is defined (i. e. e(t) = r(t) – y(t)). The control signal is a sum of a series of multiplications. First, it is a multiplication of the proportional terms k_p with the error. Then, the integral parameter k_i is multiplied by the error. Finally the control signal is completed by multiplying the derivative parameter k_d by the error as shown in Equation (4.4). Figure 4.2 presents the block diagram of PID control that has been used to track the desired trajectory generated by the trajectory planning methods. The following section describes formulations of PID altitude control and horizontal motion control.

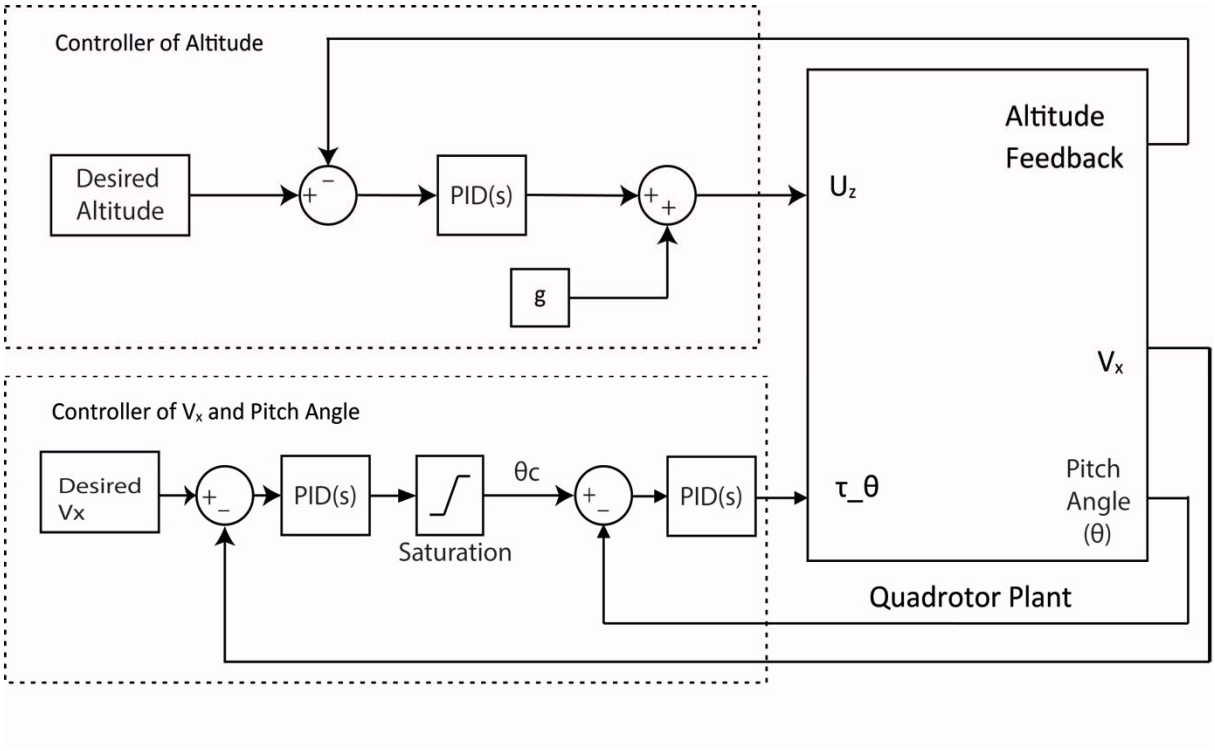

Figure 4.2: The PID control structure for a quadrotor.

4.3.1.1 Altitude Control

Altitude control is responsible for keeping the distance between the quadrotor and terrain fixed. Knowing the undulations of the terrain underneath and ahead of the quadrotor via a pointed laser at a fixed angle adds accuracy to the altitude control signals by lessening the chance of an unexpected change in the terrain. The trajectory planning algorithm feeds an optimum trajectory ahead of time to the altitude controller. Altitude control is also responsible for reducing the amount of wasted energy. Mathematically, Equation (4.2) is the only equation that is considered when evaluating the altitude control law. Where the linearized control signal for altitude control will be as shown here:

$$e_z = Z_d - Z \tag{4.5}$$

$$U(t) = m(\, g + PID(e_z)\,) \tag{4.6}$$

Equation (4.5) determines the difference between the desired altitude, which is 40 meters in this thesis, and the actual altitude of the quadrotor. This signal then forwards to the PID controller as shown in Equation (4.6). We can see from Equation (4.6) that gravity was added to the control signal before the result was multiplied by the quadrotor's mass. The reason for this is that not adding the quadrotor's mass to the dynamic quadrotor plant while adding gravity to the control signal helps the quadrotor to cancel the effect of gravity in the system.

4.3.1.2 Horizontal Motion Control

The translational movements of the quadrotor occur horizontally in the X and Y axes. In this thesis, the motion in the X axis is the only one responsible for tracking the terrain. Looking at the equations of motion, it is clear that the pitch angle affects the movement in the X direction of the body frame. Taking all of the above into account, it appears that controlling the pitch angle should be accomplished by controlling the velocity in the X direction. The dynamic equations of the motion of the X axis will be linearized around a hovering status as

shown in Equation (4.1) [3]. However, the control signal of the horizontal motion is initiated by comparing the desired velocity of the X axis with the actual velocity. Subsequently, utilizing the velocity error via PID algorithms results in the desired pitch angle command. Finally, controlling the error of the pitch angle via PID algorithms occurs after comparing the actual and desired pitch angle command. The algorithms of the control signal of the horizontal motion are shown below:

$$e_v = Vx_d - Vx \tag{4.7}$$

$$\theta_C = -\frac{1}{g} * PID(e_v) \tag{4.8}$$

$$e_\theta = \theta_C - \theta \tag{4.9}$$

$$\boldsymbol{\tau}_\theta(t) = I_{XX} * PID(e_\theta) \tag{4.10}$$

Finally, the PID gains of altitude control and motion control that will ensure the smoothness of the terrain tracking are displayed in Table 4.1.

Table 4.1: Values of the PID control gains.

Gains	**Altitude**	**Horizontal Velocity**	**Attitude**
Kp	13.8	9	0
Ki	8.27	0	0
Kd	6.07	0.5	0.01

4.3.2 LQR

The linear quadratic regulator is one of the optimal control techniques currently used on UAV platforms [41, 52]. It is based on minimizing the quadratic cost function which is then used to calculate the input to a linear system. LQR is perfect for a system that needs to turn all states to zero (i.e., a regulator) [41]. To use this control method in a tracking task, the LQR control architecture is modified by adding an integral action into the LQR feedback loop shown in Figure 4.3. In order to design an LQR controller, the dynamic system of a quadrotor is represented in a given equation:

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u \tag{4.11}$$

$$u = -Kx \tag{4.12}$$

$$J = \int_0^\infty (x^T \boldsymbol{Q}x + u^T \boldsymbol{R}u)\, dt \tag{4.13}$$

Equation (4.11) presents a system model where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the states and inputs of a quadrotor, respectively. Quadrotor system generally has twelve states as a result of six equation of motions, but because the quadrotor has four motors, the system actually has four inputs. Equation (4.12) presents the minimizing control input of LQR where K [K K_i] is a state feedback gain. A quadrotor's high performance will be achieved by minimizing the control input [41]. The cost function is a criterion that will minimize the control input as shown in Equation (4.13) where $\boldsymbol{Q}$ is a positive semi definite state control matrix, whereas $\boldsymbol{R}$ is a positive definite performance matrix. The importance of these matrices is that they help to weigh every state individually, which means a designer can choose which states are critical or not [52].

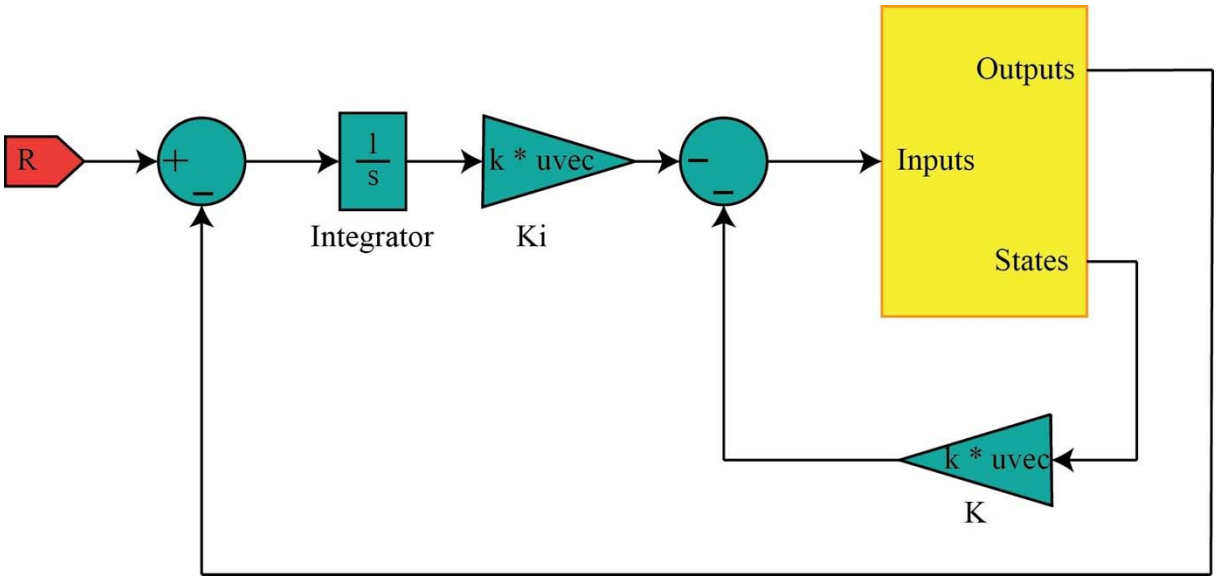

Figure 4.3: The general LQR structure for a quadrotor.

The state space model of a quadrotor based on the Equations (4.1) to (4.3) are presented in the following. Equations (4.14) presents the state space representation in time domain,

whereas Equations (4.15) presents the state space representation in digital domain. The state space representation in digital domain will be used in MPC controller as a result the MPC works only in digital domain. The states of the systems are $[\theta \quad z \quad \dot{\theta} \quad V_X \quad \dot{z}]$. From the states, the resulting state vector was $R^{5\times1}$. Now, this system is controllable and observable, which means it can be used to decouple the controller in order to simplify the system.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \left(\frac{L}{I_y}\right) \\ 0 & 0 \\ \left(\frac{1}{m}\right) & 0 \end{bmatrix} \begin{bmatrix} U \\ \boldsymbol{\tau}_\theta \end{bmatrix} \tag{4.14}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$\begin{bmatrix} x_{k+1} \\ x_{k+1} \\ x_{k+1} \\ x_{k+1} \\ x_{k+1} \end{bmatrix} = \begin{bmatrix} 1.012 & 0 & 0 & 0.0502 & 0 \\ 0 & 1 & 0 & 0 & 0.05 \\ 0 & 0 & 1 & 0 & 0 \\ 0.4925 & 0 & 0 & 1.012 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ x_k \\ x_k \\ x_k \\ x_k \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0.00125 & 0 \\ 0 & 25 \\ 0 & 0 \\ 0.05 & 0 \end{bmatrix} \begin{bmatrix} U \\ \boldsymbol{\tau}_\theta \end{bmatrix} \tag{4.15}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_k \\ x_k \\ x_k \\ x_k \end{bmatrix}$$

The altitude state space presentation is presented in Equation (4.16). It is clear now the system has been simplified and is controllable and observable. The only item missing in this system is the tuning of weighting matrices. The weighting matrices used in the altitude controller are presented in Equations (4.17). Where Q_Z is a positive- semi definite state control matrix for the position in Z axis and R_Z is a positive definite performance matrix for

the position in Z axis. K_Z is the gain feedback that is responsible for minimizing the energy and tracking the desired trajectory.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} U \tag{4.16}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$Q_Z = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 10000 \end{bmatrix} \qquad R_Z = 10 \qquad K_Z = \begin{bmatrix} 20.0002 & 6.3247 & -31.6228 \end{bmatrix} \tag{4.17}$$

The Pitch and V_x state space presentation is presented in Equation (4.18), and the values of the weighting matrices are presenting in Equation (4.19). Hence, the previous and following matrices show that the system is controllable and observable, and that it can be used in the system to track the desired trajectory as we will see in the result section.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 9.81 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{L}{I_y} \end{bmatrix} \boldsymbol{\tau}_\theta \tag{4.18}$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$Q_{\text{pitch \& x}} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix} R_{\text{pitch \& x}} = 10$$

$$K_{\text{pitch \& x}} = \begin{bmatrix} 10.1344 & 1.0201 & 4.6564 & -10 \end{bmatrix} \tag{4.19}$$

4.3.3 MPC

Model Predictive Control, also referred to as Receding Horizon Control (RHC), is an optimal control technique frequently used on UAVs. The main goal of this control technique is to minimize a given objective function over a certain future time interval, while taking the plant's constraints into account. The formulation of MPC is shown below [53]:

$$\min J(t) \tag{4.20}$$

Subject to

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

$$x_0 = \boldsymbol{x}(\boldsymbol{t})$$

$$u_{min} \leq u \leq u_{max}$$

where $J\ (t)$ is the objective function, $\dot{x}$ is a state equation, and x_0 is the initial state of the model plant. The inputs u_{min} and u_{max} specify the lower and upper boundaries of the input, respectively. The objective function used in this thesis is shown below [54], and its parameters are described in Table 4.2.

$$J = \frac{1}{2}\, x_N^T \boldsymbol{P}\, x_N + \frac{1}{2} \sum_{\boldsymbol{k=0}}^{\boldsymbol{N-1}} (x_N^T \boldsymbol{Q}\, x_N + u_k^T \boldsymbol{R}\, u_k) \tag{4.21}$$

Table 4.2: The values of MPC parameters.

#	**Parameter**	**Value**
1	Optimisation Horizon (N)	50
2	State Weight Matrix (Q)	diag(100, 0.01)
3	Control Weighting Matrix (R)	0.01
4	Terminal State Weighting Matrix (P)	diag(500,500)

4.4 Results

In order to show the effectiveness of the control system on the proposed solution, the trajectory of Gaussian filter at a 45 degree angle has been selected as the baseline for the three controllers. This chapter demonstrates the effectiveness of each controller in limiting the input energy and tracking the desired trajectory. Figure 4.4 shows the responses of the two PID controllers. It is clear that the controller with no limitations on the control input is better; however, it utilized a lot of energy to track the desired trajectory as it is shown in Figure 4.1. Figure 4.5 shows the response of the three types of controllers. It is obvious that the response of MPC is the best due to the fact that the concept of MPC takes into account the constraints of the system. The response of the three controllers is almost the same in

terms of descent as the quadrotor does not have a negative thrust. Finally, Figure 4.6 shows the value of control inputs from all controllers. The result demonstrates that the type of controller used has an impact on how much energy is consumed. Although PID exceeded the input constraint trying to fulfil the task, it is still the simplest control architecture. On the other hand, the control inputs of LQR and MPC are in the range of the constraints, but LQR has a slow response and a high overshoot as it is shown in Figure 4.5. The reason that MPC offers the best outcome is that it aims for high performance by working closely within the boundaries of the system. Generally speaking, the results prove that all of the proposed controllers can be applied in real life applications but MPC may be preferable where system constraints are critical.

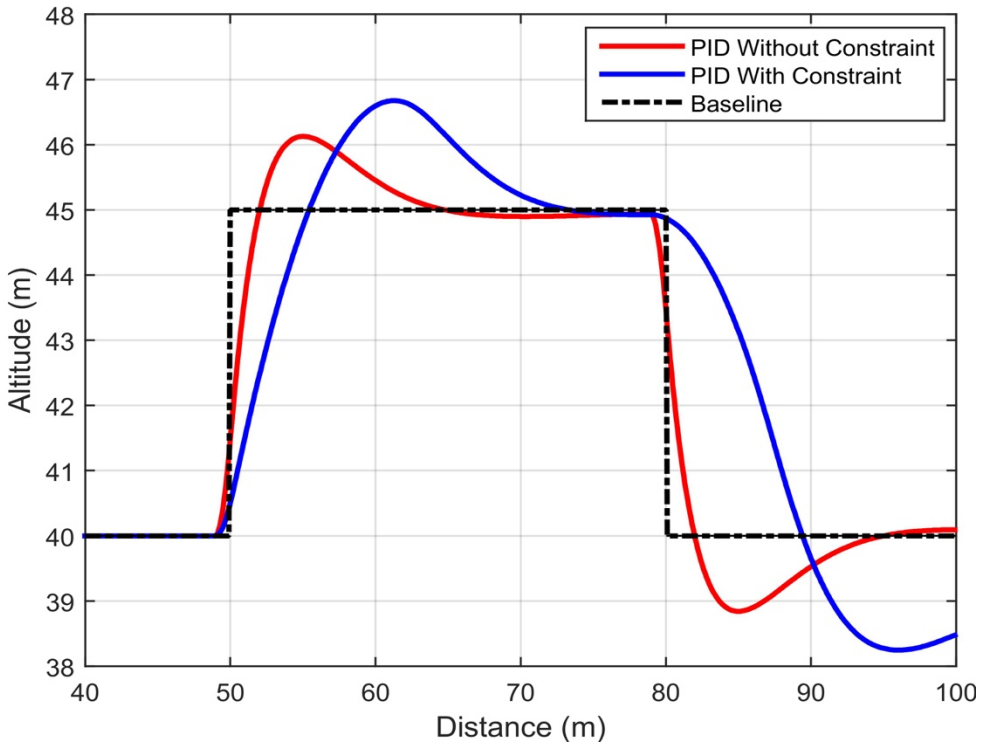

Figure 4.4: Control responses of a PID controller with and without limitations on the motors.

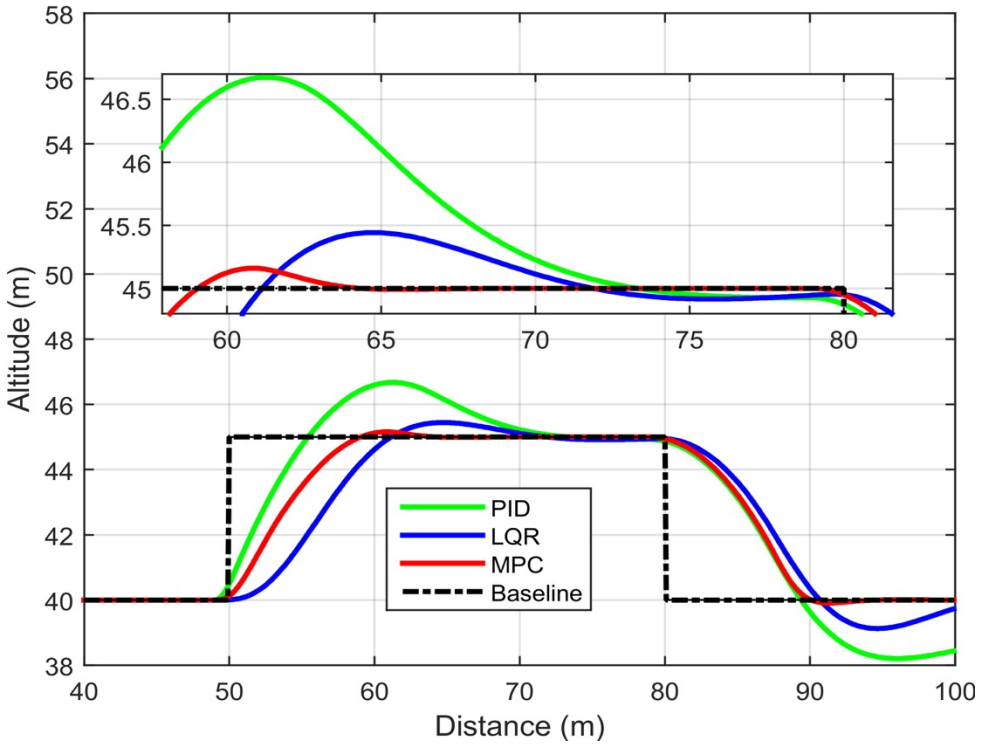

Figure 4.5: Control responses of a 45 degree angle with Gaussian filter as a trajectory planning.

