1-13-2016 12:00 AM

Application of Simultaneous Localization and Mapping Algorithms for Haptic Teleoperation of Aerial Vehicles

Bandar Hulayyil Aldhafeeri, *The University of Western Ontario*

Supervisor: Ilia Polushin, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

In this thesis, a new type of haptic teleoperator system for remote control of Unmanned Aerial Vehicles (UAVs) has been developed, where the Simultaneous Localization and Mapping (SLAM) algorithms are implemented for the purpose of generating the haptic feedback. Specifically, the haptic feedback is provided to the human operator through interaction with artificial potential field built around the obstacles in the virtual environment which is located at the master site of the teleoperator system. The obstacles in the virtual environment replicate essential features of the actual remote environment where the UAV executes its tasks. The state of the virtual environment is generated and updated in real time using Extended Kalman Filter SLAM algorithms based on measurements performed by the UAV in the actual remote environment. Two methods for building haptic feedback from SLAM algorithms have been developed. The basic SLAM-based haptic feedback algorithm uses fixed size potential field around the obstacles, while the robust SLAM-based haptic feedback algorithm changes the size of potential field around the obstacle depending on the amount of uncertainty in obstacle location, which is represented by the covariance estimate provided by EKF. Simulations and experimental results are presented that evaluate the performance of the proposed teleoperator system.

Keywords: SLAM, EKF, UAV, Haptic, Teleoperation

Acknowledgments

First and foremost, I would like to thank my supervisor Dr. Ilia Polushin for accepting me under his supervision. Without his support, guidance and encouragement, I probably could not manage to fulfil and complete my thesis. I have learned a lot from him. I am also indebted to my co-supervisor Professor Abdelhamid Tayebi at Lakehead University for his invaluable feedback.

My colleagues in the lab Mr. Mir Zayed Hasan, Dr. Amir Takhmar and Dr. Ali Moatadelro deserve my sincere thanks. We had long informative and precious discussions during the period I was doing my work in the lab. Their feedback is highly appreciated. I would like also to take this opportunity to thank the University of Western Ontario for having me as a graduate student and giving me this chance to pursue the Master's degree at their campus.

To my family, my mother, my father, my sisters and brothers, thank you so much for your love, patience, motivation, and being always supportive and helpful.

Contents

Abstract ii

Acknowledgments iii

List of Abbreviations viii

List of Figures ix

List of Tables xv

List of Appendices xvi

1 Introduction 1

1.1 Unmanned Aerial Vehicles 1

1.2 Simultaneous Localization and Mapping 2

1.3 Haptic Technology 4

1.4 Teleoperation 5

1.5 Visual and Haptic Feedback in Teleoperation 6

1.6 Motivation 8

1.7 Thesis Contribution 9

1.8 Thesis Outline 10

2 Kinematics, Dynamics, and Control of a Quadrotor UAV 12

2.1 Basic Quadrotor Operation 12

2.2 Quadrotor Kinematics 17

2.3 Quadrotor Dynamics . . . 18
2.4 Quadrotor Control Strategy . . . 21
2.4.1 Control Strategy for Linear Model . . . 22
Attitude Control for Linear Model . . . 24
Position Control for the Linear Model . . . 28
2.4.2 Control Strategy for Nonlinear Quadrotor Model . . . 32
2.4.3 Backstepping Controller . . . 36
2.4.4 Attitude Control for Nonlinear Model . . . 37
Design of Upwards Thrust Control Input $\mathbf{U}_1$ . . . 37
Design of Roll Control Input $\mathbf{U}_2$ . . . 40
Design of Pitch Control Input $\mathbf{U}_3$ . . . 43
Design of Yaw Control Input $\mathbf{U}_4$ . . . 46
Simulation Results for Attitude Backstepping Control . . . 48
2.4.5 Position Control for Nonlinear Model . . . 51
Design of Virtual Control Inputs $\mathbf{U}_x$, $\mathbf{U}_y$ . . . 53
Simulation Results for Backstepping Position Control . . . 56
2.4.6 Spatial Velocity Control for UAV . . . 57
Simulation Results for Backstepping Velocity Control . . . 62
2.5 Conclusion . . . 62

3 Simultaneous Localization and Mapping **66**

3.1 Introduction . . . 66
3.2 Structure of Probabilistic SLAM . . . 67
3.2.1 Random Variables and Belief Distributions . . . 67
3.2.2 Map Representations . . . 70
3.2.3 Probabilistic SLAM . . . 72
3.2.4 Parametric Filters . . . 73
3.3 Extended Kalman Filter SLAM Algorithm for a Quadrotor UAV . . . 75

3.3.1 Motion Model . . . 75
3.3.2 Observation Model . . . 79
3.3.3 Data Association . . . 81
3.3.4 Constructing EKF-SLAM . . . 83
EKF Prediction Step . . . 84
EKF Update Step . . . 85
System Initialization . . . 85
Adding a new beacon . . . 85
3.4 EKF-SLAM Algorithm . . . 89
3.5 Conclusion . . . 92

4 Teleoperation of UAVs with SLAM-based Haptic Feedback 93
4.1 Introduction . . . 93
4.2 Predictive Displays . . . 94
4.3 SLAM-Based Haptic Feedback . . . 95
4.4 The Artificial Potential Field . . . 97
4.4.1 Models for Artificial Potential Field . . . 99
4.4.2 Experimental results for APFs . . . 102
4.5 Control Structure of a Teleoperator System with SLAM-based Haptic Feedback 110
4.6 Algorithms for SLAM-based haptic feedback . . . 114
4.6.1 Basic SLAM-based haptic feedback algorithm . . . 115
4.6.2 Semi-experimental results for basic SLAM-based haptic feedback algorithm . . . 116
4.6.3 Robust SLAM-based haptic feedback algorithm . . . 129
4.6.4 Semi-experimental results for robust SLAM-based haptic feedback algorithm . . . 133
4.7 Conclusion . . . 143

5 Conclusion **144**

5.1 Summary 144

5.2 Future Research 145

Bibliography **146**

A Basic Probability Notions **158**

B Phantom Omni Device **161**

B.1 Introduction 161

B.2 Forward and Inverse Kinematics of Phantom Omni 162

B.3 Jacobian Matrix 163

B.4 OpenHaptics Toolkit 164

Curriculum Vitae **165**

List of Abbreviations

UAV : Unmanned Aerial Vehicle

SLAM : Simultaneous Localization and Mapping

EKF : Extended Kalman Filter

APF : Artificial Potential Field

List of Figures

1.1 Scientific and engineering disciplines related to haptics [1]. 5

1.2 Teleoperation system structure. 6

2.1 Cross configruation of the quadrotor, view from the top. 13

2.2 Generating the upwards thrust by increasing/decreasing all motors' velocities simultaneously with the same magnitude. The movement direction in this figure is upward. 14

2.3 Generating roll torque. Increase in angular velocity of motor 2 and decrease in angular velocity of motor 4 while keeping the angular velocities of motors 1 and 3 lower than that of motor 2 and greater than that of motor 4 causes the platform to roll about x axis subsequently move in the direction shown. 15

2.4 Generating the pitch torque. Increase in angular velocity of motor 3 and decrease in angular velocity of motor 1 while keeping the angular velocities of motors 2 and 4 lower than that of motor 3 and greater than that of motor 1 causes the platform to pitch about y axis subsequently move in the direction shown. 16

2.5 Generating yaw torque. Increase in angular velocities of motors 3 and 1 and decrease in angular velocities of motors 4 and 2 generates clockwise rotation due to drag torque. 16

2.6 Two common representations of the inertial frame under the assumption of flat Earth surface represented by the blue plane, where the golden sphere represents the Earth. 17

2.7 The representations of the body frame in NED and ENU coordinates. 18

2.8 Modeling the rotor as a rotating disk; the figure shows the direction of the air flow through the rotor. 19

2.9 General attitude control scheme, where $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$, and $\mathbf{U}_4$ are the control inputs. 22

2.10 General position control scheme, where $\mathbf{U}_x$ and $\mathbf{U}_y$ are the virtual inputs that control position on the $x-y$ plane. 23

2.11 Attitude regulation problem: altitude and attitude errors for PD controller. . . . 25

2.12 Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PD controller. 26

2.13 Attitude regulation problem: magnitude of the control inputs for PD controller. 27

2.14 Attitude regulation problem: altitude and attitude errors for PID controller. . . 29

2.15 Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PID controller. 30

2.16 Attitude regulation problem: the magnitude of the control inputs for the PID control. 31

2.17 Position regulation problem: errors of linear positions and yaw angle for PD controller. 33

2.18 Position regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PD controller. 34

2.19 Position regulation problem: the magnitude of the control inputs for PD controller. 35

2.20 Attitude regulation problem: altitude and attitude errors for backstepping controller. 49

2.21 Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass using backstepping controller. 50

2.22 Attitude regulation problem: magnitude of the control inputs for backstepping controller 52
2.23 Position regulation problem: errors of linear positions and yaw angle for backstepping controller. 58
2.24 Position regulation problem: the position (x, y, z) of the center of the quadrotor's mass for backstepping controller. 59
2.25 The magnitude of control inputs of backspteing controller for attitude control (regulation problem). 60
2.26 Velocity regulation problem: errors of linear velocies and yaw angle for backstepping controller. 63
2.27 Velocity regulation problem: the position (x, y, z) of the center of the quadrotor's mass for backstepping controller. 64
2.28 The magnitude of control inputs of backspteing controller for velocity control (regulation problem). 65

3.1 Bell curves of multiple belief distributions. 70
3.2 Metric map representations: a feature-based map (left); a cell-based map (right). 71
3.3 Representation of the quadrotor's frame F^L in F^G 76
3.4 3D sensor that provides observations to a beacon 80

4.1 Simple predictive display model for bilateral teleoperation systems. 94
4.2 Structure of the teleoperator system with SLAM-based haptic feedback 96
4.3 2D illustration of the method for generating the SLAM-based haptic feedback. 97
4.4 Direction of APF force vectors in the vicinity of obstacles. 99
4.5 Algorithm 4: APF based on the penetration depth only 100
4.6 Algorithm 5: APF with velocity-dependent term. 102
4.7 GUI used in the experiments with 2D APF. 104

4.8 Response of 2D APF: the stiffness and the damping components of the reflected force. 105
4.9 GUI used in the experiments with 3D APF. 107
4.10 Response of 3D APF: the stiffness and the damping components of the reflected force. 108
4.11 Control structure of a teleoperator system with SLAM-based haptic feedback. . 110
4.12 An example of a feature-based map of the environment. 112
4.13 A virtual environment constructed by EKF-SLAM algorithm on the master side which corresponds to the map in Figure 4.12. 113
4.14 3D potential force field around a beacon penetrated by the quadrotor. 114
4.15 Illustration of Algorithm 6. 116
4.16 The trajectory of the quadrotor on the slave side. 117
4.17 Estimates of the beacons' locations at the master side (blue spheres) vs. true locations (shaded red spheres). 118
4.18 Basic SLAM-based haptic feedback: GUI which shows the master (left) and slave (right) sides. 119
4.19 Basic SLAM-based haptic feedback experiment, beacon 1. Estimated and true location vs. time (left); estimation errors vs. time (right). 121
4.20 Basic SLAM-based haptic feedback experiment, beacon 1. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 122
4.21 Basic SLAM-based haptic feedback experiment, beacon 1. The reflected force components along x, y, and z axes vs. time. 122
4.22 Basic SLAM-based haptic feedback experiment, beacon 2. Estimated and true location vs. time (left); estimation errors vs. time (right). 123
4.23 Basic SLAM-based haptic feedback experiment, beacon 2. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 124

4.24 Basic SLAM-based haptic feedback experiment, beacon 2. The reflected force components along x, y, and z axes vs. time. 124

4.25 Basic SLAM-based haptic feedback experiment, beacon 3. Estimated and true location vs. time (left); estimation errors vs. time (right). 125

4.26 Basic SLAM-based haptic feedback experiment, beacon 3. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 126

4.27 Basic SLAM-based haptic feedback experiment, beacon 3. The reflected force components along x, y, and z axes vs. time. 126

4.28 Basic SLAM-based haptic feedback experiment, beacon 4. Estimated and true location vs. time (left); estimation errors vs. time (right). 127

4.29 Basic SLAM-based haptic feedback experiment, beacon 4. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 128

4.30 Basic SLAM-based haptic feedback experiment, beacon 4. The reflected force components along x, y, and z axes vs. time. 128

4.31 3D Gaussian distribution represented as an ellipsoid with different confidence levels. 130

4.32 The side-effect of buidling a fixed APF where the green, red, yellow, blue shapes represent APF, a possible true location of a beacon, a beacon's estimate and the ellipsoid of the beacon's estimate. 131

4.33 Uncertainty-dependent artificial potential field (UDAPF). 132

4.34 Robust SLAM-based haptic feedback experiment, beacon 1. Estimated and true location vs. time (left); estimation errors vs. time (right). 135

4.35 Robust SLAM-based haptic feedback experiment, beacon 1. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 136

4.36 Robust SLAM-based haptic feedback experiment, beacon 1. The reflected force components along x, y, and z axes vs. time. 136

4.37 Robust SLAM-based haptic feedback experiment, beacon 2. Estimated and true location vs. time (left); estimation errors vs. time (right). 137
4.38 Robust SLAM-based haptic feedback experiment, beacon 2. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 138
4.39 Robust SLAM-based haptic feedback experiment, beacon 2. The reflected force components along x, y, and z axes vs. time. 138
4.40 Robust SLAM-based haptic feedback experiment, beacon 3. Estimated and true location vs. time (left); estimation errors vs. time (right). 139
4.41 Robust SLAM-based haptic feedback experiment, beacon 3. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 140
4.42 Robust SLAM-based haptic feedback experiment, beacon 3. The reflected force components along x, y, and z axes vs. time. 140
4.43 Robust SLAM-based haptic feedback experiment, beacon 4. Estimated and true location vs. time (left); estimation errors vs. time (right). 141
4.44 Robust SLAM-based haptic feedback experiment, beacon 4. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right). 142
4.45 Robust SLAM-based haptic feedback experiment, beacon 4. The reflected force components along x, y, and z axes vs. time. 142

A.1 Venn diagram for the dependency and independency of two events. 159
A.2 Venn diagram that illustrates the total probability theorem. 160

B.1 Phantom Omni 161
B.2 Forward and Inverse Kinematics [3] 162
B.3 OpenHaptics Toolkit [2]. 164

List of Tables

2.1 PD controller parameters for the attitude regulation problem 24

2.2 PID controller parameters for the attitude regulation problem 28

2.3 PD controller parameters for position regulation problem 32

2.4 Backstepping controller parameters for attitude control (regulation problem) . . 51

2.5 Backstepping controller parameters for position control (regulation problem) . 57

2.6 Backstepping controller parameters for linear velocity control (regulation task) 62

4.1 The numerical parameters of stiffness and damping terms for 2D APF. 103

4.2 The numerical parameters of stiffness and damping terms for 3D APF. 104

4.3 Basic SLAM-based haptic feedback: True locations of Beacons at the Slave side. 118

4.4 The workspace of the Phantom Omni. 119

4.5 Locations of beacons at the slave side 133

4.6 Backstepping controller parameters for robust SLAM-based haptic feedback algorithm. 134

List of Appendices

Appendix A: Basic Probability Notions

Appendix B: Phantom Omni Device

Chapter 1

Introduction

In this chapter, some introductory material is presented, and the goals of the research presented in this thesis are formulated and discussed. In particular, a brief overview is given of the three major areas of engineering to which this thesis is related, which are Unmanned Aerial Vehicles (Section 1.1), Simultaneous Localization and Mapping (Section 1.2), and Haptics and Teleoperation (Sections 1.3-1.5). Motivation behind the research presented in this thesis is discussed in Section 1.6. Thesis contribution is described in Section 1.7, and thesis outline is given in Section 1.8.