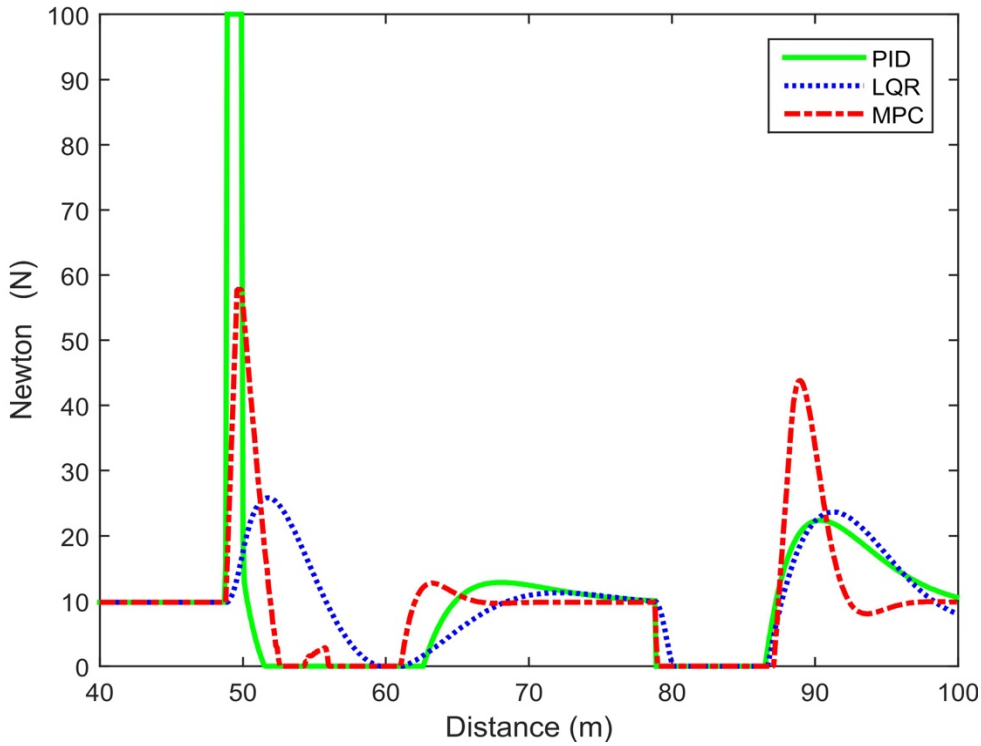

Figure 4.6: The value of control input of all controllers' techniques on the range of the input limitations.

Chapter 5: Simulation and Verifications

5.1 Overview

This chapter aims to analyze the effectiveness of utilizing a single-beam LiDAR sensor to send information to the controller about undulations in the terrain beneath and ahead of the quadrotor. The LiDAR was attached to the base of the quadrotor and then set at varying angles to gauge the effectiveness of each. The proposed solution was simulated on a software platform. This chapter starts with a brief introduction about the software platform and then presents the simulation results in Section 5.3. It concludes with summarizing the most important outcomes of the simulation results in Section 5.4.

5.2 Software Platform

For this thesis, a software platform has been developed to compare and verify the efficacy of the proposed trajectory planning and control methods. This software platform features a graphical user interface (GUI) developed in Matlab. The role of this platform was not only to validate the research objectives, but also to offer an educational tool for the engineering students to learn about motion planning and control algorithms. This platform simulates the motion of a quadrotor over a terrain profile in two dimensional planes: travelled distance and elevation. It also includes many control options for the user to create different scenarios and see how these affect the motion of the quadrotor. Figure 5.1 shows the GUI of the simulation program. The platform's features and instructions are presented in Appendix A. The software Matlab code is presented in Appendix B.

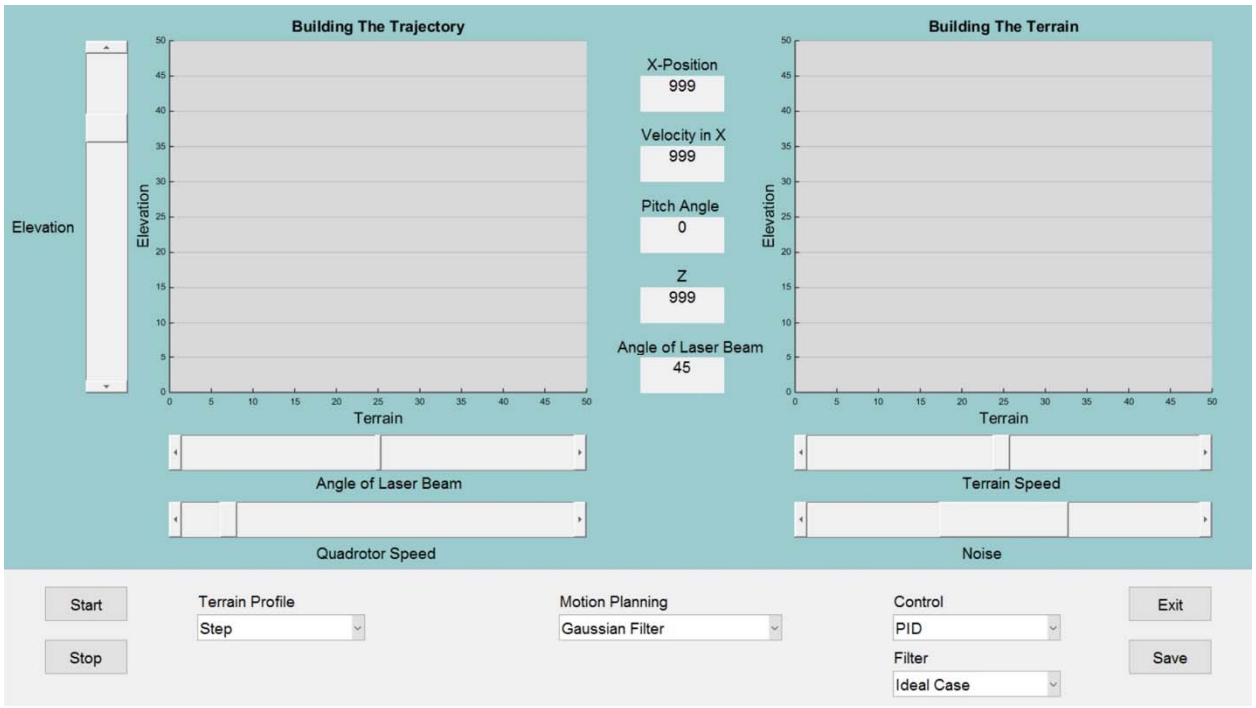

Figure 5.1: The software platform of this research.

5.3 Simulation Results

This section provides the results of the simulation to demonstrate effectiveness of the proposed solution within different scenarios or environments. All these results have been simulated numerically in the software platform. The results are presented in graphs and tables, similarly to the results in Chapters 3 and 4, and start with the differences in simulated trajectories of quadrotors using the motion planning algorithms linear spline, cubic spline, and Gaussian filter. In the case of complicated terrain profiles, the ascent and descent of the quadrotor's trajectory is also displayed. Finally, the motion planning of all these techniques are presented before the results conclude with an analysis of the performance of these trajectories if forwarded to advanced controllers like LQR and MPC. This section contains analysis of five different environments. The first three profiles are well known scenarios: trapezoidal, ramp, and sinusoidal. The other two are combined profiles: step and sinusoidal, and double step.

5.3.1 Case Study: Trapezoidal profile

A trapezoidal profile presents a case where the terrain gradually elevates. The advantage of this profile is that its gradual and predictable environmental changes are easy for the traditional method of sensing to detect, meaning that the chance of needing to make a sudden recovery or encountering missing terrain data is quite rare. Although our proposed solution may be unnecessary in this particular case, the overall results of the simulations indicate that this new technique would be more effective than the standard method when flying in more complex environments. Figure 5.2 shows the trajectories of quadrotors using a Gaussian filter and both splines, as well as a trajectory based on a vertical sensor. It is clear that the best trajectory is the traditional method seen in Figure 5.2 (d) where a LiDAR sensor is not used. However, we can see from Figure 5.2 (a) and Figure 5.2 (b) that in terms of planning ahead, the trajectory of Gaussian filter and cubic spline are the best because they are able to maintain their trajectory at a set distance from the terrain more reliably than the standard method of sensing. Table 5.1 presents the RMS of all methods. This table deduces that the trajectory based on a vertical sensor offers fewer errors than the trajectories of the splines, but more errors than the trajectory of the Gaussian filter. Generally speaking, for this type of profile, utilizing Gaussian Filter as a trajectory planning algorithm in conjunction with the proposed forward-facing LiDAR solution offers the best results in terms of planning ahead and accuracy.

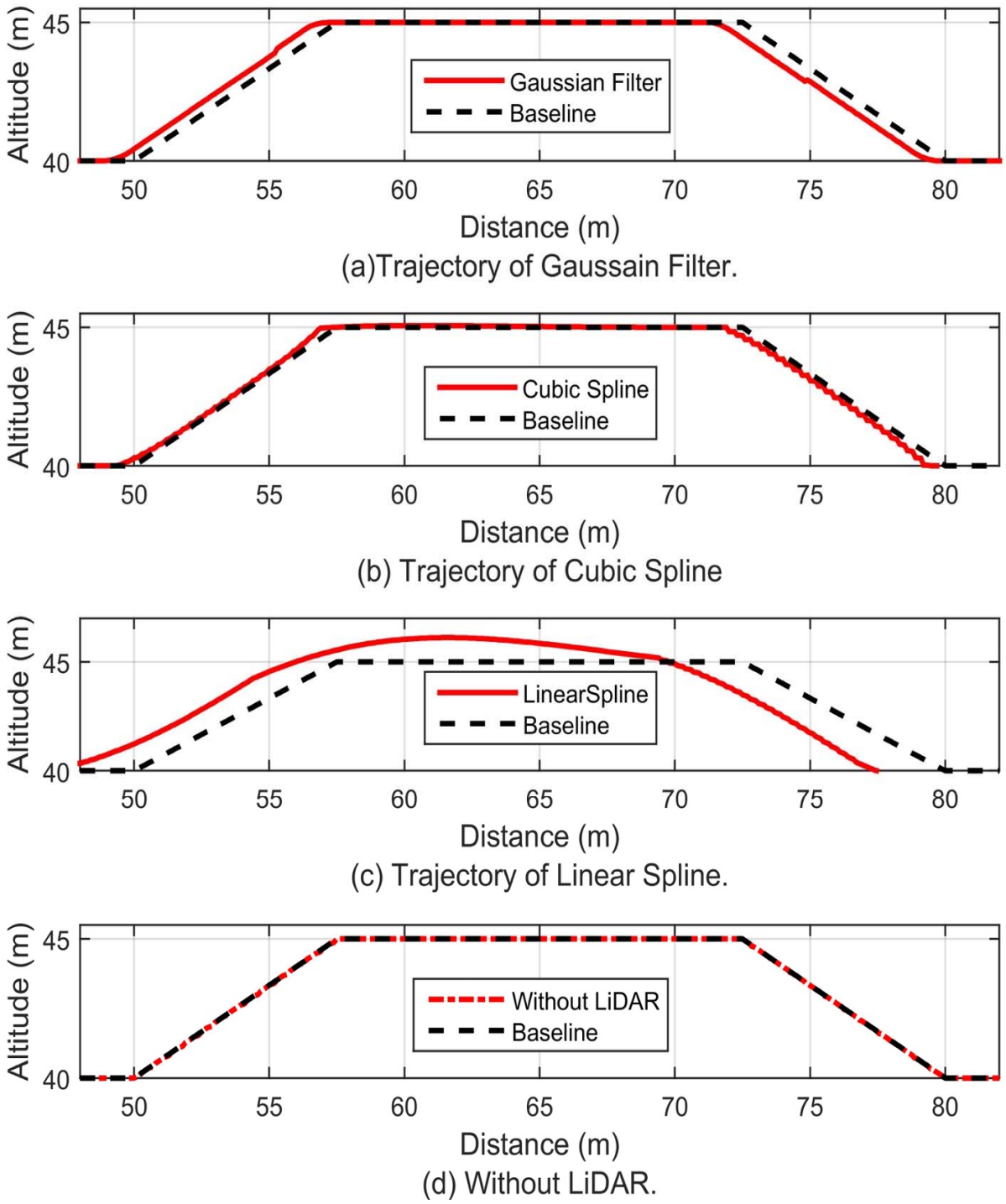

Figure 5.2: Trajectory of Trapezoidal profile in all techniques.

Table 5.1: Root mean square deviation of the different trajectory planning techniques of trapezoidal profile.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	29.0126	12.3606	6.4240	7.5845
45	28.6628	12.1566	6.3959	
35	28.7433	12.1648	6.4201	
25	28.7185	12.1723	6.4120	
15	28.6716	12.1582	6.4169	

Figure 5.3 and Figure 5.4 show the trajectory of the quadrotor in ascent and descent. As mentioned above, the traditional method of sensing works perfectly in this case, in theory. On the other hand, it is worthwhile to note that the trajectory of the traditional method of sensing is required to inform the controller to follow the imaginary profile of Trapezoidal at a defined distance from the terrain, and that this would not always be possible within its limitations. The quadrotor's controller would not always be able to follow the trajectory as precisely as required for this type of terrain, especially in terms of maintaining the quadrotor at a certain altitude. In light of this, the trajectories offered by the Gaussian and both spline methods are better than the trajectory based on data from the vertical sensor. It is clear from Figure 5.3 and Figure 5.4 that the quadrotors using cubic spline and the Gaussian filter methods are able to maintain a more reliable distance from the baseline than the quadrotor using the traditional method. Figure 5.5 also shows the performance of quadrotor motion planning when acting on this type of environment and offers a clear indication of the controllers' effectiveness in following the terrain. To further clarify these results, it should be mentioned that because of uncertainties in the environment and the quadrotor's inability to employ downward thrust, the traditional method of sensing would not be able to faithfully track the trajectory as shown, even if the motion planner had all the data about the environments and was able to forward a smooth trajectory to a PID controller. That leads us

to deduce that even though the proposed solution is unnecessary in the case of a gradually changing terrain, it helps to provide a smoothtrajectory that the controller would actually be able to follow faithfully, as shown in Figure 5.5.

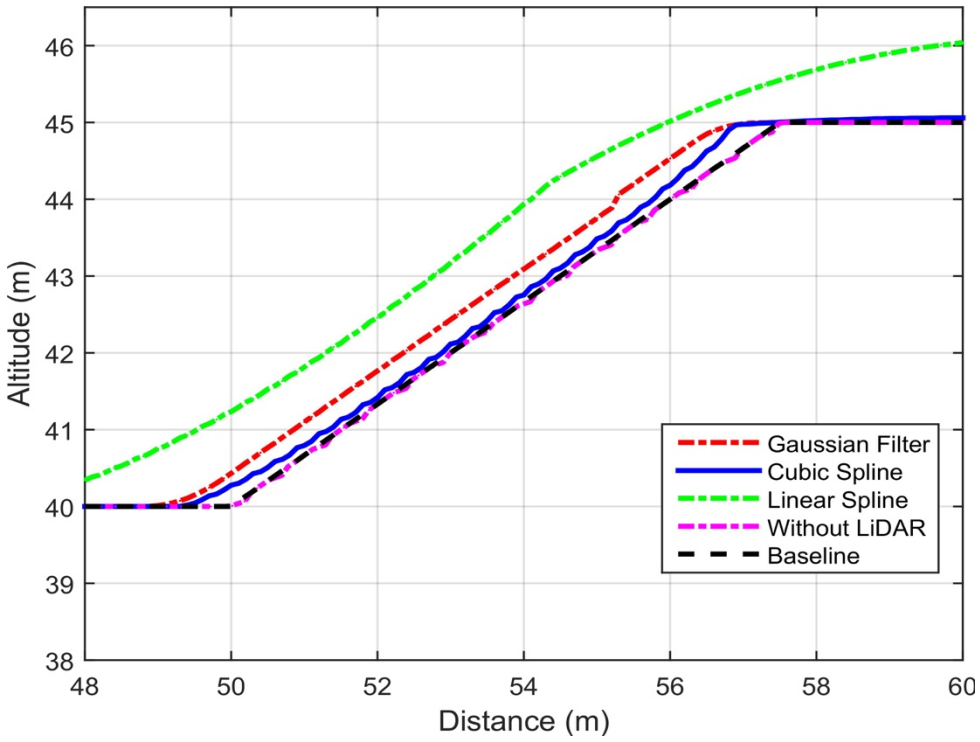

Figure 5.3: Trajectory of Trapezoidal profile in ascending case.

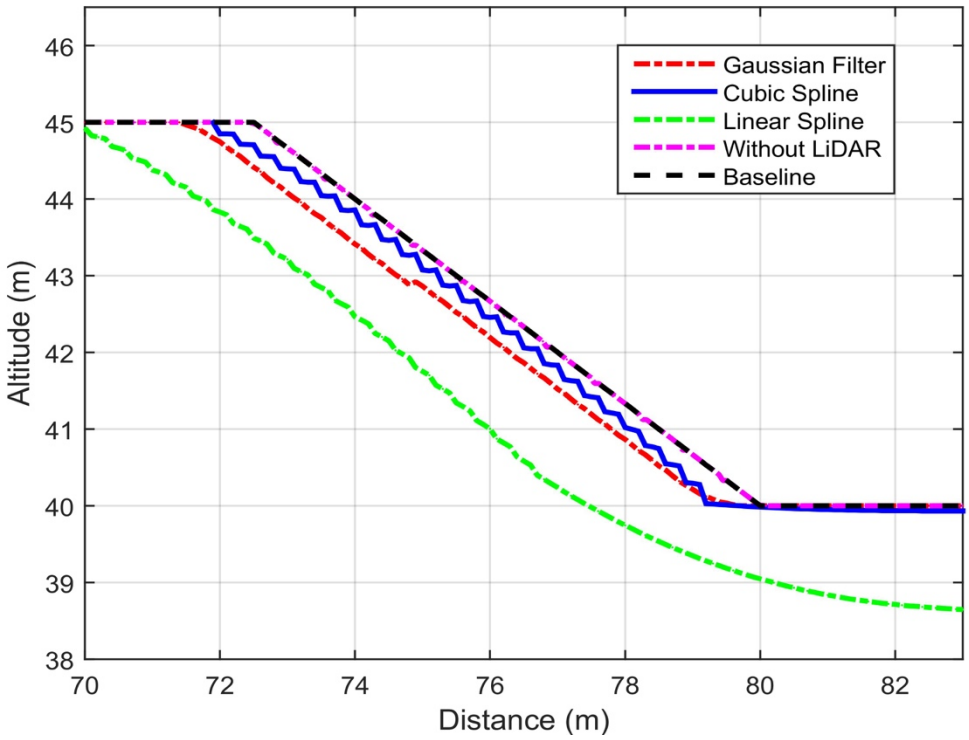

Figure 5.4: Trajectory of Trapezoidal profile in descending case.

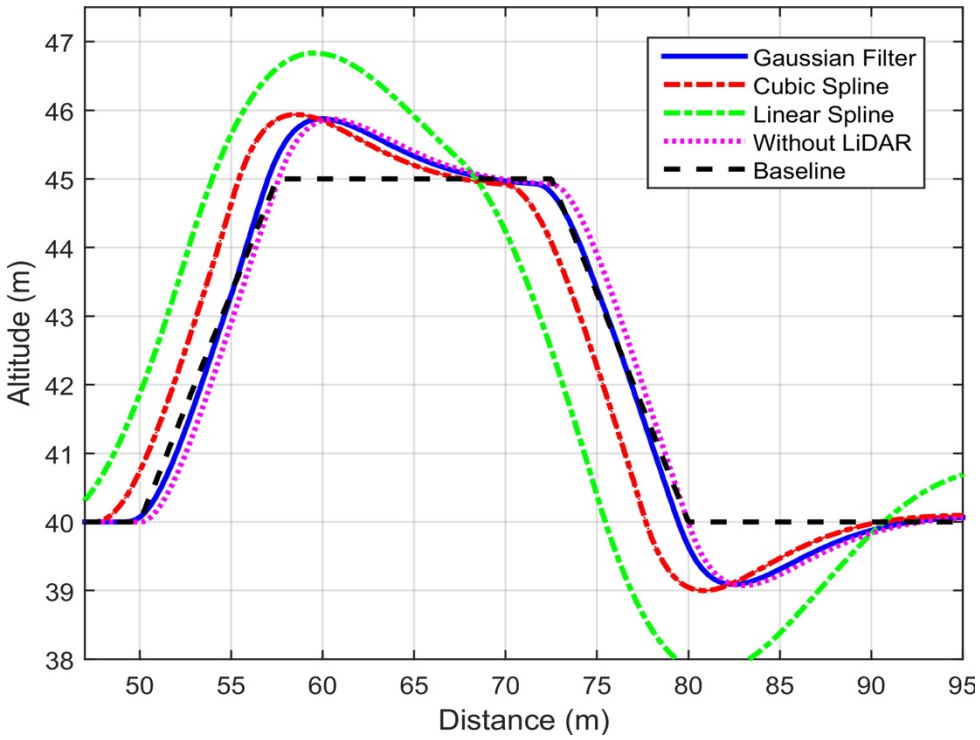

Figure 5.5: Motion planning of a quadrotor acting on a Trapezoidal profile.

In order to optimize the energy consumption for the proposed solution, different controller responses were analyzed acting on the best trajectory. Figure 5.6 shows different controller responses to the trajectory of the Gaussian filter with a 45 degree laser beam angle. It is clear that the PID controller would not be able to reliably track the trajectory under the existing constraints on the inputs of the system, whereas the other controllers would manage to accurately track the trajectory under the same conditions. It is obvious that the response of the MPC controller is the best due to the fact that the concept of MPC takes the constraints of the system into account. Figure 5.7 shows errors made by the control response.