1.1 Unmanned Aerial Vehicles

Recent advances in a number of engineering disciplines have lead to development and fabrication of efficient and powerful Unmanned Aerial Vehicles (UAVs). An UAV is defined as a machine that is capable of flying remotely without the presence of a human operator in the control cabin [4]. The ability of UAVs to perform tasks without the presence of a human operator inside the vehicle makes them suitable for numerous applications, particularly those where safety of the pilot is a major concern. Popularity of UAVs has recently increased drastically due to their applications to both military and civilian tasks [5, 6, 7, 8]. In the past, the usage of UAVs was mostly limited to military missions, however, currently there also exists substantial

and growing demand for UAVs in civilian applications [9]. In particular, UAVs can be used for scientific research [9], agricultural tasks [10], surveillance [11], mine scanning [12], search and rescue [13], aerial photography [14], *etc.* One of the most fascinating and promising civilian applications was recently announced in a letter submitted by Amazon company to the U.S. Federal Aviation Administration on July 9, 2014, which states that delivery of customers' orders might be done in 30 minutes or less by using UAVs [15]. For a survey of some current and potential applications of UAVs, both military and civilian, the reader is referred to [9].

A class of UAVs that are capable of take-off and landing vertically is known as VTOL (Vertical Take-Off and Landing) UAVs [16]. This class of aerial vehicles does not require runways, which makes them a natural choice for missions that require quick and flexible access to the incident place, such as search and rescue missions. Another important advantage of VTOL over fixed-wing aircrafts is their manoeuvrability which, in particular, allows for flying in confined spaces and/or over difficult terrains. Nowadays, VTOL vehicles come in a variety of sizes such as heavy, normal, small, mini-small, and even micro, and may have different structures such as a single rotor, two side by side rotors, two rotors with coaxial configuration, and multi-rotors [4]. The most common multi-rotor configuration is the quadrotor (four rotors aircraft), which is the type of UAV addressed in this thesis.

1.2 Simultaneous Localization and Mapping

The notion of Simultaneous Localization and Mapping (SLAM) refers to methods and algorithms that allow a robot for building a map of an unknown environment while concurrently estimating its own position and orientation on that map. Recently, SLAM has become a very active research area, and it continues to grow in popularity due to a number of promising applications. One such application is driverless cars, which has potential to change the transportation industry by fundamentally increasing safety and bringing comfort while driving over long distances. Other potential applications of SLAM techniques are related to space exploration,

mining industry, unmanned aerial vehicles, *etc.*

The term SLAM was apparently introduced in the paper [17] published in 1996. However, SLAM as a concept was known well before the term has been coined. In late 1980s, the idea of a stochastic map was introduced [18] where uncertainty is taken into account when describing transformations between spatial coordinate frames. This work can be considered as the birth of SLAM techniques, in the sense that it explicitly formulates the idea that deterministic approaches are not suitable for fully autonomous navigation, and that uncertainty must be taken into account. Nowadays, the dominant approaches for solving SLAM problem are based on probabilistic methods. From probabilistic perspective, SLAM problem can be formulated as follows: given noisy measurements and control inputs, find the probability distribution of the current state of robot and the environment under the assumption that the environment is *a priori* unknown. In Chapter 3, it will be shown how can such a probability distribution be computed using recursive Bayesian approach.

Over the last 20 years, SLAM algorithms have been implemented, simulated, and validated successfully in a variety of indoor and outdoor scenarios. In **marine environments**, the first deployable underwater mission that uses SLAM was carried out at Australian Center for Field Robotics [19]. In that project, Extended Kalman Filter (EKF)-based SLAM algorithms were implemented on an autonomous underwater vehicle (AUV) called Oberon which was used to discover and build a a feature-based map of an unknown underwater environment. Another similar project that involves autonomous underwater vehicles has been carried out at the coast of Costa Brava city, Spain [20]. EKF and Nearest Neighbor Standard Filter (NNSF) have been adopted in these experiments, while a modified Hough transform was used for obtaining line features from sonar data. In **airborne environments**, the first airborne SLAM has been implemented in [21]. The platform used in these experiments was an unmanned aerial vehicle of a fixed wing type equipped with onboard sensors (such as a camera and an inertial measurement unit). In [22], SLAM has been implemented on a UAV of a rotary wings type (*i.e.*, a mini helicopter), where a single camera was used as a sensor, hence the approach was called a visual

SLAM. The obtained results show the reliability of the visual SLAM for estimating the position of the platform relative to observed landmarks in outdoor environments. Paper [23] is an example of applications of SLAM to indoor environments. In this work, SLAM is implemented on a semi-autonomous UAV equipped with 3D laser sensor which is capable of navigating in demolished buildings and designed for handling radioactive materials. For some survey related to applications of SLAM, the reader is referred to [24, 25].

1.3 Haptic Technology

The word "haptics" is derived from a Greek word "άπτω" which means "touch" [26]. Haptic technology deals with exchanging information between the machine and a human being via the sense of touch. Nowadays, multimedia is not restricted to visual and audio feedback; in particular, haptic technology can be used in virtual environments to create more realistic scenarios. Haptic feedback become essential in numerous applications such as pilots training, surgical training, computer-aided design, and entertainment.

The two major types of haptic feedback are kinesthetic and tactile feedback. The former is related to the haptic information that human acquires via tendons, joints, and muscles, and is represented as forces/torques that are felt by the human as a result of interaction with the surrounding environment. On the other hand, the tactile feedback is acquared via sensitive receptors inside and under the skin that enable a human to feel temperature, pain, pressure, and vibrations. **Machine haptics** is a part of haptic technology that is related to design and manufacturing of haptic devices (also known as haptic interfaces or haptic displays). These are typically electro-mechanical devices that enable physical contact between the human operator and objects in the virtual/remote environment [27]. **Computer haptics** is an emerging field which deals with generating and rendering virtual objects and environments so that human operator can interact with them through the sense of touch [28]. The representation of a haptic device inside a virtual environment is called an **avatar** [1]. **Haptic rendering** is the process of

computing and generating haptic feedback [29]. Typical update rate of haptic rendering algorithms is approximately 1kHz which is higher in comparison with the typical update rates used in visual feedback; such a high update rate is necessary to guarantee stability of haptic interaction [30]. Different engineering and scientific disciplines related to haptics are schematically shown in Figure 1.1.

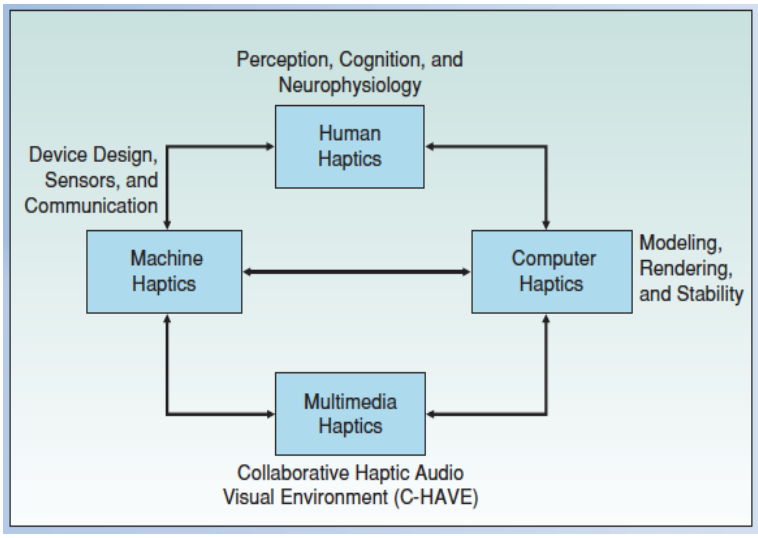


Figure 1.1: Scientific and engineering disciplines related to haptics [1].