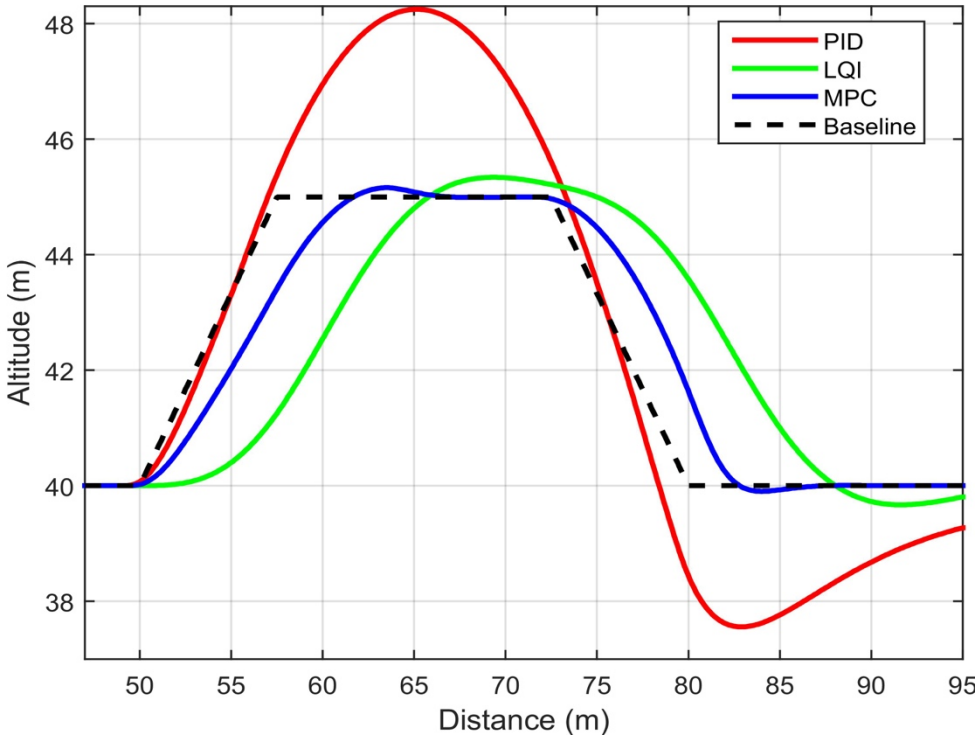

Figure 5.6: Control response of a 45 degree angle sensor with Gaussian Filter as the trajectory planning algorithm.

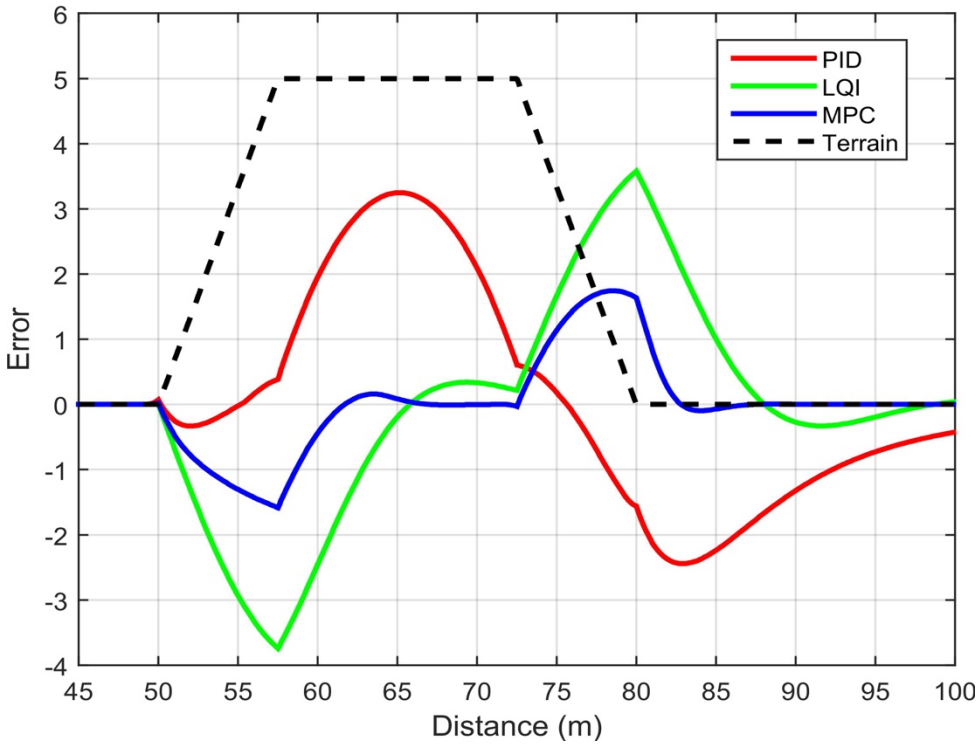

Figure 5.7: Error of the control response.

5.3.2 Case Study: Ramp profile

The Ramp profile presents a case where the elevation of the environment changes slowly and gradually up to a sharp peak. The advantages of this profile are like those of the Trapezoidal profile, but there is a greater likelihood of having missing information after the sharp peak. Figure 5.8 and Table 5.2 lead us to the same conclusion as for the previous case: where if the environment changes slowly and gradually, utilizing Gaussian Filter as a trajectory planning algorithm with the proposed forward-facing LiDAR sensor is the best option in terms of planning ahead and minimizing errors.

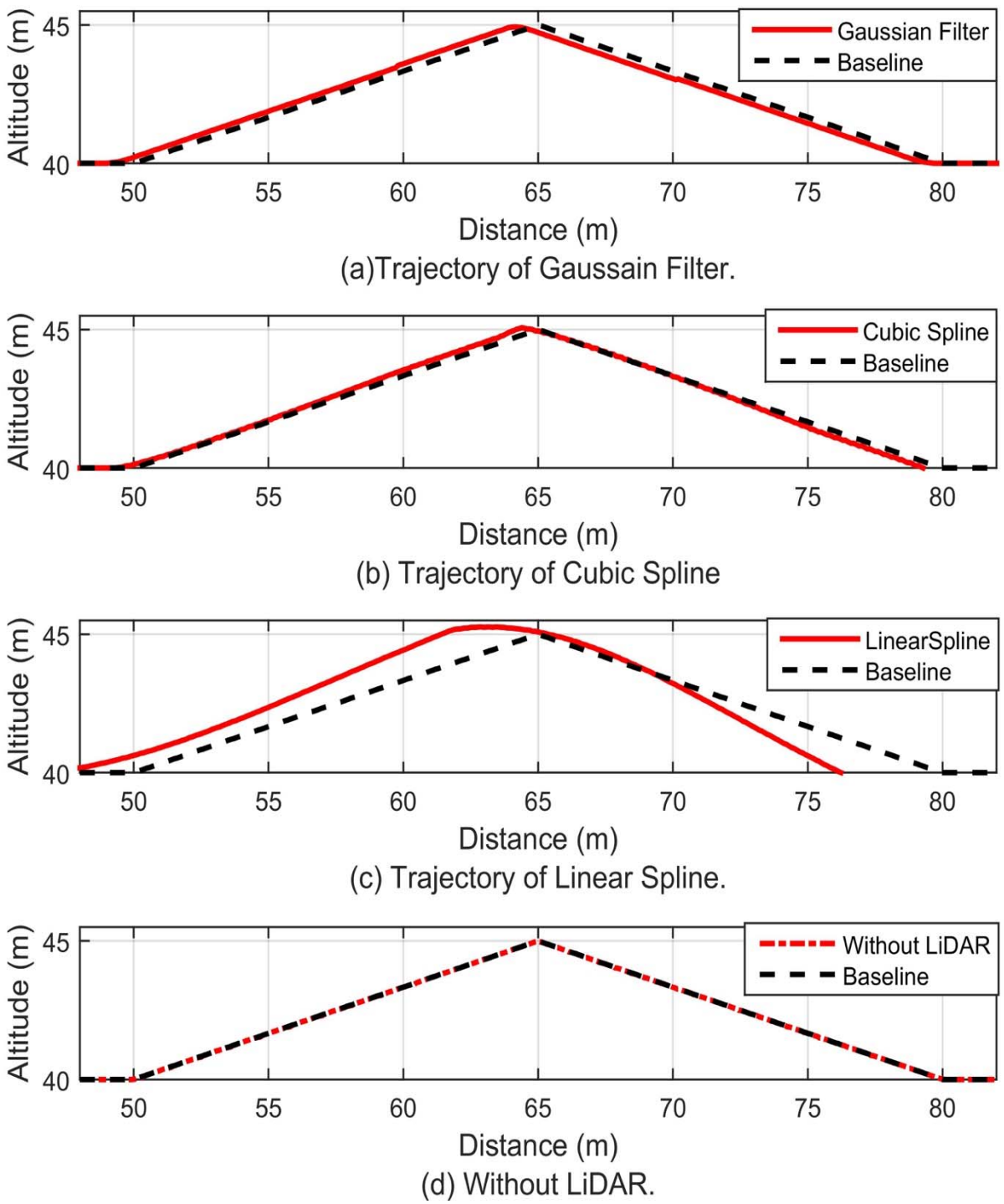

Figure 5.8: Trajectory of Ramp profile using all techniques.

Table 5.2: Root mean square deviation of the different trajectory planning techniques on a ramp profile.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	28.8004	13.7313	6.0751	6.5424
45	28.6591	13.6381	6.0481	
35	28.8008	13.7314	6.0887	
25	28.7998	13.7279	6.0866	
15	28.8086	13.7054	6.0445	

Figure 5.9 shows the trajectory of Gaussian filter and cubic spline before and after the peak. This figure presents how the differential constraints affect the trajectory. It is clear that the Gaussian trajectory is smoother and, therefore, better than that of the cubic spline, but both of them build a trajectory that helps the quadrotor to maintain its desired altitude from the terrain. This figure also shows the trajectory based on a vertical sensor. It seems perfect in terms of replicating the terrain; however, it is not feasible for a quadrotor to track. Figure 5.9 does not show the trajectory of linear spline as it would not work for this type of profile when considering the differential constraints affecting the performance of this technique. Figure 5.10 shows the motion planning of a quadrotor utilizing all the trajectories on a ramp profile. This figure leads us to the same issue in the previous case study about the controller's ability to track the trajectory. Again, we see that the proposed solution increases the quadrotor's ability to follow the terrain while maintaining a certain altitude from the terrain.

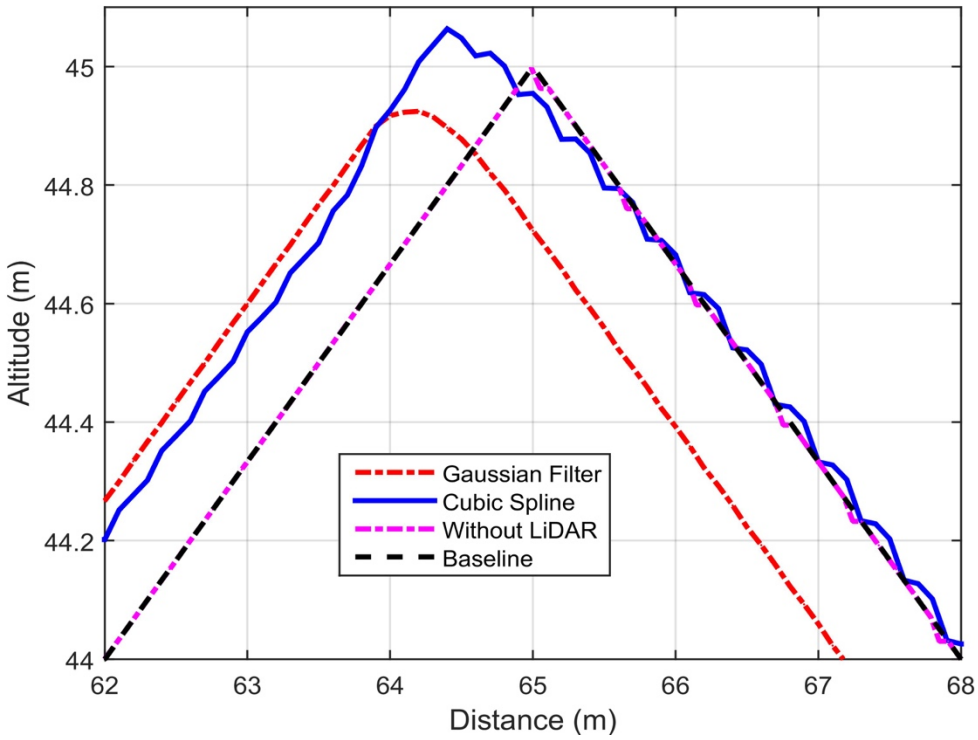

Figure 5.9: Trajectory of Ramp profile at the peak.

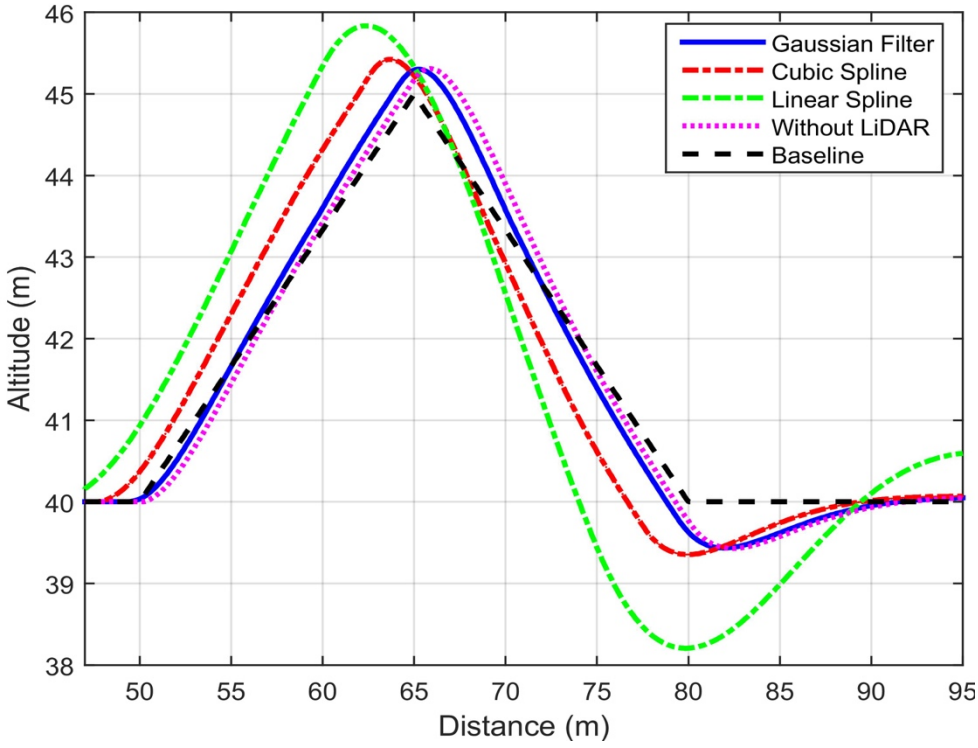

Figure 5.10: Motion planning of a quadrotor acting on a Ramp profile.

Figure 5.11 shows different control responses acting on the trajectory of Gaussian filter with a 45 degrees laser beam angle, while Figure 5.12 shows the errors of the control response. As also seen in the previous case, the response of the MPC is the best. However, in terms of descent, it is worthwhile to mention that the response of all three controllers is almost the same because most quadrotors are not able to generate negative thrust, meaning that they generally rely on gravity to descend.

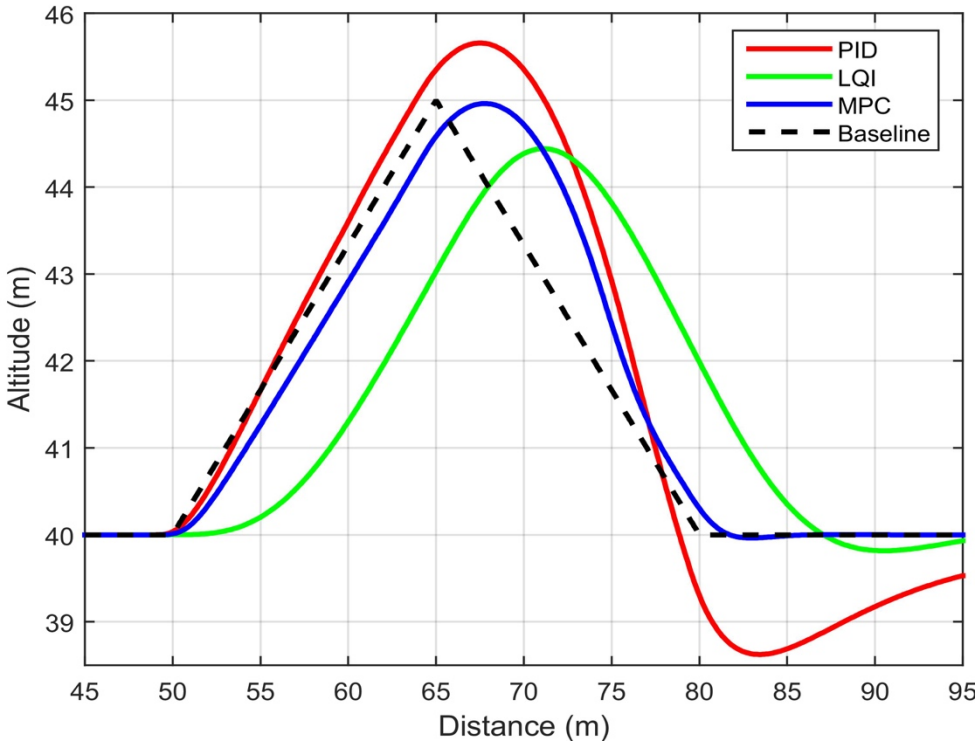

Figure 5.11: Control response of a 45 degree angle laser with Gaussian Filter as a trajectory planning algorithm.

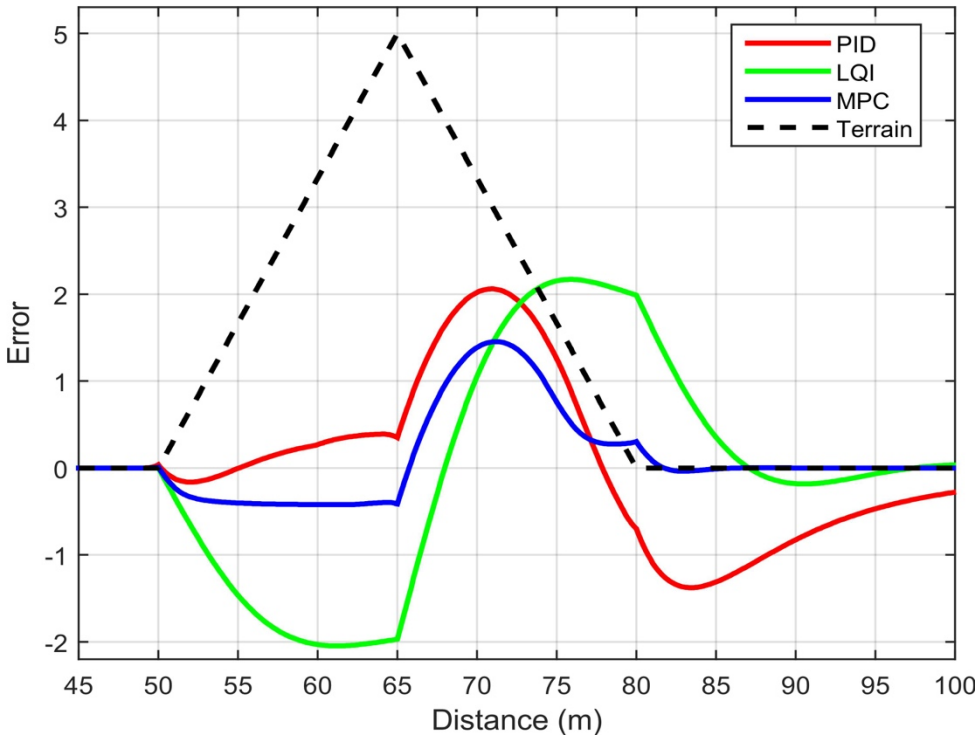

Figure 5.12: Error of the control response.

5.3.3 Case Study: Sinusoidal profile

A sinusoidal profile is one which exhibits a repetitively oscillating terrain. The reason for using this profile is that it highlights the laser beam angle effectiveness in repetitive terrain profile. In a step profile, the results indicate that having a large angle helps to reduce errors in generating the terrain. It is, however, the opposite in this case study where we see that setting a small laser beam angle assists to reduce errors when carrying out the same task. To further explain this point, the sinusoidal profile in this case study has high sinusoidal amplitude modulation which makes it impossible for the LiDAR with a large laser beam angle to detect the terrain within the dips of the profile as a part of the profile is occluded behind the peak. Therefore, it is important to set a smaller laser beam angle to reduce the amount of missing data about the terrain. Figure 5.13 shows the trajectories of all the planning methods. It seems to suggest that the best trajectory is the traditional one but, again, in reality this method is not

applicable in real world conditions for the same reasons as discussed in previous case studies. The linear spline would also not work in this type of environment because of the dramatically variable nature of the terrain. Table 5.3 presents the errors made by all techniques. It indicates that the relationship between decreasing errors in replicating the terrain is proportional to the decrease in the angle set of the laser beam.