1.4 Teleoperation

Teleoperation is defined as an extension of human operator's sensing and manipulation capabilities to a remote or otherwise unaccessible location [31]. It allows the human operator to access hazardous environments and manipulate dangerous materials without putting the operator's safety at risk [32]. It may also be used to scale up or down the human forces and motions, thus allowing the operator to perform manipulation of objects that cannot be manipulated directly as their size/weight is beyond the range of human capabilities [33]. Moreover, teleoperation technology can be used to overcome the distance barrier in different exploration applications, where it may be difficult or impossible for the human operator to perform manipulation directly; examples include space and undersea applications, among others.

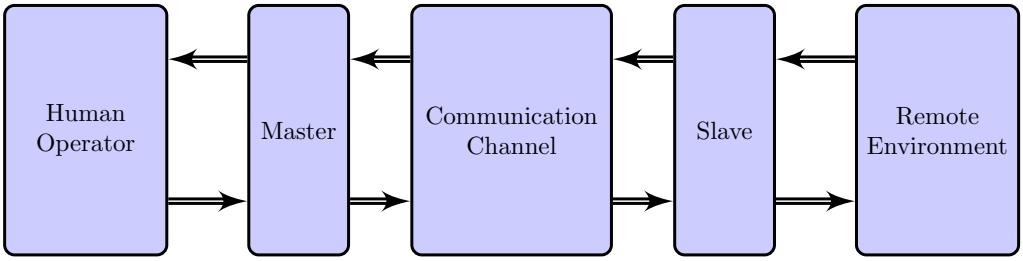


Figure 1.2: Teleoperation system structure.

The structure of a typical single-master-single-slave bilateral teleoperator system is shown in Figure 1.2. It consists of five components, which are the human operator, the master device, the slave device, an environment, and a communication channel between the master and the slave [34, 35]. The human operator controls the master device, which communicates its position, velocity, and sometimes acceleration to the remotely located slave. The slave robot follows the motion of the master thus executing a task on the remote environment. On the other hand, the slave device typically has an ability to sense the forces exerted on it due to the interaction with the remote environment; information about these forces is transferred back to the master over the communication channel. These interaction forces are subsequently presented to the human operator in the form of haptic feedback. The haptic feedback allows the human operator to feel the remote interaction, which leads to improved situation awareness and higher performance of teleoperation. **Transparency** is a measure of how well the teleoperator system replicates the motion/forces on the opposite side. A perfectly transparent teleoperator system creates an illusion for the human operator as if the task is executed directly.

1.5 Visual and Haptic Feedback in Teleoperation

In a teleoperation system, the human operator can be provided with different types of feedback, such as visual, auditory, haptic (including kinesthetic and tactile components), and in some cases even smell and taste. The purpose of all these types of feedback is to increase the

situational awareness of the human operator about the remote environment. Each type of feedback has its own advantages and disadvantages. Among all these types of feedback, the visual feedback typically provides the human operator with the most substantial information about the remote environment. Nowadays, cameras are extremely cheap and ubiquitous; moreover, they are typically light and their energy consumption is low. However, purely visual feedback suffers from numerous disadvantages. For example, a single camera does not provide the depth clues. In UAVs teleoperation, this implies that the human operator may not be able to perceive whether the platform is getting closer to or going away from the obstacles.

Providing more than one camera to acquire depth perception (such as stereo vision) is a classical solution, however, it may be unsuitable in some cases due to the fact that the payload of the platform and complexity of manipulating images will increase drastically. Moreover, cameras provide information only in the direction of view and the visual information may be limited due to occlusions; this is commonly known as the limited field of view problem. In particular, the human operator may not be able monitor blind spots to avoid potential collision.

Another common modality that is known to enhance the human operator's awareness in teleoperation is haptic feedback. Haptic feedback has some advantages over its visual counterpart. For example, the environment surrounding the slave device can be mapped into artificial force field, which eliminates the limited field of view problem. The human operator reacts faster to haptic feedback than to visual feedback [36]. Moreover, the receptors of haptic feedback covers the whole body of the human operator which means it is not limited to specific organs like ears for audition and eyes for vision [37, 38]. Another unique characteristic of haptic feedback is that the energy flow between the sender and the receiver is bidirectional [39]; in other words, action and perception are not separated in the case of haptic feedback.

Haptic feedback has also some disadvantages. Haptic rendering algorithms require higher update rate (up to 1kHz) in comparison with visual feedback. Also, one of the most substantial problems with haptic feedback is that, in the presence of communication delay, haptic feedback may lead to instability of teleoperator systems [40, 41, 42, 33]. This is due to the fact

that, in the presence of time delay, the communication channel is no longer a passive subsystem [34]. There are numerous approaches reported in the literature that deal with instability of teleoperation systems that occurs due to delayed haptic feedback. For example, **wave variables** approach stabilizes a force-reflecting bilateral teleoperator by encoding force and velocity variables into another set of variables known as wave variables, see [43, 33]. The wave variables are then sent via the communication channel and decoded once they are received to extract the force and velocity. This approach guarantees stability if the time delay is constant [44]. Other approaches suggest to abandon haptic feedback altogether and utilize other modalities such visual, auditory, vibrotactile, and graphical feedbacks to display forces. This approach is known as **sensory substitution** [45, 46, 47, 48]. One more approach to deal with instability of bilateral teleoperation generated by time delays is to rely on a virtual model of the remote environment at the master side to acquire the haptic feedback rather than exchanging haptic data over the communication channel. This approach is based on using **predictive displays**, and is described in Section 4.2.

1.6 Motivation

The primary objective of this thesis is to design a teleoperator system for remote control of a quadrotor UAV that provides the human operator with haptic and visual feedback. In particular, the goal of haptic feedback is to alert the human operator regarding proximity of the obstacles to the UAV. The physical separation between the human operator and the remote environment may make it extremely difficult to perform such a task; in particular, sufficiently rich sensory information such as auditory, visual, and haptic feedback may not be readily available due to communication constraints including delays, data loss, distortion of signals, *etc,*. This may degrade the ability of the human operator to perceive the remote environment and to control the UAV in a reliable and safe manner. One possible approach to solve the teleoperation problem under such constraints is to use the virtual environment approach, which consists of build-

ing a virtual model of the remote environment at the master side, and subsequently provide the human operator with the feedback from the virtual model rather than the actual remote environment. Even though such an approach gives a solution in the case where the data flow between master and slave is delayed and unreliable, however, it requires a sufficiently detailed and precise model of the remote environment to be available. In the case of teleoperation of UAV, this may not be possible if the UAV performs a task over unknown terrain or inside confined spaces for which a detailed map is not available. Therefore, there is a need for development of a teleoperator system with the ability to build a virtual model of the remote environment in real-time while executing the task on the actual remote environment.

1.7 Thesis Contribution

The main contribution of this thesis consists of developing a new type of teleoperator system for remote control of UAVs where the haptic feedback is generated using SLAM algorithms. Specifically, SLAM algorithms are utilized in this work for the purpose of building a virtual environment on the master site of the teleoperation system. The objects in this virtual environment replicate essential features of the actual remote environment where the UAV executes its tasks. The haptic feedback is subsequently generated at the master site by building artificial potential field around obstacles in the virtual environment. Two methods for building haptic feedback from SLAM algorithms are developed. The first basic algorithm utilizes a fixed size potential field around each obstacle in virtual environment. The second proposed algorithm changes the size of the potential field around the obstacle depending on the amount of uncertainty in the obstacle's location, the latter is represented by the covariance estimate provided by EKF SLAM algorithm. It is worth to mention that, in today's scientific literature, SLAM and haptics are treated as two completely separated disciplines. The haptic technology aims to provide the human operator with haptic feedback, which makes it naturally directed towards human-centered applications, such as teleoperation, human-machine interaction, *etc.*.

The SLAM techniques, on the other hand, are currently directed to applications where the primary goal is to make the robot as autonomous and independent of the human intervention as possible. This thesis is apparently the first work where the SLAM and the haptic technologies are combined together to solve a meaningful technological problem.