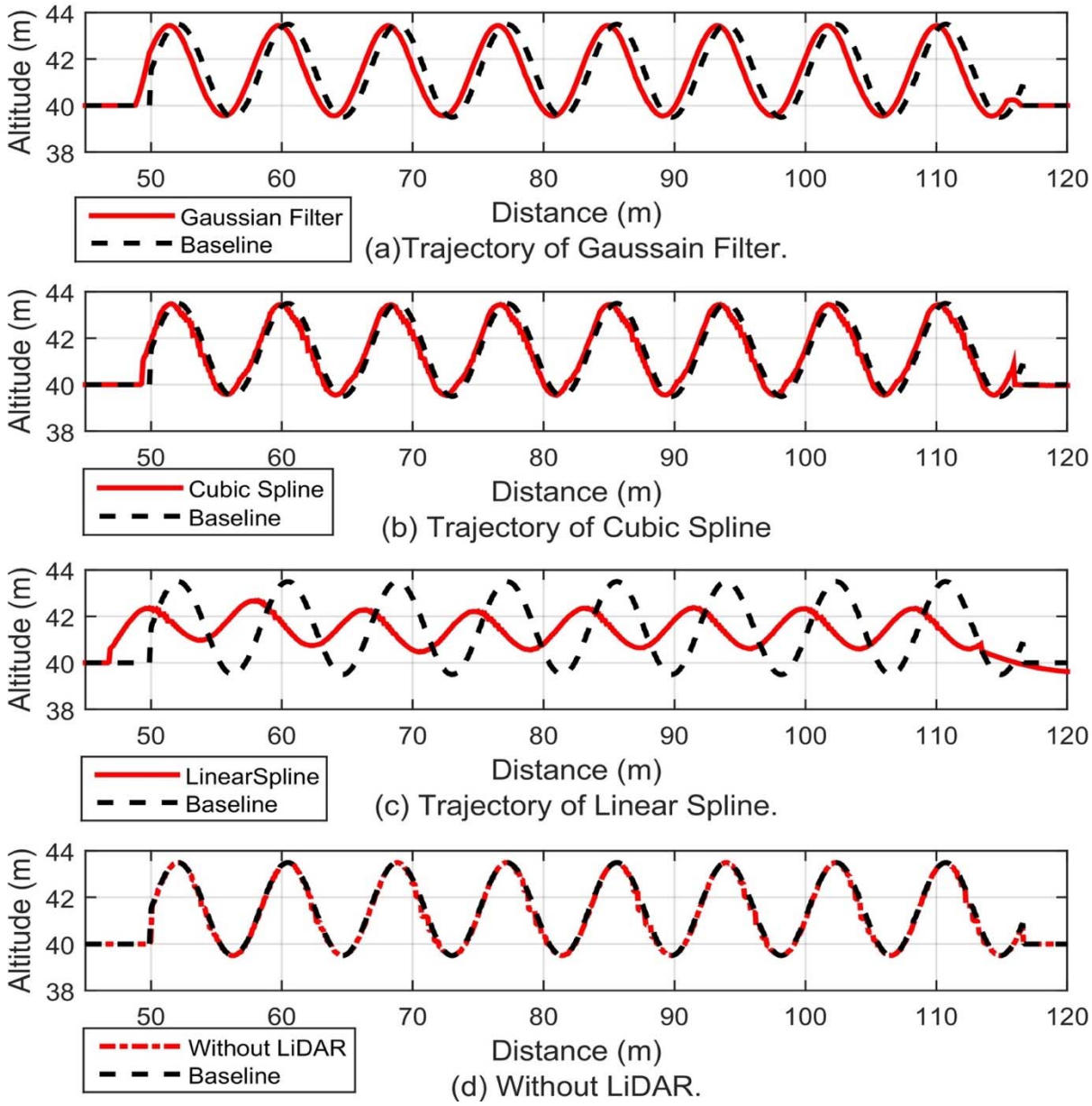

Figure 5.13: Trajectory of the Sinusoidal profile using all techniques.

Table 5.3: Root mean square deviation of the different trajectory planning techniques of the sinusoidal profile.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	50.0532	43.4024	20.3869	31.8090
45	51.2742	44.1655	20.9437	
35	48.2115	33.9871	16.4083	
25	46.1970	23.5530	20.7916	
15	46.1130	23.1026	21.5569	

Figure 5.14 shows the motion planning of a quadrotor acting on a sinusoidal profile. It is worthwhile to note that none of the motion planning techniques reach the peak of the baseline because they are attempting to maintain the altitude of the quadrotor at a certain distance from the terrain. The motion planning performance of the Gaussian filter method is noticeably more advanced than the others. Thus, we can surmise that gathering more information about the undulations in the terrain ahead assists the quadrotor to perform better in terms of navigation in difficult environments.

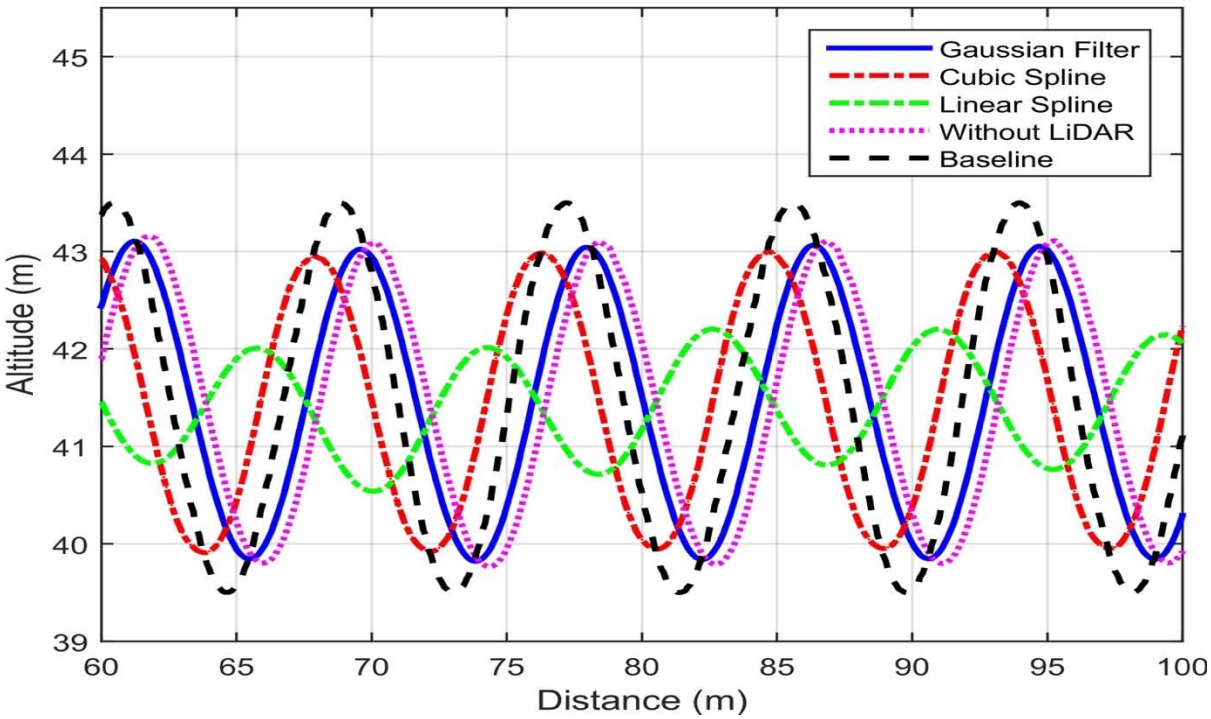

Figure 5.14: Motion planning of a quadrotor acting on Sinusoidal profile.

Figure 5.15 shows all the response of the different controllers on the trajectory of Gaussian filter with a 35 degree laser beam angle and Figure 5.16 shows the errors made by the control response. Based on these figures, classic controllers like PID would not be able to track difficult terrain profiles because of their inability to follow diverse or constantly changing terrains. Even though the LQI is one of the more advanced control techniques, it offers a poor performance when tracking its trajectory; the MPC controller, on the other hand, is not able

to precisely track the trajectory, but is able to minimize the error. Hence, we can see that for a quadrotor to be able to faithfully follow the terrain in GPS denied environments, advanced controllers offer a better performance than the classic control options.

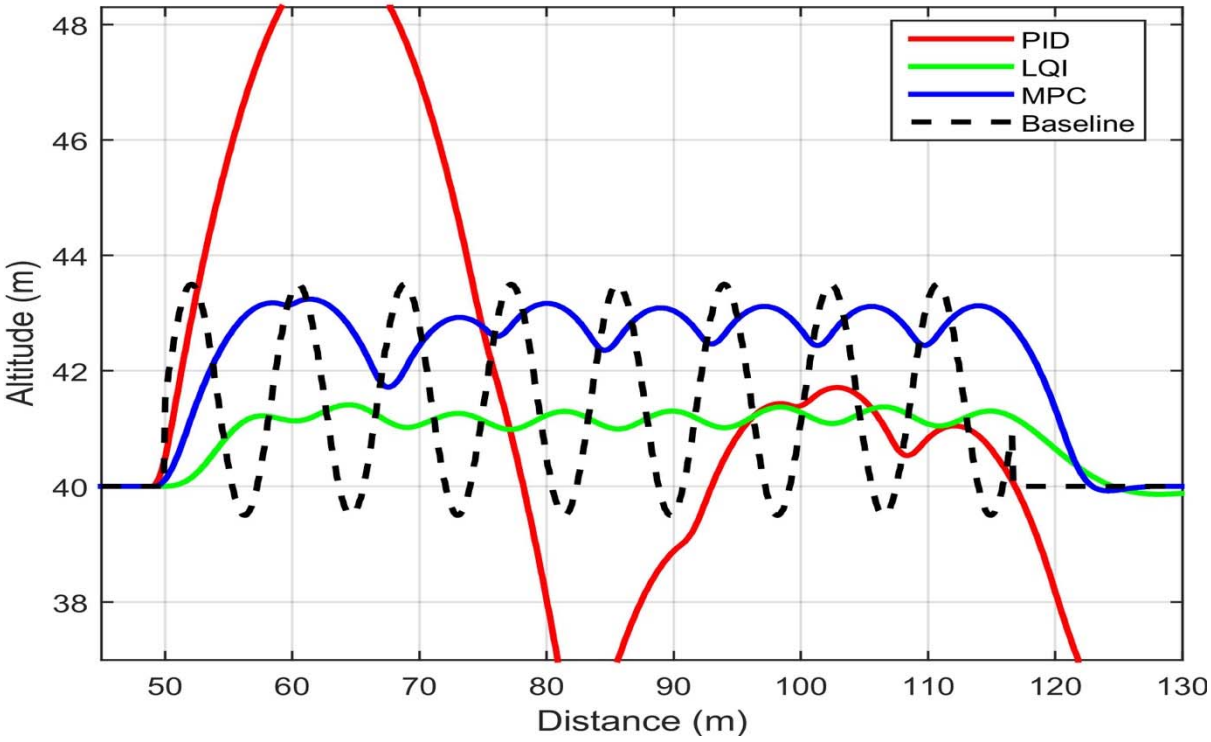

Figure 5.15: Control response of a 35 degree laser beam angle with Gaussian Filter as a trajectory planning method.

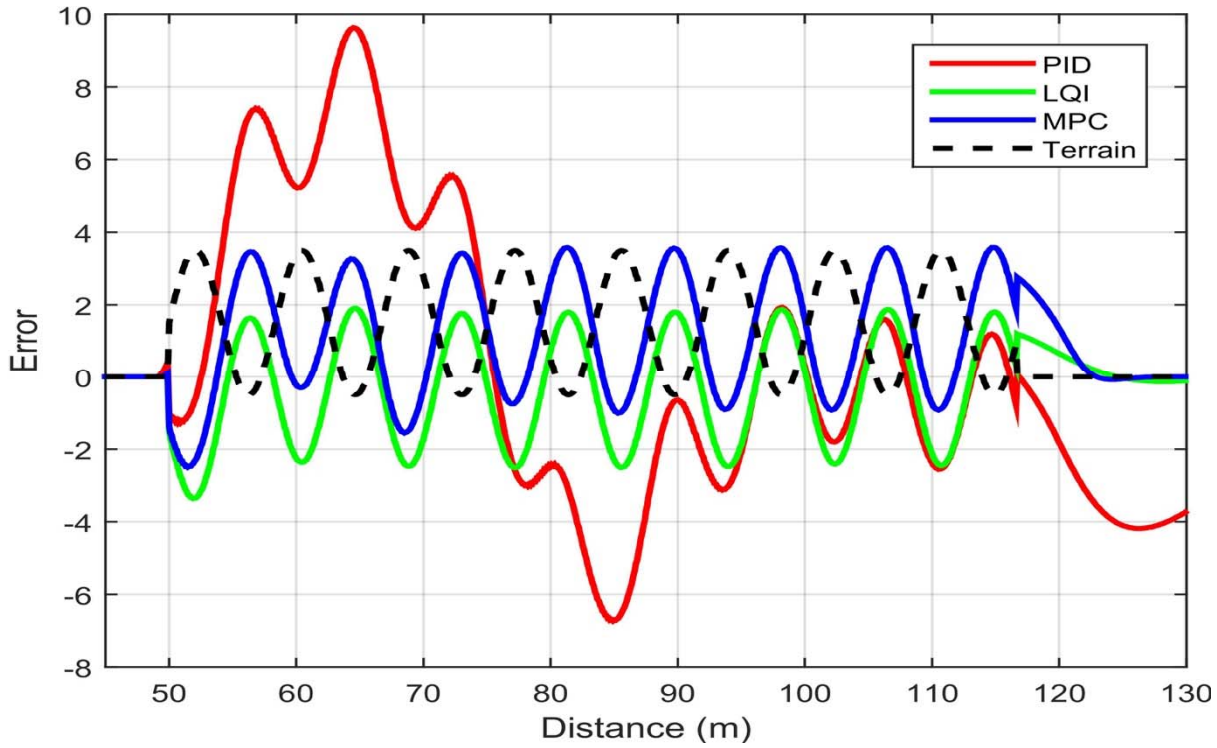

Figure 5.16: Error of the control response.

5.3.4 Case Study: Combined Profile 1 (Step and Sinusoidal profiles)

This case study presents a terrain profile which incorporates two different types of profiles. It is a profile that simulates both continuous and abrupt changes to the environment. From analyzing the preceding scenarios, we see that each terrain profile highlights specific trajectory-planning conditions which improve the performance of the quadrotor, but it cannot be assumed that the proposed solution of retrieving information from ahead of the quadrotor also works well for complex terrain profiles such as we see in this example. This case study, however, deduces that the performance of a quadrotor based on the trajectory of the proposed solution outperforms the performance of a quadrotor based on the traditional vertical sensor solution. Figure 5.17 shows the trajectories based on all the techniques, including the vertical sensor, and Table 5.4 presents the RMS of all trajectory planning algorithms acting on this combined profile. General observations of both the graphs and the table suggest that

while the relationship between decreasing errors in replicating the terrain is proportional to decreasing the angle of the laser beam, there is still a high chance of hitting protruding objects especially if the laser angle beam is really small.

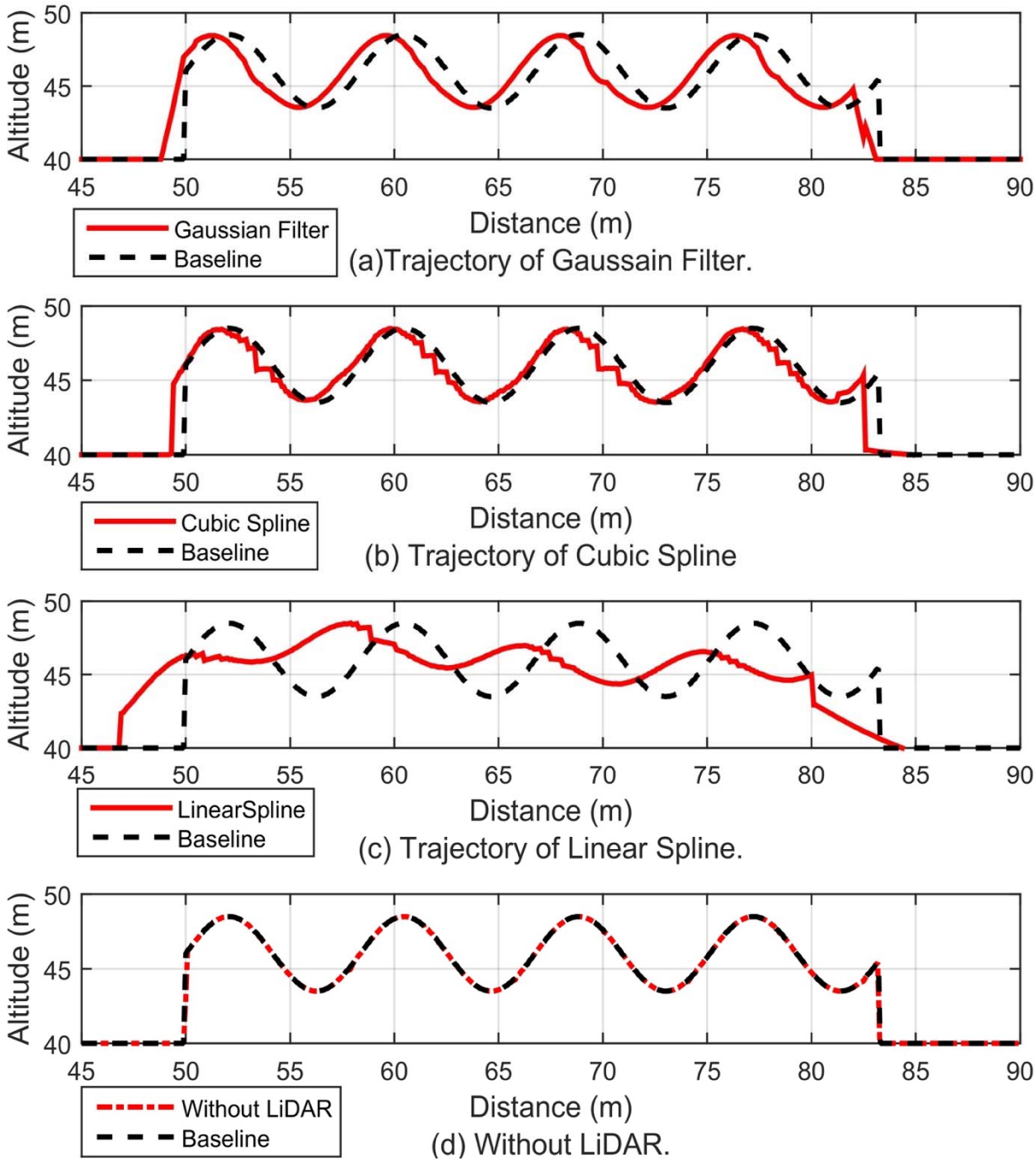

Figure 5.17: Trajectory of first combined profile in all techniques.

Table 5.4: Root mean square deviation of the different trajectory planning techniques of the first combined profile.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	57.1759	42.3530	21.8762	36.3171
45	57.8239	44.6021	21.7722	
35	59.7599	45.5961	21.6901	
25	58.8499	34.8899	21.2150	
15	59.2101	34.2445	22.6338	

Figure 5.18 and Figure 5.19 show the trajectory of Gaussian filter and cubic spline compared with the trajectory based on a vertical sensor. The trajectory of Gaussian filter and cubic spline help the quadrotor to follow the terrain while maintaining its altitude, whereas the trajectory based on a vertical sensor is almost impossible to track as a result of its lack of control abilities. However, there is a noteworthy point on these two figures: Neither trajectory is able to reliably maintain the altitude of the quadrotor, especially not the cubic spline's, because of lack of data or missing information. Figure 5.18 is an example of generating a trajectory based on very few data while Figure 5.19 is an example of generating a trajectory based on missing data. These trajectories are difficult to follow if there is a constraint on the input or energy of the system. Hence, even though the proposed solution helps the quadrotor to get information about what is ahead, there is a chance the quadrotor will not be able to follow the terrain smoothly at a fixed distance. That means that the traditional method of navigation which is based on a vertical sensor would also not work for this type of profile if the goal of using the quadrotor is to follow the terrain at a fixed altitude. Figure 5.20 shows the motion planning of all trajectory planning methods acting on this first combined profile. Figure 5.21 shows the control responses that are based on the Gaussian filter trajectory with a 25 degree laser beam angle and Figure 5.22 shows the errors.

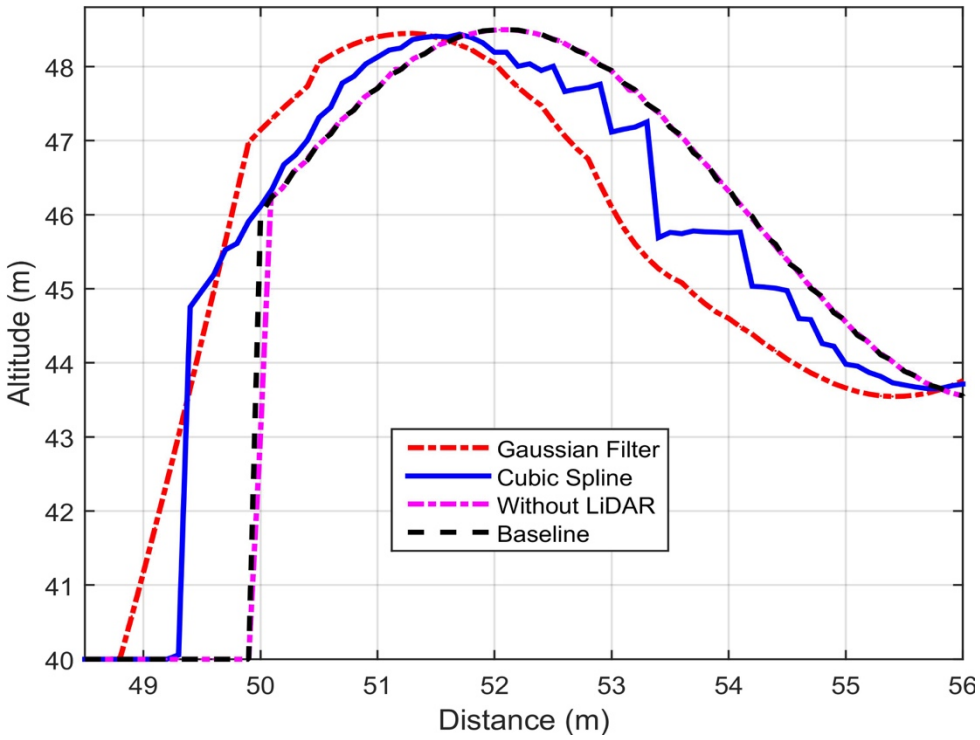

Figure 5.18: Trajectory of first combined profile in an ascending case.

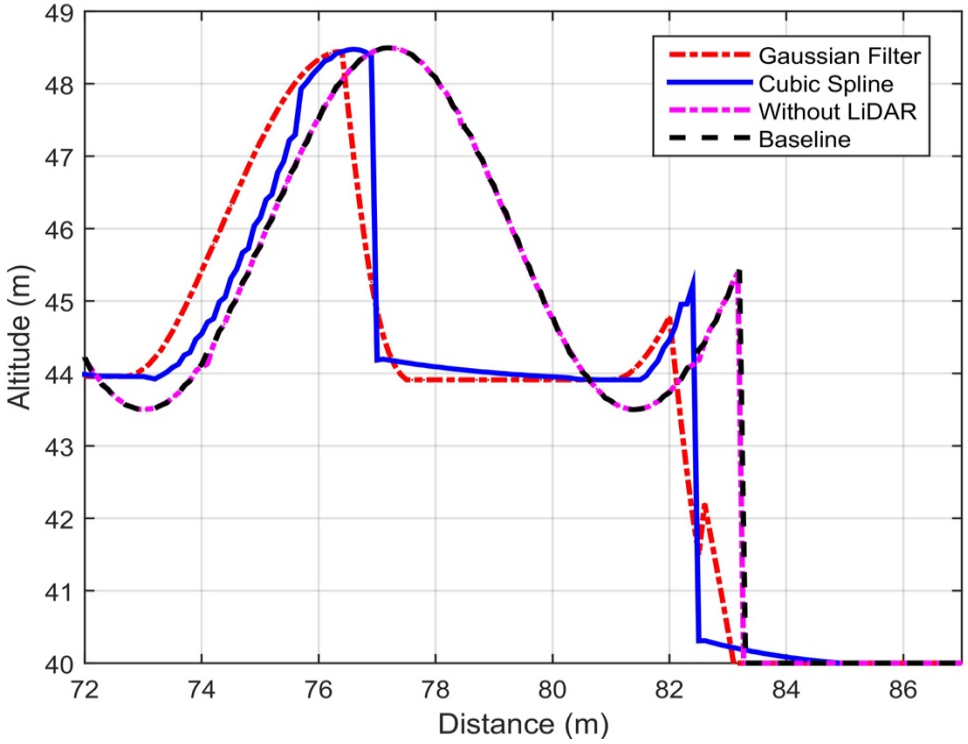

Figure 5.19: Trajectory of first combined profile in a descending case.

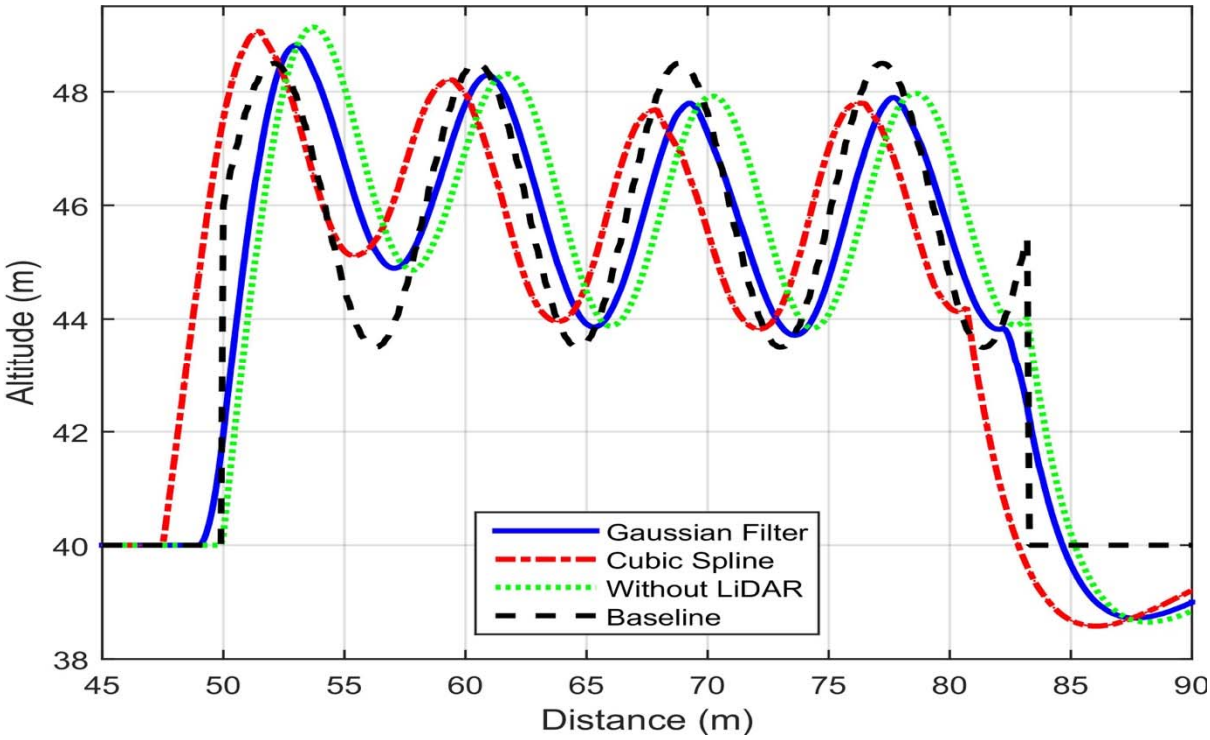

Figure 5.20: Motion planning of a quadrotor acting on first combined profile.

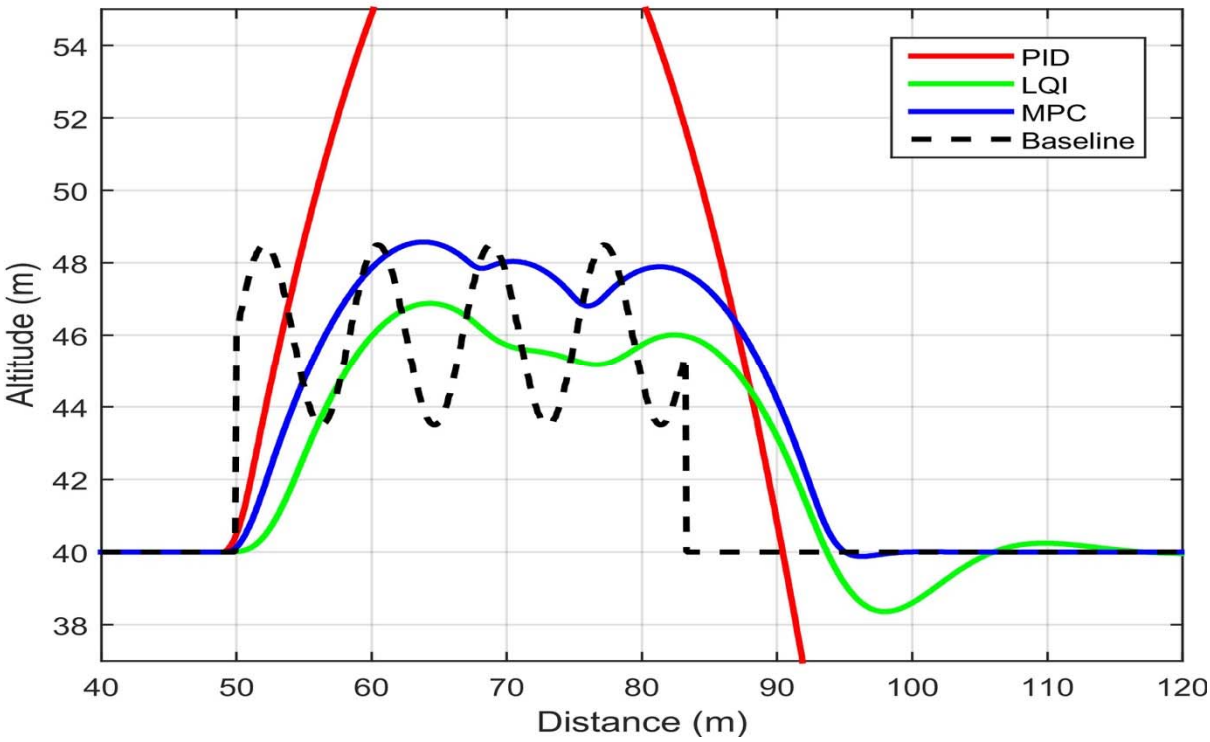

Figure 5.21: Control response of a 25 degree angle with Gaussian Filter as the trajectory planning method.

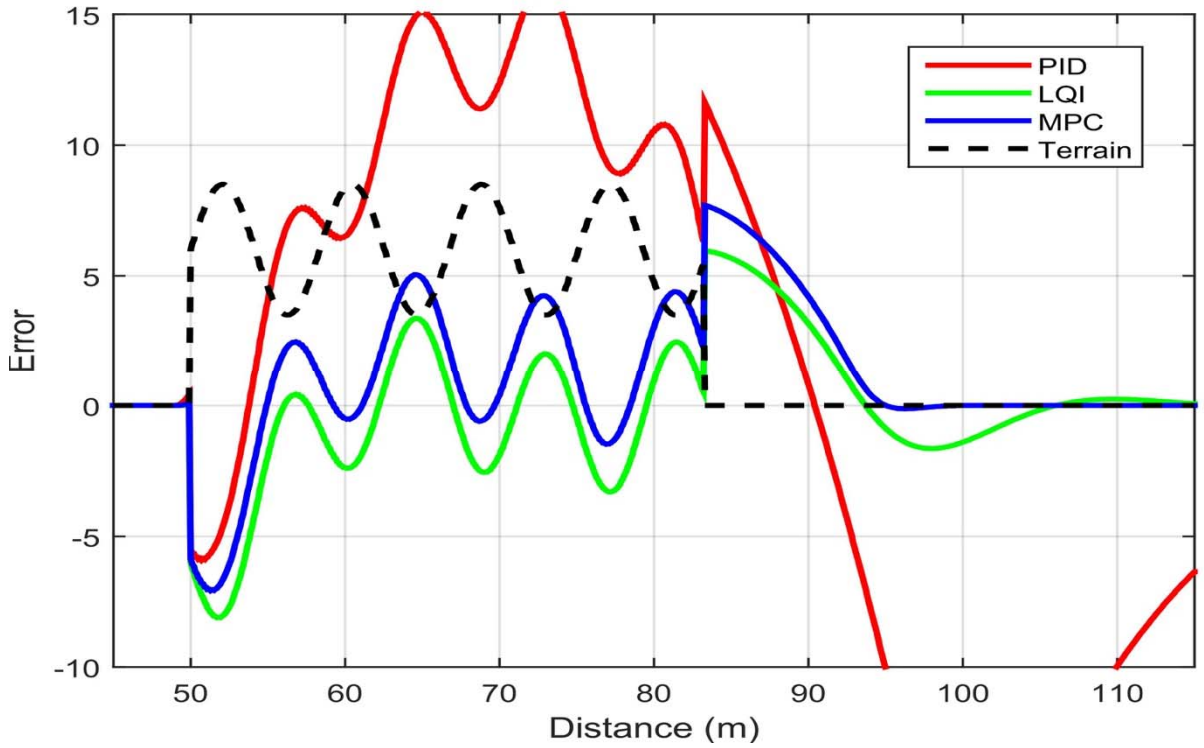

Figure 5.22: Error of the control response.

5.3.5 Case Study: Combined Profile 2 (Double Step profile)

This case study is another example of a complex environment with different types of terrain profiles. The advantage of using this profile as a simulation environment is that it is able to test the effectiveness of the proposed solution using different trajectory planning methods where there is missing sections of the terrain. This terrain profile consists of step profiles with a small gap between them. The difficulty in following this terrain profile is that the LiDAR sensor cannot detect the region behind the first step in the profile as it would instead immediately detect the second step behind the first. Because of the requirement to maintain the altitude, the trajectory planning should generate a trajectory that fulfills this condition even though there is missing data. Figure 5.23 shows all the trajectories. This figure proves that the trajectory based on a vertical sensor is able to faithfully replicate the terrain profile, but is ultimately impossible for a quadrotor to follow while maintaining a certain altitude

from the terrain. In contrast, the trajectories based on the proposed solution are more suitable for a quadrotor to track as they are generated with the quadrotor's abilities in mind while still avoiding collisions with any of the sudden protrusions of the step terrain profiles. Table 5.5 presents the RMS of all trajectories and demonstrates that the trajectory based on the proposed solution outperforms the others.

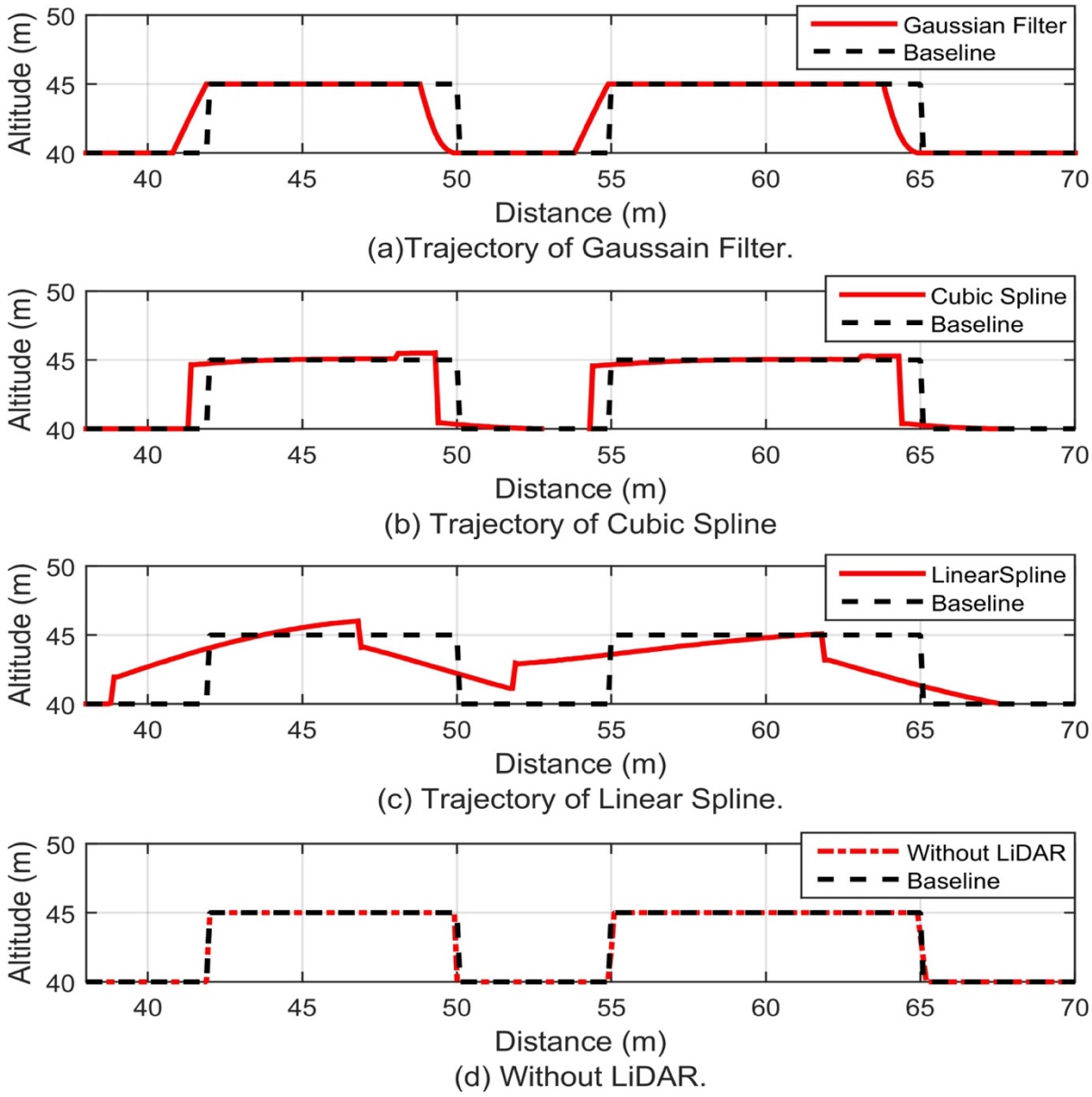

Figure 5.23: Trajectory of second combined profile in all techniques.

Table 5.5: Root mean square deviation of the different trajectory planning techniques of second combined profile.

Angle (degrees)	Linear Spline	Cubic Spline	Gaussian Filter	Vertical Sensor
50	42.8812	38.1700	22.1844	31.3568
45	40.9079	29.6529	22.3745	
35	40.8563	29.9847	22.5932	
25	40.4553	29.9328	22.5791	
15	40.7066	29.8339	22.4062	

Figure 5.24 shows the trajectories that are generated with no information about the gap between the two steps except for what is gathered by the traditional method of sensing. This figure indicates that the trajectory based on the vertical sensor would lead the quadrotor to crash as it would not be able to avoid the second vertical object in the step terrain profile. Contrarily, the trajectories based on the proposed solution take this gap into account when they generate the trajectory as shown in Figure 5.24. This does not mean there is no chance of hitting the sharp object, but there is less chance than when following the trajectory based on the vertical sensor. Figure 5.25 shows the motion planning of all methods. This figure indicates that the motion planning based on the cubic spline trajectory has the least chance of hitting the sharp object. Figure 5.26 shows the control responses that are based on the Gaussian filter trajectory with a 45 degree laser beam angle and Figure 5.27 shows the errors.

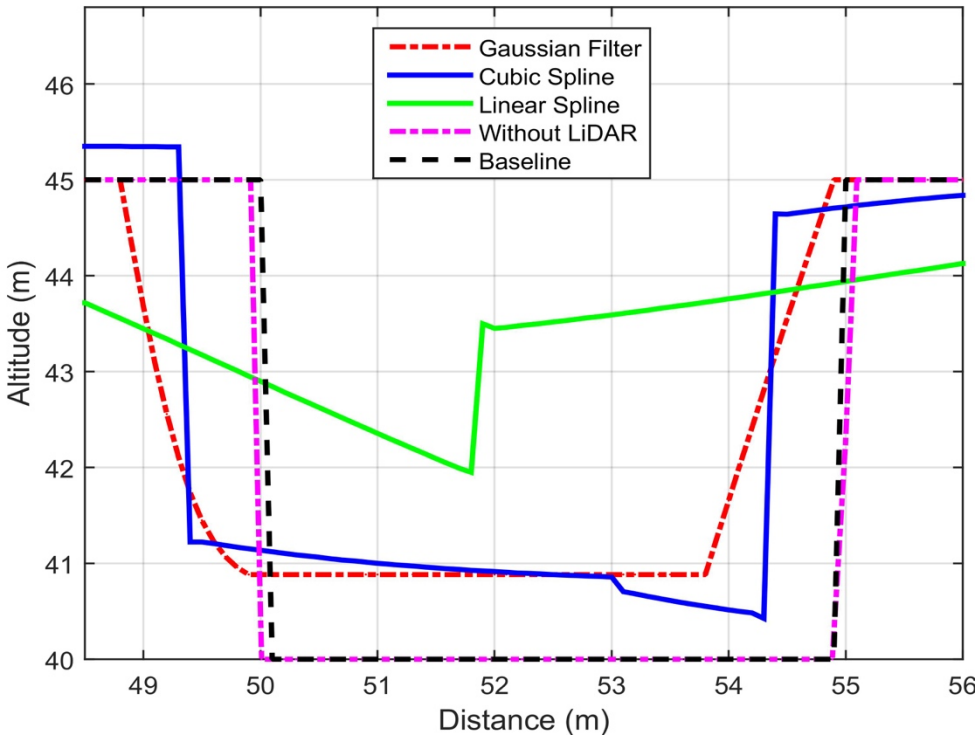

Figure 5.24: Trajectory of first combined profile in ascending and descending cases.