1.8 Thesis Outline

The structure of the remaining part of the thesis can be described as follows:

In Chapter 2, the kinematics, dynamics, and control algorithms for a quadrotor UAV are addressed. In particular, both attitude control and position control problems are formulated, and the control strategies for both linearized and fully nonlinear models of the quadrotor are developed in detail. In the case of linearized model, PD and PID controllers are used, while the control strategies for nonlinear model are based on the integrator backstepping approach. Comprehensive mathematical derivation of the backstepping control algorithm for a quadrotor UAV is provided. In both cases of linearized and nonlinear models, simulations have been carried out using Matlab and C++ to evaluate the performance of the controllers, and the results are discussed.

In Chapter 3, the problem of Simultaneous Localization and Mapping (SLAM) is addressed from probabilistic perspective. A general form of probabilistic SLAM approach based on Bayesian framework is presented. Different types of metric maps and parametric filters are discussed. The Extended Kalman Filter (EKF) implementation of the SLAM algorithms is described in detail, and a complete EKF-SLAM algorithm for a quadrotor UAV is developed.

In Chapter 4, a teleoperator system with SLAM based haptic feedback for remote control of a quadrotor UAV is developed. A predictive display approach to teleoperation is discussed, and the structure of a teleoperator system with SLAM-based haptic feedback is

developed. An artificial potential field approach to haptic feedback is described, and semi-experimental results are provided that evaluate the influence of the stiffness and damping terms of the potential field. Two types of algorithms for SLAM based haptic feedback are proposed, where the basic algorithm uses fixed size potential field around obstacles, while the robust algorithm changes the size of potential field around the obstacle depending on the amount of uncertainty in obstacle location, which is represented by the covariance estimate provided by EKF. Semi-experimental results are presented that evaluate performance of the developed teleoperator system with SLAM based haptic feedback.

In Chapter 5, conclusions to the thesis are provided, and possible directions for future research are discussed.

Chapter 2

Kinematics, Dynamics, and Control of a Quadrotor UAV

In this Chapter, kinematics, dynamics, and control of a quadrotor aircraft are addressed. Section 2.1 gives a general introduction into quadrotor operation and shows how forces/torques generated by the rotors affect the platform's movement. Section 2.2 discusses the basics of quadrotor's kinematics. Section 2.3 focuses on the dynamic behaviour of the quadrotor; in particular, it shows how the dynamic equations can be derived using Euler-Newton approach. Section 2.4 deals with control algorithms for quadrotor. In this section, linear and nonlinear models of the quadrotor are addressed, and algorithms for attitude control and position control are derived, with special emphasis on the nonlinear control design using backstepping methods. Numerical simulations of different control algorithms are performed using Matlab and C++, and the results are presented. Conclusions are given in Section 2.5.

2.1 Basic Quadrotor Operation

The quadrotor platform is an example of VTOL UAVs with cross configuration that has four propellers connected separately to four motors which play the role of actuators. The configuration space of a quadrotor has 6 degrees-of-freedom (DOFs), which include three DOFs for

translational movement and three DOFs for rotational movement. As the number of DOFs is higher than the number of actuators, the quadrotor is an underactuated system [7]. The forces and the torques that create the translational and rotational movements of the quadrotor depend on the motors' angular velocities, which are denoted by ω_i, where $i = \{1, 2, 3, 4\}$. The i-th spinning motor generates a vertical force F_i and a torque M_i that are proportional to the motor's squared angular velocity ω_i, *i.e.*,

$$F_i = k_F \omega_i^2, \tag{2.1}$$

$$M_i = k_M \omega_i^2, \tag{2.2}$$

where k_F and k_M are constants that can be acquired through an experimental test [49], [50]. Figure 2.1 shows a top view of the platform, where the red arrows denote the rotational direc-

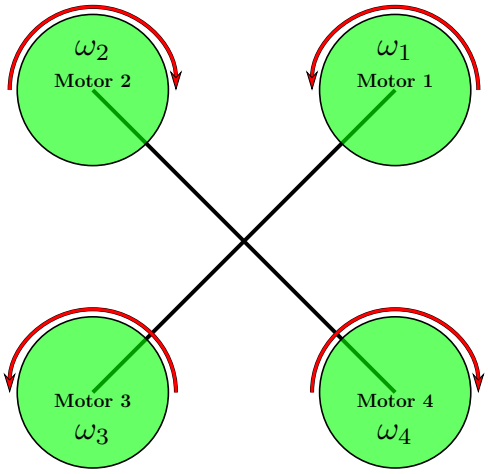


Figure 2.1: Cross configruation of the quadrotor, view from the top.

tion of propellers; specifically, Motors 1 and 3 rotate counter-clockwise whereas Motors 2 and 4 rotate clockwise. Notation **U** is the control input which can be force or torque. The **upwards thrust** $\mathbf{U}_1$ is defined as follows,

$$\mathbf{U}_1 = \sum_{i=1}^{4} F_i. \tag{2.3}$$

Notation $\mathbf{U}_1$ in (2.3) represents the total force that is responsible for lifting the quadrotor up. The pure upwards thrust is generated when all motors are rotated with the same angular velocity, see Figure 2.2. In the aforementioned figure, the axes of the body frame are defined in which the positive part of the z axis points to the ground. The **roll torque** $\mathbf{U}_2$ is the torque

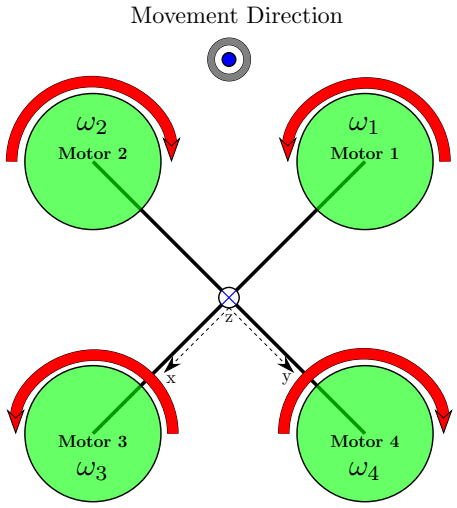


Figure 2.2: Generating the upwards thrust by increasing/decreasing all motors' velocities simultaneously with the same magnitude. The movement direction in this figure is upward.

that causes the platform to rotate about its x-axis. It is generated by increasing the angular velocity of motor 2 and decreasing the angular velocity of motor 4 for achieving the clockwise rotation (*i.e.*, the right turn), see Figure 2.3. For the counter-clockwise rotation about x-axis, the process is reversed, *i.e.*, the angular velocity of motor 4 increases and the angular velocity of motor 2 decreases. The formula for roll torque $\mathbf{U}_2$ is as follows:

$$\mathbf{U}_2 = l(F_4 - F_2), \tag{2.4}$$

where l is the distance from each rotor to the center of the quadrotor's mass. The **pitch torque** $\mathbf{U}_3$ is the torque that causes the platform to rotate about its y-axis. It is generated by increasing the angular speed of motor 3 and decreasing the angular speed of motor 1 for achieving the clockwise rotation, or *vice versa* for the counter-clockwise rotation see Figure 2.4. Torque $\mathbf{U}_3$

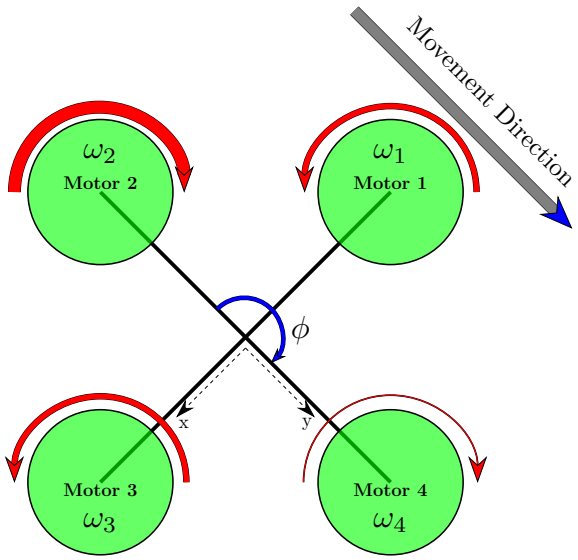


Figure 2.3: Generating roll torque. Increase in angular velocity of motor 2 and decrease in angular velocity of motor 4 while keeping the angular velocities of motors 1 and 3 lower than that of motor 2 and greater than that of motor 4 causes the platform to roll about x axis subsequently move in the direction shown.

is given as follows,

$$\mathbf{U}_3 = l(F_3 - F_1). \tag{2.5}$$

The **yaw torque** $\mathbf{U}_4$ is the torque that causes the platform to rotate about its z-axis. Whenever the angular velocities of motors 3 and 1 increase with same magnitude while the angular velocities of the other two motors decrease, a drag torque is generated that causes the platform to rotate in the direction opposite to the rotations of motors 3 and 1. Figure 2.5 shows the rotation of the platform in the clockwise direction.

$$\mathbf{U}_4 = K_w(F_3 + F_1 - F_4 - F_2), \tag{2.6}$$

where K_w is the ratio of propeller torque constant to propeller force constant [55].