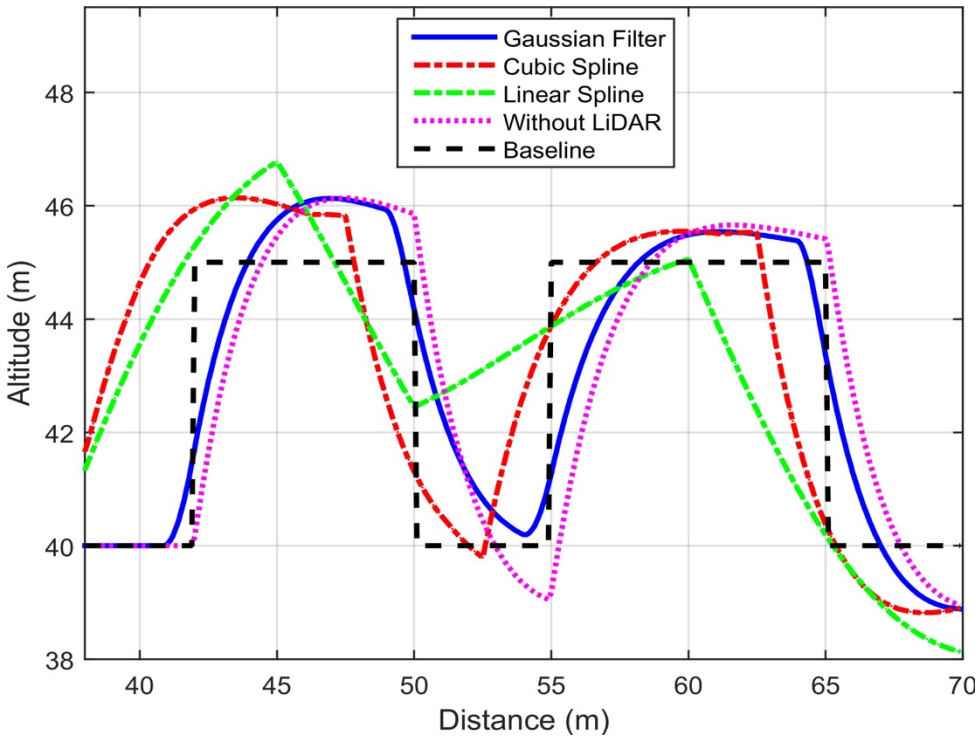

Figure 5.25: Motion planning of a quadrotor acting on second combined profile.

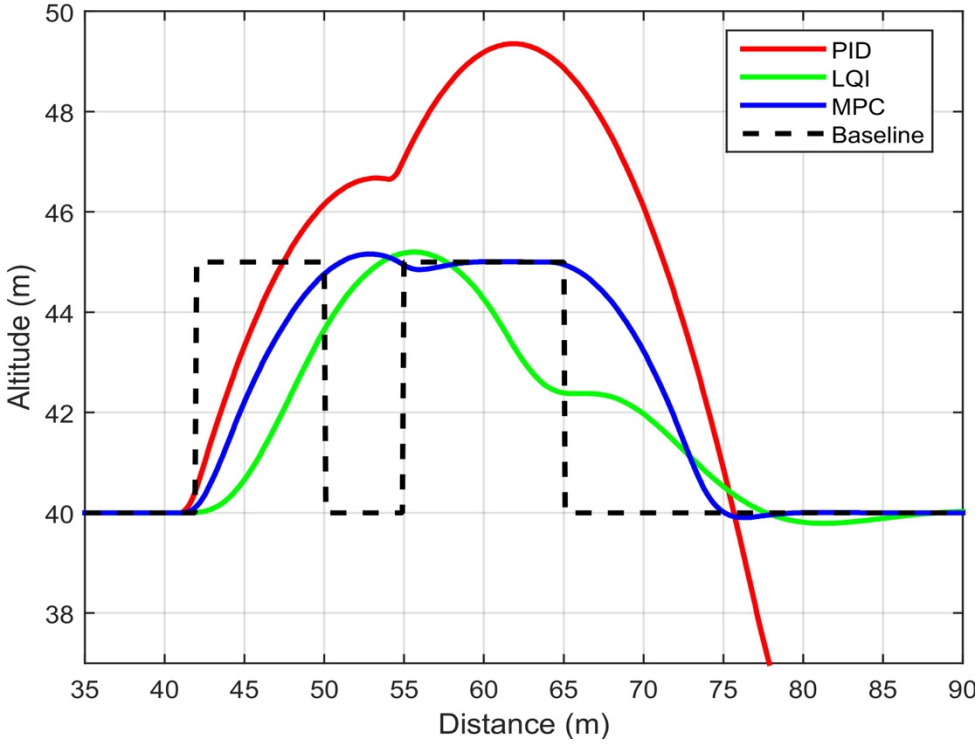

Figure 5.26: Control response of 45 degree angle beam with Gaussian Filter as trajectory planning algorithm.

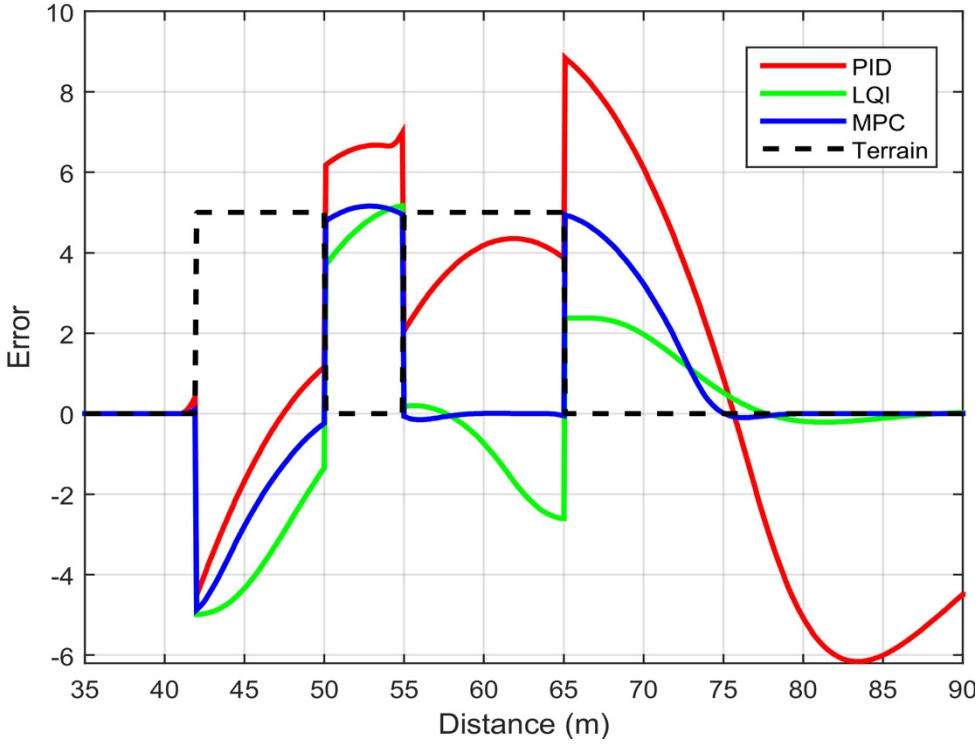

Figure 5.27: Error of the control response.

5.4 Summary

In this section, we would like to highlight the most important outcomes or necessary points when applying this new framework in real applications:

- The angle of the laser beam depends on the type of terrain profile. In case of gradual terrain changes, it is preferable to have a laser beam set at a larger angle because it will retrieve reliable and detailed information about the terrain which will help the motion planning algorithms to generate a smooth path. On the other hand, if the terrain exhibits dramatic changes, especially in terms of height, it is preferable to have a smaller laser beam angle as the LiDAR sensor will be able to retrieve more reliable data about the terrain.
- Utilizing Gaussian filter as a motion planning technique is the best option in terms of replicating the terrain, but it can lead the quadrotor to crash if there is missing information, especially in terms of a sharp descent. Cubic spline, on the other hand, is better for accurately ascending or descending a sharply raised object, but sometimes fails at keeping the quadrotor's altitude at a fixed distance from the terrain.
- This new navigational method works perfectly with advanced control techniques in terms of minimizing energy consumption. The best control algorithm according to the software platform is model predictive control (MPC). It helps to reduce energy consumption while taking into account the quadrotor's capabilities. The only concern about MPC is its long computational process.

Chapter 6: Conclusions and Recommendations

6.1 Summary

The utility of UAVs in civilian applications is constantly increasing, and this then requires increased flexibility from UAV navigation systems. One modern requirement is for UAVs to be able to fly over obstacles or infrastructure systems including pipes, bridges and superstructures like buildings. Maintaining a certain distance between the UAV and the obstacle it is flying over require the creation of a flight trajectory for the UAV controller which has visibility of the terrain ahead of the UAV. The main contribution of this thesis was the introduction of a new method of navigation for quadrotors that allows them to maintain altitude at a consistent distance from the terrain underneath and at the same time maximize flight time by avoiding the need for sudden energy-sapping corrections. The proposed approach involves measuring the distance between the UAV and the terrain using a rangefinder installed at a given angle to create an optimal flight trajectory for the UAV, subject to measurement uncertainty ahead of time. An optimal controller is then used to follow the trajectory, subject to input constraints. The efficacy of the proposed method was then verified through the use of simulation in presence of measurement noise and input constraints. The conclusions of this work will be as follows:

- Assisting an unmanned quadrotor to follow unknown terrains in a GPS denied environment was the core objective of this work. In Chapter 1, the relevant literature was introduced, through which we concluded that previous work relied on *a prior* knowledge of the terrain uploaded into the quadrotor. From the existing quadrotor frameworks within the literature, we concluded that while it is not impossible for a quadrotor using these techniques to follow certain terrain, the existing technology is certainly not sufficient for all environments.

- Utilizing a LiDAR sensor as an extra sensor for feeding information back to the quadrotor about undulations beneath and ahead is an essential component of the new navigational method because: (i) it outperforms the other sensors in terms of both providing accurate measurements and its lightweight form; (ii) it improves the reliability of the sensing system especially when the sensors are positioned at an angle. In Chapter 2, the new framework was introduced after giving a brief introduction about the dynamic system and system architecture of the quadrotor.

- Forwarding the terrain data to a motion planning technique is a key process in the improvement of the quadrotor's navigation system. Ultimately, this approach concludes that the effectiveness of this new navigational method depends on the goal for which the quadrotor will be used or, in other words, the purpose and general terrain of the mission should be defined before the filtering window values or laser angles are set. Chapter 3 tackled this issue by introducing three different motion planning algorithms: linear spline, cubic alpine, and a Gaussian filter. A case study of step terrain profiles used with the proposed method validated the concept.

- Taking best advantage of the most suitable trajectory planning algorithm with the proposed solution requires the use of an advanced or 'optimal' controller. One of the reasons the controller's role is so crucial to the overall effectiveness of the navigational system is that the quadrotor does not have the ability to generate negative thrust. It must therefore take advantage of an optimal controller's ability to plan and accurately follow a trajectory in order to minimize the impact of its limitations. In Chapter 4, three different control algorithms were introduced. A case

study of a step terrain profile with the proposed solution was introduced to compare the responses of three controllers.

- Designing a software platform to simulate a quadrotor's capacity to follow certain terrain was the second core objective of this work. This platform not only helped us to validate the new navigational method, but also to visualize the quadrotor's reactions in different scenarios. In Chapter 5, the validation of the new method used within different terrain profiles was introduced. All mentioned experiments were carried out in the software platform. This platform also has the potential to be an educational tool for engineers who specialize in control systems, motion planning and robotics.

6.2 Future Work

Future work in this field is suggested as follows:

- The implementation of the proposed method in a quadrotor can be pursued in the future.

- The new navigational framework includes only one forward facing LiDAR sensor at the base of the quadrotor. Another LiDAR sensor could also be added to the bottom of the quadrotor pointing in the opposite direction. There are two advantages of this addition. Firstly, it could be used to verify the estimation of the first LiDAR sensor using a redundant technique to reduce the uncertainty in the data. Secondly, it gives the quadrotor the flexibility to take off or fly either forwards or backwards since there would now be two sensors, one at the front and one at the rear.

- The control system integrated into this new navigational method can be expanded to cover the nonlinearity of the system. The current analysis of the control system is

based on the linearization of a hovering status. Further research on this topic may include the design of a full control system which considers all quadrotor states. In this work, we have considered only the quadrotor state in the Z direction with the pitch angle and velocity of the quadrotor in the X direction.

- The quadrotor's performance in following the terrain using this new navigational method may be improved through the use of a deep learning technique to detect the features of a terrain and profile if fully based on a limited number measurements. This new quadrotor framework could also provide an excellent research platform for deep learning.

Bibliography

[1] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 477-483: IEEE.

[2] D. W. Mellinger, "Trajectory generation and control for quadrotors," Doctor of Philosophy (PhD) Dissertation, Mechanical Engineering & Applied Mechanics, University of Pennsylvania, 2012.

[3] H. Bolandi, M. Rezaei, R. Mohsenipour, H. Nemati, and S. M. Smailzadeh, "Attitude control of a quadrotor with optimized PID controller," *Intellignet Control and Automation,* vol. 4, pp. 335-342, 2013.

[4] H. Bouadi, M. Bouchoucha, and M. Tadjine, "Sliding mode control based on backstepping approach for an UAV type-quadrotor," *World Academy of Science, Engineering and Technology,* vol. 26, no. 5, pp. 22-27, 2007.

[5] L. D. Minh and C. Ha, "Modeling and control of quadrotor MAV using vision-based measurement," in *Strategic Technology (IFOST), 2010 International Forum on*, 2010, pp. 70-75: IEEE.

[6] R. He *et al.*, "On the design and use of a micro air vehicle to track and avoid adversaries," *The International Journal of Robotics Research,* vol. 29, no. 5, pp. 529-546, 2010.

[7] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 1649-1654: IEEE.

[8] S. Shen, "Autonomous navigation in complex indoor and outdoor environments with micro aerial vehicles," Doctor of Philosophy (PhD) Dissertation, Electrical and Systems Engineering, University of Pennsylvania, 2014.

[9] H. Najjaran, "Uncertainty Management Method for a Terrain Scanning Robot," Doctor of Philosophy (PhD) Dissertation, Mechanical and Industrial Engineering, University of Toronto 2005.

[10] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 7410.

[11] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics,* vol. 29, no. 2, pp. 315-378, 2012.

[12] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2002, vol. 1, pp. 72-77: IEEE.

[13] E. Altug, J. P. Ostrowski, and C. J. Taylor, "Control of a quadrotor helicopter using dual camera visual feedback," *The International Journal of Robotics Research,* vol. 24, no. 5, pp. 329-342, 2005.

[14] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 2004, vol. 5, pp. 4393-4398: IEEE.

[15] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 9-15: IEEE.

[16] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles,* vol. 1, no. 4, pp. 217-228, 2009.

[17] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics,* vol. 28, no. 1, pp. 90-100, 2012.

[18] I. Dryanovski, W. Morris, and J. Xiao, "An open-source pose estimation system for micro-air vehicles," in *Robotics and automation (ICRA), 2011 IEEE international conference on*, 2011, pp. 4449-4454: IEEE.

[19] A. Kushleyev, B. MacAllister, and M. Likhachev, "Planning for landing site selection in the aerial supply delivery," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 1146-1153: IEEE.

[20] A. Bry, A. Bachrach, and N. Roy, "State estimation for aggressive flight in GPS-denied environments using onboard sensing," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1-8: IEEE.

[21] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 957-964: IEEE.

[22] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa, "Stereo vision based indoor/outdoor navigation for flying robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3955-3962: IEEE.

[23] S. Weiss *et al.*, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *Journal of Field Robotics,* vol. 30, no. 5, pp. 803-831, 2013.

[24] F. Fraundorfer *et al.*, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 4557-4564: IEEE.

[25] S. M. LaVALLE. (2011) Motion Planning part I: The Eessentials. *IEEE Robotics & Automation Magazine* 79 - 89.

[26] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991, p. 672.

[27] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[28] N. Dadkhah and B. Mettler, "Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance," *Journal of Intelligent & Robotic Systems,* vol. 65, no. 1-4, pp. 233-246, 2012.

[29] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, 2009, pp. 65-100: Springer.

[30] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica,* vol. 33, no. 03, pp. 463-497, 2015.

[31] A. Mohandes, "Motion planning based on uncertain robot states in dynamic environments: a receding horizon control approach," University of British Columbia, 2014.

[32] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing," in *Robotics and automation (ICRA), 2011 IEEE international conference on*, 2011, pp. 2472-2477: IEEE.

[33] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New York, 2006.

[34] S. M. La Valle. (2011) Motion Planning part II: Wild Frontiers

. *IEEE Robotics & Automation Magazine*. 108-118.

[35] B. Erginer and E. Altug, "Modeling and PD control of a quadrotor VTOL vehicle," in *Intelligent Vehicles Symposium, 2007 IEEE*, 2007, pp. 894-899: IEEE.

[36] H. Voos, "Nonlinear control of a quadrotor micro-UAV using feedback-linearization," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, 2009, pp. 1-6: IEEE.

[37] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," Master Thesis, Department of Automatic Control, Lund University 2008.

[38] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles," *IEEE Robotics and Automation magazine,* vol. 20, no. 32, 2012.

[39] S. Tang and V. Kumar, "Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 2216-2222: IEEE.

[40] M. K. Joyo, D. Hazry, S. F. Ahmed, M. H. Tanveer, F. A. Warsi, and A. Hussain, "Altitude and horizontal motion control of quadrotor UAV in the presence of air

turbulence," in *Systems, Process & Control (ICSPC), 2013 IEEE Conference on*, 2013, pp. 16-20: IEEE.

[41] A. A. Ghaffar and T. Richardson, "Model reference adaptive control and LQR control for quadrotor with parametric uncertainties," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering,* vol. 9, no. 2, 2015.

[42] M. Belkheiri, A. Rabhi, A. El Hajjaji, and C. Pegard, "Different linearization control techniques for a quadrotor system," in *Communications, Computing and Control Applications (CCCA), 2012 2nd International Conference on*, 2012, pp. 1-6: IEEE.

[43] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, "Robust adaptive control design for quadcopter payload add and drop applications," in *Control Conference (CCC), 2015 34th Chinese*, 2015, pp. 3252-3257: IEEE.

[44] D. Lee, H. J. Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *International Journal of control, Automation and systems,* vol. 7, no. 3, pp. 419-428, 2009.

[45] A. A. Mian and D.-b. Wang, "Dynamic modeling and nonlinear control strategy for an underactuated quad rotor rotorcraft," *Journal of Zhejiang University-Science A,* vol. 9, no. 4, pp. 539-545, 2008.

[46] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Control and Automation, 2008 16th Mediterranean Conference on*, 2008, pp. 1258-1263: Ieee.

[47] J. Z. Kolter and A. Y. Ng, "Task-space trajectories via cubic spline optimization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 1675-1682: IEEE.

[48] B. Frank, C. Stachniss, N. Abdo, and W. Burgard, "Using Gaussian Process Regression for Efficient Motion Planning in Environments with Deformable Objects," in *Automated Action Planning for Autonomous Mobile Robots*, 2011.

[49] S. K. Gan, K. Yang, and S. Sukkarieh, "3d path planning for a rotary wing uav using a gaussian process occupancy map," in *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[50] J. Li and Y. Li, "Dynamic analysis and PID control for a quadrotor," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, 2011, pp. 573-578: IEEE.

[51] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," in *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*, 2013, pp. 1-6: IEEE.

[52] H. Jafari, M. Zareh, J. Roshanian, and A. Nikkhah, "An optimal guidance law applied to quadrotor using LQR method," *Transactions of the Japan Society for Aeronautical and Space Sciences,* vol. 53, no. 179, pp. 32-39, 2010.

[53] C. Liu, H. Lu, and W.-H. Chen, "An explicit MPC for quadrotor trajectory tracking," in *Control Conference (CCC), 2015 34th Chinese*, 2015, pp. 4055-4060: IEEE.

[54] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained control and estimation: an optimisation approach*. Springer Science & Business Media, 2006.

Appendices

Appendix A Software Platform

This section presents the features of the software platform. It aims to give a brief overview for every control button option in the main Graphic User Interface (GUI) screen. The main control option in the software platform are:

- Terrain Profile
- Motion Planning
- Control
- Filter

We will talk briefly about those features after presenting the GUI and the animation on the GUI.

A.1 GUI

This GUI is built based on Matlab language as is mentioned in Chapter 5. There are two methods of building GUI in Matlab: writing a code or inserting blocks. This GUI is based on writing a code method which allows it to be adjusted to easily change the parameters of the system. The quadrotor animation is represented in two figures: The left figure presents the quadrotor trajectory based on the motion planning methods selected by the user, and also the control responses acting on that trajectory. The right figure simulates the motion of a quadrotor which has a LiDAR sensor at the base to gather information and act on the terrain profile. The quadrotor is represented by a point in the animation and it travels from left to right in the display screen of the graphic user interface. The light of the LiDAR sensor is represented by a line from the quadrotor, a point in the animation, to the terrain profile. The

length of the line is based on the calculation of the angle of the laser beam and the elevation of the quadrotor. In the case where the quadrotor dramatically ascends, the line of the LiDAR sensor in the animation will not reach the terrain profile. In addition, underneath and next to these two figures are five scrolls which help the user to adjust the value of the angle of the laser beam, quadrotor and terrain speed, noise, and elevation of the quadrotor.Figure A.1 shows the GUI when it is running.

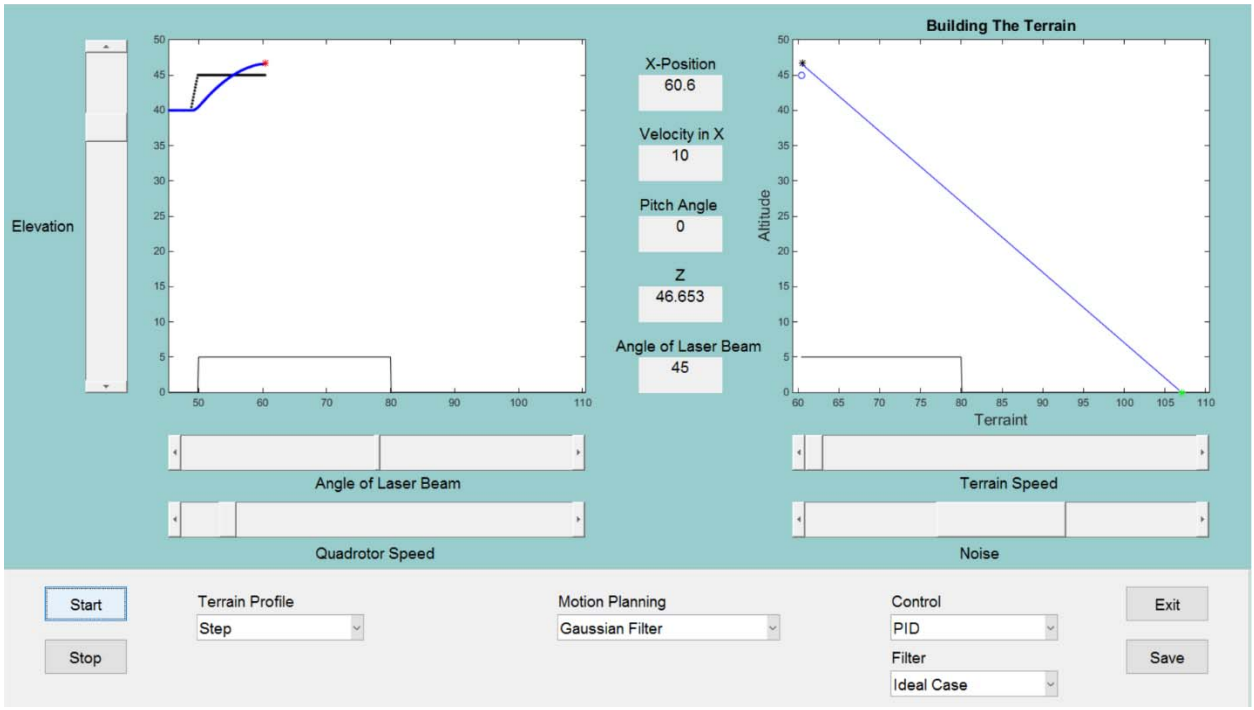

Figure A.1: Main outlook of GUI when it is building the trajectory of a quadrotor.

A.2 Terrain Profiles

This software platform generates several theoretical terrain profiles including: step, trapezoid, sinusoidal ramp, and two mixed profiles as shown in Figure A.**2**. The terrain profile travels from right to left in the display screen of the GUI. The speed of the simulated terrain can also be adjusted by scrolling the terrain speed control underneath the right figure. The parameters of each terrain profile can be adjusted in the code.

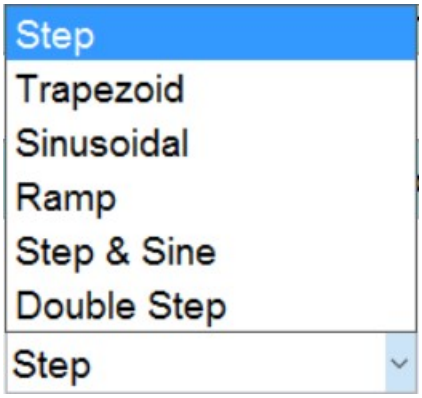

Figure A.2: Dropdown menu of terrain profiles.

A.3 Motion Planning

This software platform obtains the desired trajectory planning by utilizing several motion planning techniques. All methods are represented in

Figure A.3. The dropdown menu also has an extra option (no path) which presents the traditional method. The parameters of each method, especially that of the Gaussian filter, can be adjusted in the code. The window size of the Gaussian filter is based on the standard deviation or 'sigma'. As explained in Chapter 3, the Gaussian filter has different sigma when the quadrotor is ascending or descending. In the code, the sigma can be adjusted by activating the sigma down control if the user wants to minimize or maximize the window size when the quadrotor is descending, and vice versa.

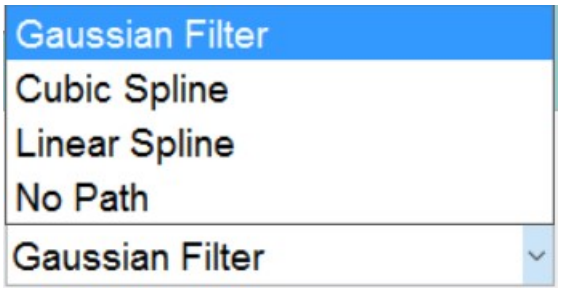

Figure A.3: Dropdown menu of motion planning algorithms.

A.4 Control Algorithms

This software platform provides one classic control technique and two optimal control techniques as shown in Figure A.4. The values of gains are adjustable. The full steps relating to the control methods are represented in the code and can be found in Appendix B.

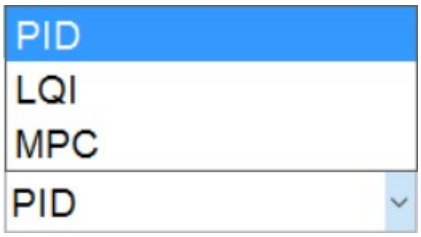

Figure A.4: Dropdown menu of control algorithms.

A.5 Filters

To create a realistic quadrotor animation, uncertainty has also been considered. Figure A.5 shows the filter options.The ideal case would be where no random numbers are added into the LiDAR data. The second option is a case where there are random numbers added into the environment data without using a technique to reduce the effectiveness of the uncertainty on the quadrotor's performance. The final technique is similar to the latter option, but also uses a low pass filter technique to reduce the effectiveness of the uncertainty. The parameters of each terrain profile can be adjusted in the code.

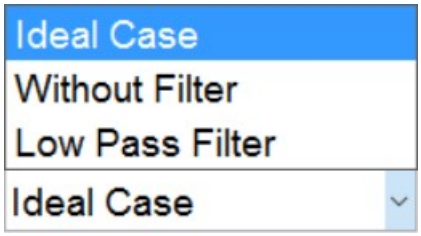

Figure A.5: Dropdown menu of filters.

Appendix B Simulation Code

B.1 Code for Generating GUI

```
function GUI_sim20_AppendixA
% Clean Code
clear; clc
close all
global GUIProp Xp PathData ZZ XX Zd U;
%% Initialization of GUI:
GUIProp.EnableRun   = 0;  % Do not start simulation
GUIProp.WaitPeriod  = 0.05;
GUIProp.NoisePower  = 25; %
GUIProp.WindowMeter = 50; % meter
GUIProp.TerrainMode = 1;  % Step
GUIProp.ControlMode = 1;  % PID
GUIProp.FilterMode  = 1;  % Ideal Case
GUIProp.PathMode    = 1;  % Gaussian Filter
%% Initialization of the Main Concept of the Problem Statement:
PathData.LasserAngle   = 45*pi/180; % Radian
PathData.EleDes        = 40;        % meter
PathData.xdot_des      = 10;        % meter/sec
PathData.PointPerMeter = 10;        % meter
PathData.Terrain.Rep   = 5;         % number of the repetitions
%% GUI Screen:
%------------------- Check the window size -------------------------
% get screen information size and depth
ScreenSize   = get(0, 'ScreenSize' );
ScreenWidth  = ScreenSize(3);
ScreenHight  = ScreenSize(4);
% Divide the screen to tabs for organizing the GUI Screen
WidthTabNo   = 30;
HightTabNo   = 20;
WidthTab     = ScreenWidth/WidthTabNo;
HightTab     = ScreenHight/HightTabNo;
MainWindowPos = [0,0,ScreenWidth,ScreenHight];
%% GUI Structure:
MainWindow = figure('Color',[0.6 0.8 0.8], ...
    'Name','Fixed Altitude', ...
    'NumberTitle','off',...
    'Units','pixels',...
    'Position',MainWindowPos,...
    'Resize', 'off',...
    'Tag','TMainWindow');
bg = uibuttongroup('Visible','on',...
    'Units','pixels', ...
    'Position',[0*WidthTab 0*HightTab 30*WidthTab 4*HightTab],...
    'SelectionChangedFcn',@bselection);
Start_button = uicontrol('Parent',MainWindow, ...
    'Style','Pushbutton',...
    'Max',1,...
    'Min',0,...
    'Units','pixels', ...
    'Position',[1*WidthTab 2.5*HightTab 2*WidthTab 1*HightTab], ...
    'String','Start', ...
    'TooltipString','Start the simulation',...          % hint for user
    'Callback',@StartButton,...
    'Fontsize',14,...
    'Tag','TStrat');
Stop_button = uicontrol('Parent',MainWindow, ...
```

```
    'Style','Pushbutton',...
    'Max',1,...
    'Min',0,...
    'Units','pixels', ...
    'Position',[1*WidthTab 1*HightTab 2*WidthTab 1*HightTab], ...
    'String','Stop', ...
    'TooltipString','Stop the simulation',...          % hint for user
    'Callback',@StopButton,...
    'Fontsize',14,...
    'Tag','TStop');
Close_button = uicontrol('Parent',MainWindow, ...
    'Style','Pushbutton',...
    'Max',1,...
    'Min',0,...
    'Units','pixels', ...
    'Position',[27*WidthTab 2.5*HightTab 2*WidthTab 1*HightTab], ...
    'String','Exit', ...                               % Changing the string from
close to exist
    'TooltipString','Exit the simulation',...          % hint for user
    'Callback',@CloseButton,...
    'Fontsize',14,...
    'Tag','TClose');
Save_button = uicontrol('Parent',MainWindow, ...
    'Style','Pushbutton',...
    'Max',1,...
    'Min',0,...
    'Units','pixels', ...
    'Position',[27*WidthTab 1*HightTab 2*WidthTab 1*HightTab], ...
    'String','Save', ...
    'TooltipString','Save the simulation',...         % hint for user
    'Callback',@SaveButton,...
    'Fontsize',14,...
    'Tag','TClose');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[0.2*WidthTab 12*HightTab 2*WidthTab 2*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Elevation', ...
    'Fontsize',14,...
    'Tag','StaticText1');
EleDes_scroll= uicontrol('Parent',MainWindow, ...
    'Style','slider',...
    'Position',[2*WidthTab 9*HightTab 1*WidthTab 10*HightTab], ...
    'Min',0,...
    'Max',50,...
    'Value', PathData.EleDes,...
    'SliderStep',[0.1 0.1],...
    'TooltipString','Desired Altitude for the Quadrotor',...
    'Callback',@EleDesSliding,...
    'Tag','AltitSlid');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[7.5*WidthTab 6.1*HightTab 4*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Angle of Laser Beam', ...
    'Fontsize',14,...
    'Tag','StaticText1');
LasserAngle_scroll= uicontrol('Parent',MainWindow, ...
    'Style','slider',...
    'Position',[4*WidthTab 6.8*HightTab 10*WidthTab 1*HightTab], ...
    'Min',0,...
```

```
    'Max',pi/2,...
    'SliderStep',[0.01 0.01],...
    'Value', PathData.LasserAngle,...
    'TooltipString','Desired Angle of the Laser Beam',...
    'Callback',@LasserAngleSliding,...
    'Tag','AngleSlid');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[7.5*WidthTab 4.1*HightTab 3*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Quadrotor Speed', ...
    'Fontsize',14,...
    'Tag','StaticText1');
xdot_des_scroll= uicontrol('Parent',MainWindow, ...
    'Style','slider',...
    'Position',[4*WidthTab 4.9*HightTab 10*WidthTab 1*HightTab], ...
    'Min',0,...
    'Max',100,...
    'SliderStep',[0.05 0.05],...
    'Value', PathData.xdot_des,...
    'TooltipString','Desired Speed of the Quadrotor',...
    'Callback',@xdot_desSliding,...
    'Tag','QSpeedSlid');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[23*WidthTab 6.1*HightTab 3*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Terrain Speed', ...
    'Fontsize',14,...
    'Tag','StaticText1');
WaitPeriod_scroll= uicontrol('Parent',MainWindow, ...
    'Style','slider',...
    'Position',[19*WidthTab 6.8*HightTab 10*WidthTab 1*HightTab], ...
    'Min',0.001,...
    'Max',0.1,...
    'SliderStep',[0.05 0.05],...
    'Value', GUIProp.WaitPeriod,...
    'TooltipString','Terrain Speed',...
    'Callback',@WaitPeriodSliding,...
    'Tag','SpeedSlid');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[23*WidthTab 4.1*HightTab 3*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Noise', ...
    'Fontsize',14,...
    'Tag','StaticText1');
% name of noise power
NoisePower_scroll= uicontrol('Parent',MainWindow, ...
    'Style','slider',...
    'Position',[19*WidthTab 4.9*HightTab 10*WidthTab 1*HightTab], ...
    'Min',0,...
    'Max',50,...
    'Value', GUIProp.NoisePower,...
    'SliderStep',[0.5 0.5],...
    'TooltipString','No. of pixels for filtering',...
    'Callback',@NoisePowerSliding,...
    'Tag','NoiseSlid');
uicontrol('Parent',MainWindow, ...
    'Units','points', ...
```

```
    'Position',[3.5*WidthTab 1.5*HightTab 3*WidthTab 1*HightTab],...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Terrain Profile', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Terrain_pop = uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'ListboxTop',0, ...
    'Position',[3.5*WidthTab 1*HightTab 3*WidthTab 1*HightTab], ...
    'String',['Step       ';
    'Trapezoid  ';
    'Sinusoidal ';
    'Ramp       ';
    'Step & Sine';
    'Double Step'], ...
    'Style','popupmenu', ...
    'Tag','TShape', ...
    'TooltipString','Terrain shape',...
    'Fontsize',14,...
    'Callback',@TerrainShaping);
uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'Position',[16*WidthTab 1.5*HightTab 3*WidthTab 1*HightTab],...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Control', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Control_pop = uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'ListboxTop',0, ...
    'Position',[16*WidthTab 1*HightTab 3*WidthTab 1*HightTab], ...
    'String',['PID                    ';
    'LQI                  ';
    'MPC                  '], ...
    'Style','popupmenu', ...
    'Tag','TShape', ...
    'TooltipString','Control Techniques',...
    'Fontsize',14,...
    'Callback',@ControlShaping);
uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'Position',[16*WidthTab 0.4*HightTab 3*WidthTab 0.9*HightTab],...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Filter', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Filter_pop = uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'ListboxTop',0, ...
    'Position',[16*WidthTab 0*HightTab 3*WidthTab 0.8*HightTab], ...
    'String',['Ideal Case         ';
    'Without Filter     ';
    'Low Pass Filter    ';], ...
    'Style','popupmenu', ...
    'Tag','TShape', ...
    'TooltipString','Filter Techniques',...
    'Fontsize',14,...
    'Callback',@FilterShaping);
uicontrol('Parent',MainWindow, ...
    'Units','points', ...
```

```
    'Position',[10*WidthTab 1.5*HightTab 3*WidthTab 1*HightTab],...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Motion Planning', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Method_pop = uicontrol('Parent',MainWindow, ...
    'Units','points', ...
    'ListboxTop',0, ...
    'Position',[10*WidthTab 1*HightTab 4*WidthTab 1*HightTab], ...
    'String',['Gaussian Filter    ';
    'Cubic Spline       ';
    'Linear Spline      ';
    'No Path            '], ...
    'Style','popupmenu', ...
    'Tag','TShape', ...
    'TooltipString','Path Planning Techniques',...
    'Fontsize',14,...
    'Callback',@PathShaping);
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[15.3*WidthTab 18*HightTab 2*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String','X-Position', ...
    'Fontsize',14,...
    'Tag','StaticText1');
X_edit=uicontrol('Parent',MainWindow, ...
    'Position',[15.3*WidthTab 17*HightTab 2*WidthTab 1*HightTab],...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String',999, ...
    'Fontsize',14,...
    'Tag','StaticText1');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[15.3*WidthTab 16*HightTab 2.5*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Velocity in X', ...
    'Fontsize',14,...
    'Tag','StaticText1');
X_dot_edit=uicontrol('Parent',MainWindow, ...
    'Position',[15.3*WidthTab 15*HightTab 2*WidthTab 1*HightTab],...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String',999, ...
    'Fontsize',14,...
    'Tag','StaticText1');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[15.3*WidthTab 14*HightTab 2*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','left',...
    'Style','text', ...
    'String','Pitch Angle', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Pitch_edit=uicontrol('Parent',MainWindow, ...
    'Position',[15.3*WidthTab 13*HightTab 2*WidthTab 1*HightTab],...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String',0, ...
    'Fontsize',14,...
```

```
    'Tag','StaticText1');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[15.3*WidthTab 12*HightTab 2*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String','Z', ...
    'Fontsize',14,...
    'Tag','StaticText1');
Z_edit=uicontrol('Parent',MainWindow, ...
    'Position',[15.3*WidthTab 11*HightTab 2*WidthTab 1*HightTab],...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String',999, ...
    'Fontsize',14,...
    'Tag','StaticText1');
uicontrol('Parent',MainWindow, ...
    'BackgroundColor',[0.6 0.8 0.8],...
    'Position',[14.5*WidthTab 10*HightTab 4*WidthTab 0.6*HightTab], ...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String','Angle of Laser Beam', ...
    'Fontsize',14,...
    'Tag','StaticText1');
LasserAngle_edit=uicontrol('Parent',MainWindow, ...
    'Position',[15.3*WidthTab 9*HightTab 2*WidthTab 1*HightTab],...
    'HorizontalAlignment','center',...
    'Style','text', ...
    'String',PathData.LasserAngle*180/pi, ...
    'Fontsize',14,...
    'Tag','StaticText1');
LeftAxes      = axes('Parent',MainWindow, ...
    'Units','pixels', ...
    'Color',[.85 .85 .85], ...
    'FontSize',8, ...
    'Position',[4*WidthTab 9*HightTab 10*WidthTab 10*HightTab], ...
    'Tag','LeftAxes', ...
    'XColor',[0 0 0], ...
    'XGrid','off', ...
    'YColor',[0 0 0], ...
    'YGrid','on', ...
    'ZColor',[0 0 0]);
axis([0 PathData.EleDes+10 0 GUIProp.WindowMeter]);
xlabel('Terrain','FontSize',14);
ylabel('Elevation','FontSize',14);
title('Building The Trajectory','FontSize',14);
RightAxes      = axes('Parent',MainWindow, ...
    'Units','pixels', ...
    'Color',[.85 .85 .85], ...
    'FontSize',8, ...
    'Position',[19*WidthTab 9*HightTab 10*WidthTab 10*HightTab], ...
    'Tag','RightAxes', ...
    'XColor',[0 0 0], ...
    'XGrid','off', ...
    'YColor',[0 0 0], ...
    'YGrid','on', ...
    'ZColor',[0 0 0]);
axis([0 PathData.EleDes+10 0 GUIProp.WindowMeter]);
xlabel('Terrain','FontSize',14);
ylabel('Elevation','FontSize',14);
title('Building The Terrain','FontSize',14);

%% Calling Back Functions
```

```
    function StartButton (varargin)
        TerrainShaping();
        disp('Strat simulation')
        GUIProp.EnableRun = 1;
        Q   = QuadPara();
        PathData.dt = 0.01;
        Length = length(PathData.Terrain.X);
        Xp  = zeros(12,Length);
        U   = zeros(4,Length);
        LasserData  = zeros(1,Length);
        LasserTime  = zeros(1,Length);
        Zd = PathData.EleDes*ones(1,Length);
        horizon0 = 2.5*10;
        L0 = (PathData.EleDes / cos(PathData.LasserAngle));
        X0 = (PathData.EleDes * tan(PathData.LasserAngle));
        Xa = X0*ones(1,Length)+PathData.Terrain.X;

        XX          = zeros(1,Length);
        ZZ          = zeros(1,Length);
        LQI_error           = zeros(2,Length);
        VelCommand          = zeros(2,Length);
        AtitCommand         = zeros(2,Length);
        AltAttCommand       = zeros(4,Length);
        AltAttCommand(4,:)  = ones (1,Length) * PathData.EleDes;
        VelError            = zeros(2,Length);
        AltAttError         = zeros(4,Length);
        LM = zeros(1,Length);

        IndCross0           = 60;
        VelError_int        = [0;0];
        AltAttError_int     = [0;0;0;0];
        zold = 0;
        m0 = 0;
        i = 0;
        u_ = 9.8;
        z_ = 40;
        xplan = [40;0];
        while (GUIProp.EnableRun == 1)
            i = i + 1;
            if (i == Length/(PathData.Terrain.Rep-1)); i = 1; end
            %%
            if i == 1                   % i  = 0 -> initial state
                tc = 0;
                Xp(6,i)     = PathData.EleDes;      % initial Z position
                Xp(10,i)    = PathData.xdot_des;    % Linear Velocity for x
            elseif i > 1                % i >= 2 -> next step
                tc = tc + PathData.dt;
            end
            %% Cacluate Lasser Length
            Maxl = 70;     % The lenght of the lasser beam
            xl = [Xp(4,i)  Xp(4,i)+Maxl*sin(PathData.LasserAngle)];   % the lenght
of the x axis
            yl = [Xp(6,i)  Xp(6,i)-Maxl*cos(PathData.LasserAngle)];   % the lenght
of the y axis
            [xi, yi] = polyxpoly(xl,yl,PathData.Terrain.X,PathData.Terrain.Ele);
            if (isempty(xi))
                xi = xl(2);
                yi = yl(2);
            end
            L = sqrt( (xl(1)-xi(1))^2 + (yl(1) - yi(1))^2);
            XF = [Xp(4,i) xi(1)];
            LF = [Xp(6,i) yi(1)];
            %% Lasser Data Filter
```

```
            switch GUIProp.FilterMode
                case(2)
                    n = 1;
                    xf = L + (rand(1)*n-n/2);
                    L =  xf;
                case(3)
                    n = 1;
                    r = L + (rand(1)*n-n/2);
                    B               = [1 0.88];
                    A               = 1;
                    lpf             = filter(B,A,r);
                    L = lpf;
            end
            LasserData(i) = L;
            LasserTime(i) = tc;
            %% Data Computation from Lasser Data
            Ld = L0 - LasserData(i);
            zd = Ld * cos(PathData.LasserAngle);
            xa = ((LasserData(i)*X0)/L0);
            m = floor((xa + Xp(4,i))*10);
            if (i == 1)
                m0 = m-1;
            end
            Xa(m0:m)= linspace(Xa(m0),xa + Xp(4,i),m-m0+1);  % it is a new way of
presenting x.
            Zd(m0+1:m) = zd + Xp(6,i);
            m0 = m;
            %% Method of Path Planning
            switch  GUIProp.PathMode
                case (1)
                    % Guassian
                    segma_up   = 25;
                    segma_down = 5;
                    segma      = segma_up;
                    if Xp(6,i) - PathData.EleDes > 3.5
                        segma = segma_down;
                    elseif Xp(6,i)- PathData.EleDes < 2
                        segma = segma_up;
                    end
                    VL           = Zd(i:i+10);
                    gaus         = fspecial('gaussian',[1 2*length(VL)],segma);
                    g            = gaus(1:length(VL));
                    Norm         = sum(g);
                    % PATH PLANING:
                    XX(i) =  Xp(4,i);
                    ZZ(i) =  (sum(VL .* g)/Norm) ;
                case (2)
                    horizon1 = i + horizon0;
                    horizon2 = i + 2*horizon0;
                    horizon3 = (horizon1+horizon2)/2;
                    x_bef_spline = [Xp(4,i)  Xa(horizon1) Xa(round(horizon3))
Xa(horizon2)];
                    y_bef_spline = [Xp(6,i)  Zd(horizon1) Zd(round(horizon3))
Zd(horizon2)];
                    xspline = Xp(4,i):1:Xa(horizon2);
                    yspline = interp1(x_bef_spline,y_bef_spline,xspline,'cubic');
                    action = 3;
                    XX(i) = xspline(action);
                    ZZ(i) = yspline(action);
                case (3)
                    horizon2 = i + 2*horizon0;
                    x_bef_spline = [Xp(4,i)   Xa(horizon2)];
                    y_bef_spline = [Xp(6,i)   Zd(horizon2)];
```

```
                    xspline = Xp(4,i):1:Xa(horizon2);
                    yspline = interp1(x_bef_spline,y_bef_spline,xspline,'linear');
                    action = 3;
                    XX(i) = xspline(action);
                    ZZ(i) = yspline(action);
                case (4)
                    horizon = i + horizon0;
                    Ys = Zd(horizon);
                    x0 = [Xp(6,i);Xp(12,i)];
                    z_ = Q.mpcPath.runMPC(x0,Ys,z_,00.0,[Ys;0]);
                    XX(i) = Xp(4,i) + 0.1;
                    ZZ(i) = z_;
                    disp( ' here = L ')
                case (5)
                    XX(i) = Xa(i);
                    ZZ(i) = Zd(i);
            end
            %% Command Order
            i0 = i;
            displacment = 50;
            AltAttCommand(4,i)  = ZZ(i0);
            VelCommand(1,i)     = PathData.xdot_des;
            %% Control
            switch GUIProp.ControlMode
                case(1)
                    [AltAttCommand(1:2,i),VelError(:,i+1),VelError_int] =
VelocityControl(Q, Xp(:,i), VelCommand(:,i)   , VelError(:,i)   , VelError_int   ,
PathData.dt);
                    [U(:,i),AltAttError(:,i+1),AltAttError_int]         =
AltAttControl  (Q, Xp(:,i), AltAttCommand(:,i), AltAttError(:,i), AltAttError_int,
PathData.dt);
                    U(2,i) = min(max(U(2,i),-5),5 );
                    U(4,i) = min(max(U(4,i),0),100);
                case(2)
                    tmp_ = [10;AltAttCommand(4,i)]-[Xp(10,i);Xp(6,i)];
                    LQI_error(:,i+1) = tmp_ * PathData.dt + LQI_error(:,i);
                    U_2 = -Q.K*(-
[0;40;0;10;0;0;0]+[Xp(2,i);Xp(6,i);Xp(8,i);Xp(10,i);Xp(12,i);LQI_error(:,i)]);
                    U_2(1) = min(max(U_2(1),-5),5 );
                    U_2(2) = min(max(U_2(2)+Q.g,0),100);
                    U(:,i) = [0;U_2(1);0;U_2(2)];
                case(3)
                    [AltAttCommand(1:2,i),VelError(:,i+1),VelError_int] =
VelocityControl(Q, Xp(:,i), VelCommand(:,i)   , VelError(:,i)   , VelError_int   ,
PathData.dt);
                    [U(:,i),AltAttError(:,i+1),AltAttError_int]         =
AltAttControl  (Q, Xp(:,i), AltAttCommand(:,i), AltAttError(:,i), AltAttError_int,
PathData.dt);
                    ys = AltAttCommand(4,i);
                    us = 9.8;
                    xs = [ys;0] ;
                    Umpc = Q.mpcControl.runMPC(Xp([6,12],i),ys,u_,us,xs);
                    u_ = Umpc(1);
                    U(4,i) = u_;
            end

            %% System Dynamics
            Xp_dot = QM(Q,Xp(:,i),U(:,i));
            Xp(:,i+1) = Integra(Xp_dot,Xp(:,i), PathData.dt);
            Update(Xp(:,i+1));

            %% Plot
            if i < 16*PathData.PointPerMeter
```

```
                start = 1;
            else
                start = i - 15*PathData.PointPerMeter;
            end
            if GUIProp.PathMode == 1 || GUIProp.PathMode == 3 || GUIProp.PathMode
== 5
                plot(RightAxes,...
                    Xp(4,i+1)                       ,Xp(6,i+1)       ,'*k',...
                    PathData.Terrain.X(i:i+500)
,PathData.Terrain.Ele(i:i+500),'k',...
                    XF       ,LF  ,'b',...
                    XF(2)    ,LF(2)    ,'*g',...
                    XX(i),ZZ(i),'Ob',...
                    Xa(horizon1),Zd(horizon1),'*b',...
                    Xa(round(horizon3)),Zd(round(horizon3)),'*r',...
                    Xa(horizon2),Zd(horizon2),'*g')
                axis(RightAxes,[PathData.Terrain.X(i)-1 PathData.Terrain.X(i+500) 0
50]);
                xlabel('Terraint','FontSize',14);
                ylabel('Altitude','FontSize',14);
                title('Building The Desired Path','FontSize',14);

                plot(LeftAxes,...
                    XX,ZZ,'.k',...
                    Xp(4,1:i+1),Xp(6,1:i+1),'.b',...
                    Xp(4,i+1),Xp(6,i+1),'*r',...

PathData.Terrain.X(start:i+500),PathData.Terrain.Ele(start:i+500) ,'k')
                axis(LeftAxes,[PathData.Terrain.X(i)-15 PathData.Terrain.X(i+500) 0
50]);
                xlabel('Terraint','FontSize',14);
                ylabel('Altitude','FontSize',14);
                title('Building The Terrain','FontSize',14);

                pause(GUIProp.WaitPeriod);
            else
                plot(RightAxes,...
                    Xp(4,i+1)                       ,Xp(6,i+1)       ,'*k',...
                    PathData.Terrain.X(i:i+500)
,PathData.Terrain.Ele(i:i+500),'k',...
                    XF       ,LF  ,'b',...
                    XF(2)    ,LF(2)    ,'*g',...
                    XX(i),ZZ(i),'Ob',...
                    xspline,yspline,'m',...
                    x_bef_spline,y_bef_spline,'r')
                axis(RightAxes,[PathData.Terrain.X(i)-1 PathData.Terrain.X(i+500) 0
50]);
                xlabel('Terraint','FontSize',14);
                ylabel('Altitude','FontSize',14);
                title('Building The Desired Path','FontSize',14);

                plot(LeftAxes,...
                    XX,ZZ,'.k',...
                    Xp(4,1:i+1),Xp(6,1:i+1),'.b',...
                    Xp(4,i+1),Xp(6,i+1),'*r',...

PathData.Terrain.X(start:i+500),PathData.Terrain.Ele(start:i+500) ,'k')
                axis(LeftAxes,[PathData.Terrain.X(i)-15 PathData.Terrain.X(i+500) 0
50]);
                xlabel('Terraint','FontSize',14);
                ylabel('Altitude','FontSize',14);
                title('Building The Terrain','FontSize',14);
```

```
                pause(GUIProp.WaitPeriod);
            end
        end
    end
%%
    function StopButton (varargin)
        GUIProp.EnableRun = 0;
        disp('Stop button');
    end
%%
    function CloseButton (varargin)
        GUIProp.EnableRun = 0;
        close(MainWindow);
        clear;
    end
%%
    function SaveButton (varargin)
        GUIProp.EnableRun = 0;
        disp('Save button');
        FileName = input('Enter FileName','s');
        save(FileName,'GUIProp','Xp','PathData','XX','ZZ','Zd','U');
        disp('Exit button');
        close(MainWindow);
        clear;
    end
%%
    function EleDesSliding (varargin)
        PathData.EleDes  = get(EleDes_scroll,'value');
        disp(['Elevation = ',num2str(PathData.EleDes)]);
    end
%%
    function LasserAngleSliding (varargin)
        PathData.LasserAngle = get(LasserAngle_scroll,'value');
        disp(['Lasser Angle= ',num2str(PathData.LasserAngle)]);
        set(LasserAngle_edit  ,'string',PathData.LasserAngle*180/pi);
    end
%%
    function xdot_desSliding (varargin)
        PathData.xdot_des = get(xdot_des_scroll,'value');
        disp(['QSpeed= ',num2str(PathData.xdot_des)]);
    end
%%
    function WaitPeriodSliding (varargin)
        %size (varargin)
        GUIProp.WaitPeriod = get(WaitPeriod_scroll,'value');
        disp(['Speed= ',num2str(GUIProp.WaitPeriod)]);
    end
%%
    function NoisePowerSliding (varargin)
        GUIProp.NoisePower = get(NoisePower_scroll,'value');
        disp(['Noise= ',num2str(GUIProp.NoisePower)]);
    end
%%
    function PathShaping (varargin)
        GUIProp.PathMode = get(Method_pop,'value');
        switch GUIProp.PathMode
            case(1)
                PathString = 'Gaussian Filter';
            case(2)
                PathString = 'Cubic Spline';
            case(3)
                PathString = 'Linear Spline';
            case(4)
```

```
                PathString = 'MPC';
            case(5)
                PathString = 'No Path';
        end
        disp(['Path planning mode = ',PathString]);
    end
%%
    function FilterShaping (varargin)
        GUIProp.FliterMode = get(Filter_pop,'value');
        switch GUIProp.FliterMode
            case(1)
                ModeString = 'Ideal Case';
            case(2)
                ModeString = 'without Filter';
            case(3)
                ModeString = 'Low Pass Filter ';
        end
        disp(['Fliter mode = ',ModeString]);
    end
%%
    function ControlShaping (varargin)
        GUIProp.ControlMode = get(Control_pop,'value');
        switch GUIProp.ControlMode
            case(1)
                ModeString = 'PID';
            case(2)
                ModeString = 'LQI';
            case(3)
                ModeString = 'MPC';
            case(4)
                ModeString = 'Feedback Linearization';
        end
        disp(['Control mode = ',ModeString]);
    end
%%
    function TerrainShaping (varargin)
        GUIProp.TerrainMode = get(Terrain_pop,'value');
        PathData.Terrain.Width = 30;
        PathData.Terrain.sinWidth = 1.5;
        XT = 0: 1/PathData.PointPerMeter : (GUIProp.WindowMeter +
PathData.Terrain.Width);
        XTFT = 0: 1/PathData.PointPerMeter : (GUIProp.WindowMeter +
(PathData.Terrain.Width-10));  % FOR DOUBLE STEP
        XTFR = 0: 1/PathData.PointPerMeter : 30;  % FOR DOUBLE STEP

        switch GUIProp.TerrainMode
            case (1)
                ModeString = 'Step';
                x0 = GUIProp.WindowMeter;
                xf = GUIProp.WindowMeter + PathData.Terrain.Width;
                Trapezoidel = trapmf(XT,[x0 x0 xf xf]);
                EleSignal = 5*Trapezoidel;
            case(2)
                ModeString = 'Trap';
                dx = PathData.Terrain.Width/4;
                x0 = GUIProp.WindowMeter;
                x1 = x0 + 1*dx;
                x2 = x0 + 3*dx;
                x3 = x0 + 4*dx;
                Trapezoidel = trapmf(XT,[x0 x1 x2 x3]);
                EleSignal = 5*Trapezoidel;
            case(3)
                ModeString = 'Sin';
```

```
                sinfunction =  sin(0.5*PathData.Terrain.X) ;
                sinfunction =
2*sinfunction(1:PathData.Terrain.sinWidth:100*PathData.PointPerMeter)+1.5;
                EleSignal = [zeros(1,GUIProp.WindowMeter*PathData.PointPerMeter)
sinfunction];
            case(4)
                ModeString = 'Ramp';
                dx = PathData.Terrain.Width/4;
                x0 = GUIProp.WindowMeter;
                x1 = x0 + 2*dx;
                x2 = x0 + 4*dx;
                Trapezoidel = trapmf(XT,[x0 x1 x1 x2]);
                EleSignal = 5*Trapezoidel;
            case(5)
                ModeString = 'Sin';
                sinfunction =  sin(0.5*PathData.Terrain.X) ;
                sinfunction =
2.5*sinfunction(1:PathData.Terrain.sinWidth:50*PathData.PointPerMeter)+6;
                EleSignal = [zeros(1,GUIProp.WindowMeter*PathData.PointPerMeter)
sinfunction];
            case(6)
                ModeString = 'Double Step';
                x0 = 0.7*60;%GUIProp.WindowMeter1;
                xf = 0.5*60 + (PathData.Terrain.Width-10);
                x1 = 55;%GUIProp.WindowMeter;
                xf1= 55 + (PathData.Terrain.Width-20);
                Trapezoidel = trapmf(XTFT,[x0 x0 xf xf])+ trapmf(XTFT,[x1 x1 xf1
xf1]);
                EleSignal = 5*Trapezoidel;
            case(7)
                ModeString = 'Random';
                x0 = rand(1);
                xf = rand(1);
                Trapezoidel = trapmf(XTFR,[x0 x0 xf xf])
                EleSignal = 5*Trapezoidel;
        end
        PathData.Terrain.Ele = repmat(EleSignal,1,PathData.Terrain.Rep);
        PathData.Terrain.X =  0: 1/PathData.PointPerMeter :
(length(PathData.Terrain.Ele)-1)/PathData.PointPerMeter;
        disp(['Terrain mode = ',ModeString]);
    end
%%
    function Update(Xp)
        set(Pitch_edit  ,'string',Xp(2));
        set(X_edit      ,'string',Xp(4));
        set(Z_edit      ,'string',Xp(6));
        set(X_dot_edit  ,'string',Xp(10));
    end
end
```