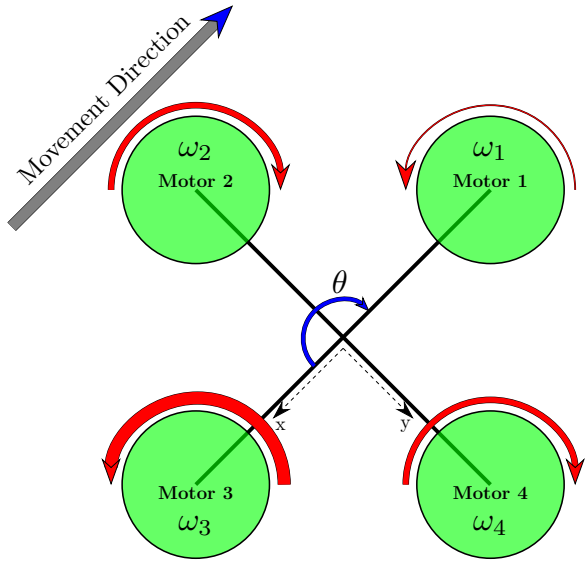


Figure 2.4: Generating the pitch torque. Increase in angular velocity of motor 3 and decrease in angular velocity of motor 1 while keeping the angular velocities of motors 2 and 4 lower than that of motor 3 and greater than that of motor 1 causes the platform to pitch about y axis subsequently move in the direction shown.

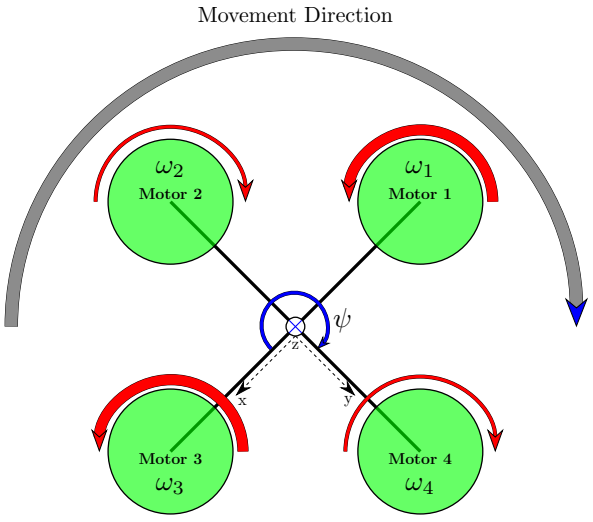
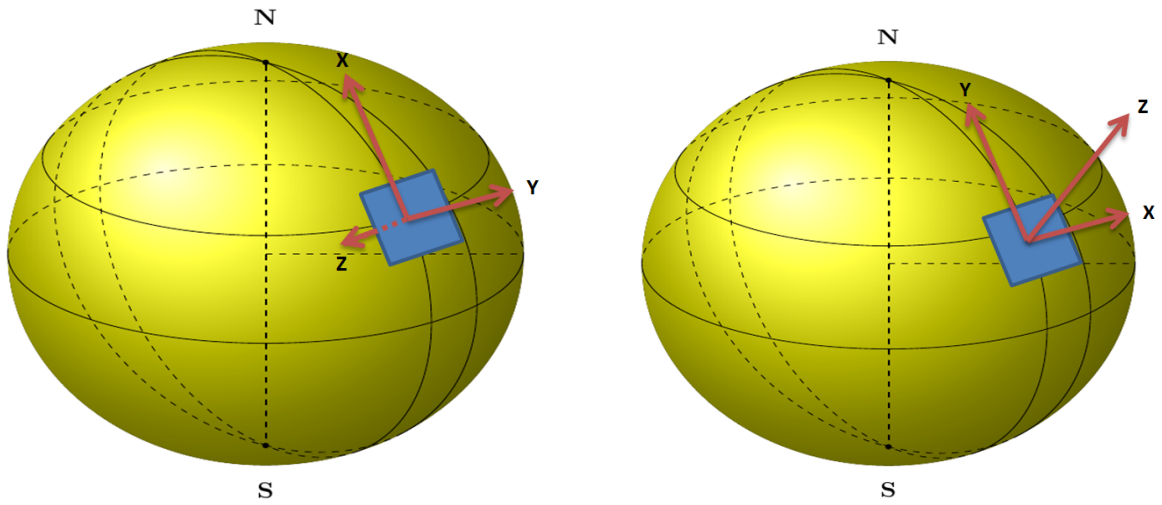


Figure 2.5: Generating yaw torque. Increase in angular velocities of motors 3 and 1 and decrease in angular velocities of motors 4 and 2 generates clockwise rotation due to drag torque.

2.2 Quadrotor Kinematics

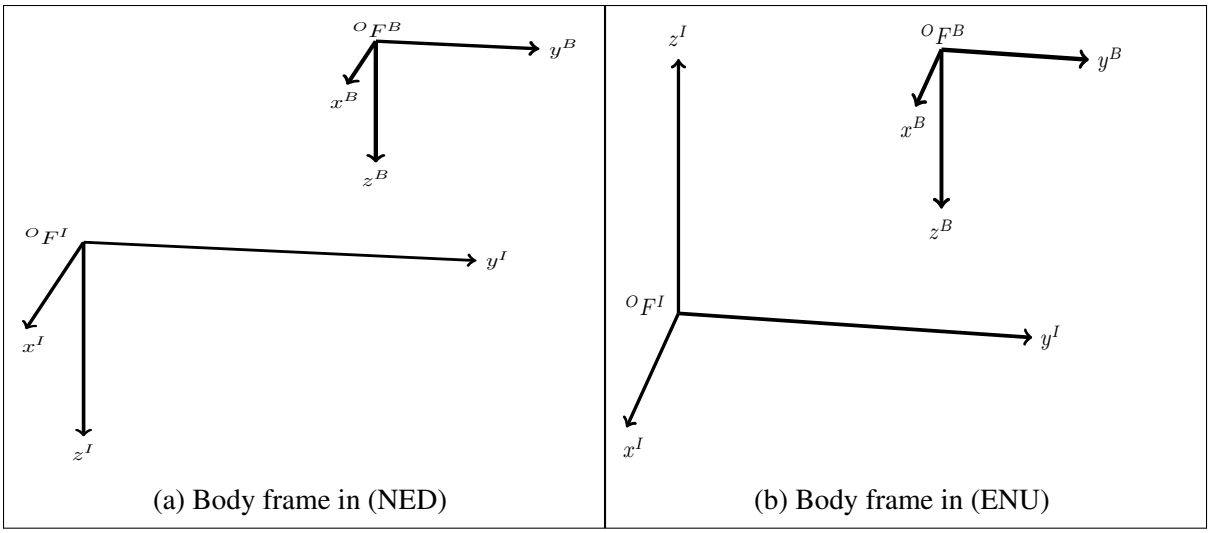
Kinematics of a rigid body describes the motion of the body without considering forces/moments that cause this motion. An assumption that the quadrotor can be represented as a rigid and symmetrical body is widely used in literature [51, 6, 52]; this assumption simplifies the model to a certain extent. For representation of position and orientation of a quadrotor's body in space, two coordinate frames are required, which are called the inertial frame and the body frame, respectively. The center of an **inertial frame** F^I is usually attached to a given point on the Earth's surface; the latter is assumed to be flat [53]. One possible configuration of the inertial frame is such that its x-, y-, and z-axes are directed towards the North, East, and the center of the Earth, respectively. This type of frame configuration is known as North-East-Down (NED) coordinate system [54]. Another possible configuration of the inertial frame is where the x-, y-, and z-axes point toward the East, North, and in the Upward direction (ENU), respectively [55]. Figure 2.6 shows the NED and the ENU coordinate systems. On the other hand, the **body**