B.2 Code for Parameters

```
function Q = QuadPara()
Q.g     = 9.8;
Q.m     = 1;
Q.Ixx   = 0.002;
Q.Iyy   = 0.002;
Q.Izz   = 0.01;
%% Parametrs for PID Controller
% Q.AAC.Kp = [R_Kp ; P_Kp ; W_Kp ; Z_Kp ]
Q.AAC.Kp   = [0.0  ; 0    ; 0.0  ;  13.8 ];
```

```
Q.AAC.Ki   = [0.0  ; 0    ; 0.0  ;  8.27 ];
Q.AAC.Kd   = [0.0  ; 0.01 ; 0.0  ;  6.07 ];

% Q.VC.Kp = [Xdot_Kp ; Ydot_Kp ]
Q.VC.Kp    = [9      ; 0.0  ];
Q.VC.Ki    = [0      ; 0.0  ];
Q.VC.Kd    = [0.5    ; 0.0  ];

%% LQI
% Z Ssyetm
ZA   = [ 0 1; 0 0];
ZB   = [ 0 (1/Q.m)]';
ZC   = [ 1 0];
ZD   = 0;
ZSys = ss(ZA,ZB,ZC,ZD);
ZQ   = diag([ 0.01 0.01  10000]);
ZR   = 10*eye(1,1);
Kz = lqi(ZSys,ZQ,ZR);

%Pitch and X Ssyetm
PAF   = [ 0 0 1 0; 0 0 0 1;0 0 0 0;Q.g 0 0 0 ];
PBF   = [ 0 0 (1/Q.Iyy) 0]';
PCF   = [ 0 0 0 1];
PDF   = 0;
PSys = ss(PAF,PBF,PCF,PDF);
Msys_ = minreal(PSys);
PQ   = diag([ 100 10 10 1000]);
PR   = 10*eye(1,1);
Kpx = lqi(Msys_,PQ,PR);

Q.K = [Kpx(1)   0       Kpx(2) Kpx(3)   0  Kpx(4)  0;
         0     Kz(1)     0         0  Kz(2) 0    Kz(3)];

%% Continues System for MPC Control:
Ac_ = [ 0   1 ;
        0   0];
Bc_ = [0;
       1];
Cc_ = [1  0];
Dc_ = 0;
Sysc_ = ss(Ac_,Bc_,Cc_,Dc_);
[nc_,mc_] = size(Bc_);
P_ = 500*eye(nc_);
Q_ = diag([100,0.01]);
R_ = 0.01;
N_ = 50; % Prediction horizon
M_ = 50; % Control horizon
umin_ = 0;
umax_ = 100;
dumin_ = -3;
dumax_ =  3;
ymax_ = 50;
ymin_ = 0;
xmax_ = [ ymax_;  10.0];
xmin_ = [-ymin_; -10.0];
dt_ = 0.01;
```

```
Q.mpcControl =
MPC('cSys',Sysc_,'Q',Q_,'R',R_,'S',P_,'N',N_,'M',M_,'dt',dt_,...
    'umax',umax_,'umin',umin_,'dumax',dumax_,'dumin',dumin_,...
    'ymax',ymax_,'ymin',ymin_,'xmax',xmax_,'xmin',xmin_);

end
```

B.3 Code for Modeling System

```
%% Quadrotor Modeling
function Xp_dot = QM(Q,Xp,U)

% Xp = [roll;pitch;yaw;x;y;z;roll_dot;pitch_dot;yaw_dot;x_dot;y_dot;z_dot
]
% U  = [Ur;Up;Uw;Uz]

% Definition:
    R      = 0;    %Xp(1);
    P      = Xp(2);
    W      = 0;    %Xp(3);
    X      = 0;    %Xp(4)
    Y      = 0;    %Xp(5)
    Z      = 40;   %Xp(6)

% Dynamics Equation:
    R_dot  = 0;     %Xp(7);
    P_dot  = Xp(8);
    W_dot  = 0;     %Xp(9);
    X_dot  = Xp(10);
    Y_dot  = 0;     %Xp(11);
    Z_dot  = Xp(12);

    R_ddot = 0;            %((Q.Iyy - Q.Izz)*W_dot*P_dot + U(1))/Q.Ixx;
    P_ddot =  U(2)/Q.Iyy;
    W_ddot = 0;            %((Q.Ixx - Q.Iyy)*P_dot*R_dot + U(3))/Q.Izz;
    X_ddot = P* U(4)/Q.m;  %(  sin(W)*sin(P) + cos(W)*sin(P)*cos(R))*
    Y_ddot = 0;            %(- cos(W)*sin(P) +
sin(W)*sin(P)*cos(R))*U(4)/Q.m;
    Z_ddot = -Q.g + cos(R)*cos(P)*U(4)/Q.m;



    Xp_dot  =
[R_dot;P_dot;W_dot;X_dot;Y_dot;Z_dot;R_ddot;P_ddot;W_ddot;X_ddot;Y_ddot;Z_
ddot];
end
```

B.4 Code for Altitude Control

```
%% Altitude Control (Z axis)
function [U,AltAttError,AltAttError_int] = AltAttControl(Q, Xp,
AltAttCommand,AltAttError_old,AltAttError_int_old,dt)
AltAttError     =  AltAttCommand - [Xp(1,1);Xp(2,1);Xp(3,1);Xp(6,1)];
AltAttError_dot = (AltAttError - AltAttError_old) / dt;
```

```
AltAttError_int =  (AltAttError+AltAttError_old)*dt/2 +
AltAttError_int_old;
U              =  AltAttError .* Q.AAC.Kp + AltAttError_dot .* Q.AAC.Kd +
AltAttError_int .* Q.AAC.Ki +[0;0;0;Q.g];
end
```

B.5 Code for Velocity Control

```
%% Velocity Control
function [AtitCommand,VelError,VelError_int] = VelocityControl(Q, Xp,
VelCommand,VelError_old,VelError_int_old,dt)
VelError       =  VelCommand(:,1) - [Xp(10,1);Xp(11,1)];
VelError_dot   = (VelError - VelError_old) / dt;
VelError_int   =  VelError*dt + VelError_int_old;
Command        =  VelError .* Q.VC.Kp + VelError_dot .* Q.VC.Kd +
VelError_int .* Q.VC.Ki;
PCom           =  Command(2,1); % from x-axis -> pitch
RCom           =  Command(1,1); % from y-axis -> roll
AtitCommand    = [PCom;RCom];
AtitCommand    = min(max(AtitCommand,-0.2),0.2);
End
```