(a) North-East-Down (NED) coordinate system (b) East-North-Up (ENU) coordinate system

Figure 2.6: Two common representations of the inertial frame under the assumption of flat Earth surface represented by the blue plane, where the golden sphere represents the Earth.

frame F^B is a frame whose origin is attached to the center of the quadrotor's mass. Figure 2.7 shows the body frame in the NED and ENU inertial frames. With this configuration, the posi-



(a) Body frame in (NED) (b) Body frame in (ENU)

Figure 2.7: The representations of the body frame in NED and ENU coordinates.

tion of the center of quadrotor's mass is specified in the inertial frame through three variables denoted as x, y and z, whereas the orientation of the quadrotor in the body frame is specified through three angles ϕ, θ and ψ, that describe the rotation around x^B, y^B, and z^B axes, respectively. Orientation of a rigid body in 3D space can be carried out using various methods. The most common and widely used approach is based on **Euler angles** [56, 54, 57]. This method describes orientation of a rigid body in 3D space by specifying three angles known as **yaw** (ψ), **pitch** (θ) and **roll** (ϕ).

2.3 Quadrotor Dynamics

The dynamics of a rigid body describe the relationship between the motion of the body in space and the forces/torques that cause this motion. In the case of the quadrotor system, the primary forces (*i.e.*, F_1, F_2, F_3, and F_4) that cause the quadrotor to fly are generated by the four motors. The i-th spinning rotor contributes to the whole upwards thrust by generating the vertical thrust F_i and the drag torque M_i, which are given by the following formulas [58]:

$$F_i = C_F \rho A \omega_i^2 R^2, \tag{2.7}$$

$$M_i = C_M \rho A \omega_i^2 R^3, \tag{2.8}$$

where $C_F > 0$, $C_M > 0$, $\rho > 0$, $R > 0$, and $A > 0$ are thrust coefficient, torque coefficient, the air density, the rotor radius, and the area of the rotor disc, respectively. A spinning rotor can be considered as a rotating disk that interacts with air in its vicinity [59]; because of this interaction, difference in air pressure between the top and the bottom parts of the rotating disk is created which makes the air flows in a streamtube manner. Modelling the rotor as an actuator disk allows for applying the momentum theory. This theory determines the relationship between the upwards thrust that a propeller generates, the induced velocity, and the induced power of the rotor via Bernoulli's equation [59]. The dynamics of the quadrotor are also sub-

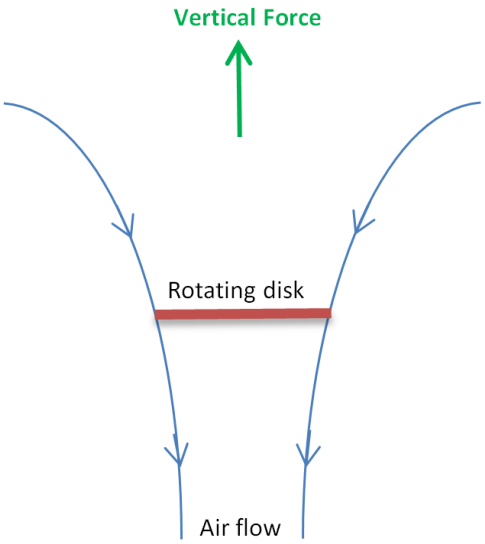


Figure 2.8: Modeling the rotor as a rotating disk; the figure shows the direction of the air flow through the rotor.

ject to external forces that impede quadrotor's motion. For the sake of simplicity, it is assumed throughout this thesis that the only external force that acts on the quadrotor's body is the gravitational force F_g. The dynamics of the quadrotor are a combination of the translational and the rotational dynamics [7], in which the translational dynamics depend on the rotational dynamics but not *vice versa*, as will be shown below.

The two most common approaches for deriving dynamical equations of the quadrotor are based on the Euler-Lagrange and the Newton-Euler methods. The Newton-Euler approach

applies Newton's laws directly to derive the mathematical model of the quadrotor by considering all forces/torques that cause the motion. The general form of the dynamics equations of an aerial vehicle under the influence of external forces/torques can be expressed through the following equations:

$$\dot{\Upsilon} = \boldsymbol{V}, \tag{2.9}$$

$$m\dot{\boldsymbol{V}} = F_{\Upsilon}, \tag{2.10}$$

$$\dot{R} = R\hat{\Omega}, \tag{2.11}$$

$$\mathbb{J}\dot{\Omega} = -\Omega \times \mathbb{J}\Omega + M_{\eta}, \tag{2.12}$$

where $\boldsymbol{\Upsilon} := (x, y, z) \in \mathbb{R}^3$ is the position of the center of quadrotor's mass and $\boldsymbol{V} \in \mathbb{R}^3$ is the linear velocity of the center of the quadrotor's mass expressed in the inertial frame, $F_{\Upsilon} \in \mathbb{R}^3$ and $M_{\eta} \in \mathbb{R}^3$ are the vectors of translational forces and rotational moments, respectively, $\Omega \in \mathbb{R}^3$ is the angular velocity expressed in the body frame, $\hat{\Omega} \in \mathbb{R}^{3\times3}$ is a skew symmetric matrix which is defined as follows,

$$\hat{\Omega} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix}$$

and $R \in SO(3)$ is the orthogonal rotation matrix.

The mathematical model of the quadrotor in this thesis is adopted from [55]; according to this work, the dynamics of the quadrotor are described by the following nonlinear equations:

$$\ddot{x} = \frac{\mathrm{U}_1}{m}(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi), \tag{2.13}$$

$$\ddot{y} = \frac{\mathrm{U}_1}{m}(\cos\phi \sin\theta \sin\psi - \sin\phi \sin\psi), \tag{2.14}$$

$$\ddot{z} = \frac{\mathrm{U}_1}{m}(\cos\phi \cos\theta) - g, \tag{2.15}$$

$$\ddot{\phi} = \frac{J_y - J_z}{J_x}\dot{\theta}\dot{\psi} + \frac{l}{J_x}\mathrm{U}_2, \tag{2.16}$$

$$\ddot{\theta} = \frac{J_z - J_x}{J_y}\dot{\phi}\dot{\psi} + \frac{l}{J_y}\mathrm{U}_3, \tag{2.17}$$

$$\ddot{\psi} = \frac{J_x - J_y}{J_z}\dot{\phi}\dot{\theta} + \frac{1}{J_z}\mathrm{U}_4. \tag{2.18}$$

where the parameters of the quadrotor are given in the following Table [55]

Parameter	Description	Value	Unit
l	Distance from pivot to rotor	0.25	m
m	The quadrotor's mass	0.75	kg
J_x	Inertia moment about x-axis	0.019688	kgm^2
J_y	Inertia moment about y-axis	0.019681	kgm^2
J_z	Inertia moment about z-axis	0.03938	kgm^2

2.4 Quadrotor Control Strategy

The quadrotor is a typical example of a substantially nonlinear dynamical system. Also, it is an inherently unstable system, which makes it even more difficult to design an appropriate control strategy. In the literature, numerous control algorithms have been proposed that aim to stabilize the quadrotor system. Some controllers are designed based on a linearized model of the platform, whereas others are designed to handle the nonlinear model. A linearized model of the platform, which is widely used in the literature [51, 50, 49, 60], is based on the assumption that the quadrotor does not perform aggressive manoeuvres. Specifically, the Euler angles are assumed to be small, which leads to a near hovering status. The quadrotor control algorithms for both models (*i.e.*, linear and nonlinear) can be divided into two classes, namely the **attitude** control and the **position** control. In the **attitude control** algorithms, the goal is to control the quadrotor's orientation; in a more general setting (adopted in this thesis) the attitude control problem also includes the **altitude** control. Figure 2.9 shows a general block diagram for the attitude control in which ϕ_d, θ_d and ψ_d are the desired roll, pitch, and yaw angles, respectively, whereas z_d is the desired altitude. In **position control**, one seeks to control the translational position of the center of the platform's mass, therefore given a spatial trajectory in which the desired translational position x_d, y_d and z_d are provided, the platform should follow it in a stable

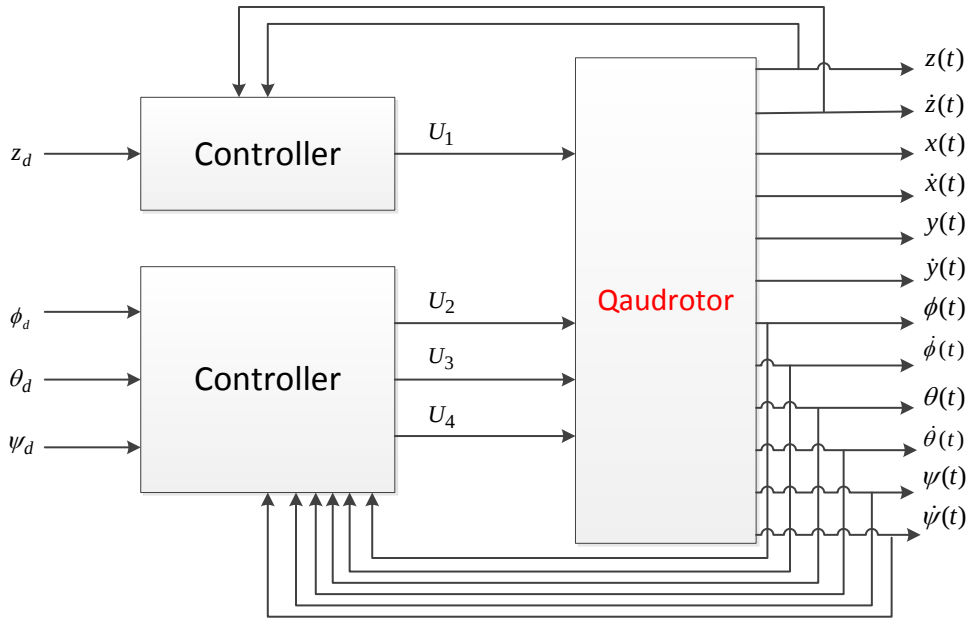


Figure 2.9: General attitude control scheme, where $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$, and $\mathbf{U}_4$ are the control inputs.

manner. Position control is more difficult than the attitude control due to the fact that the system is underactuated. In literature, the common approach for position control is implementing two control loops called outer and inner loops instead of just one control loop as in the attitude control case. Figure 2.10 shows a general block diagram for position control, in which x_d and y_d represent the desired position in $x-y$ plane, respectively. The red blocks define the outer loop whereas the blue one defines the inner loop. The popular controllers for stabilizing the linearized model of the platform include PD, LQR, PID controllers, among others.

2.4.1 Control Strategy for Linear Model

Using small angle assumption (*i.e.*, assuming the quadrotor is approximately in the hovering state), the following approximations [60] can be used to linearize the platform's model,

$$\mathbf{U}_1 \approx mg,$$

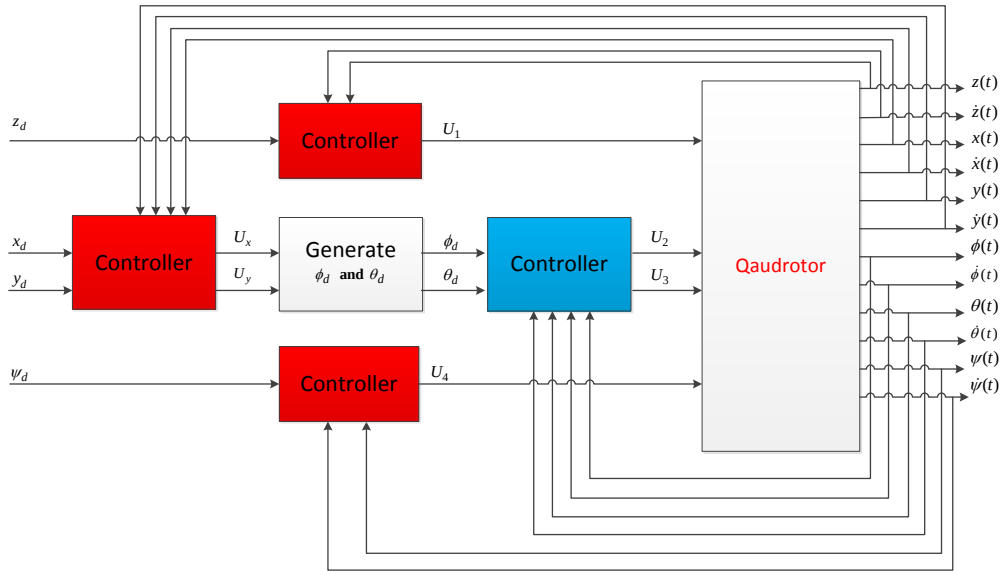


Figure 2.10: General position control scheme, where $\mathbf{U}_x$ and $\mathbf{U}_y$ are the virtual inputs that control position on the $x-y$ plane.

$$\cos\varpi \approx 1,$$

$$\sin\varpi \approx \varpi.$$

Consequently, the nonlinear model (2.13)-(2.18) can be reduced to the following linearized model,

$$\ddot{x} = g\theta, \tag{2.19}$$

$$\ddot{y} = -g\phi, \tag{2.20}$$

$$\ddot{z} = \frac{\mathrm{U}_1}{m} - g, \tag{2.21}$$

$$\ddot{\phi} = \frac{L}{J_x}\mathrm{U}_2, \tag{2.22}$$

$$\ddot{\theta} = \frac{L}{J_y}\mathrm{U}_3, \tag{2.23}$$

$$\ddot{\psi} = \frac{1}{J_z}\mathrm{U}_4. \tag{2.24}$$

Attitude Control for Linear Model

In the **attitude regulation** problem, the desired roll ϕ_d, pitch θ_d, and yaw ψ_d angles, as well as the desired altitude z_d are assumed constant, and the goal is to achieve asymptotic convergence of the actual variables ϕ, θ, ψ and z to their desired constant values, and the convergence of the corresponding velocities to zero. Mathematically, the goal is to guarantee that $\phi(t) \to \phi_d$, $\theta(t) \to \theta_d$, $\psi(t) \to \psi_d$, $z(t) \to z$, and $\dot{\phi}(t) \to 0$, $\dot{\theta}(t) \to 0$, $\dot{\psi}(t) \to 0$, $\dot{z}(t) \to 0$ as $t \to \infty$. Using PD controller, the control inputs for the attitude control can be specified as follows

$$\mathbf{U}_1 = m\left(g + K_p^z(z_d - z) + K_d^z\frac{d}{dt}(z_d - z)\right), \tag{2.25}$$

$$\mathbf{U}_2 = K_p^\phi(\phi_d - \phi) + K_d^\phi\frac{d}{dt}(\phi_d - \phi), \tag{2.26}$$

$$\mathbf{U}_3 = K_p^\theta(\theta_d - \theta) + K_d^\theta\frac{d}{dt}(\theta_d - \theta), \tag{2.27}$$

$$\mathbf{U}_4 = K_p^\psi(\psi_d - \psi) + K_d^\psi\frac{d}{dt}(\psi_d - \psi), \tag{2.28}$$

where $K_p^\varrho > 0$ and $K_d^\varrho > 0$ are the proportional and derivative gains, respectively, and $\varrho \in \{z, \phi, \theta, \psi\}$. In order to illustrate the performance of the PD controller in the attitude regulation problem, the closed-loop system (2.19)-(2.28) was simulated in Matlab. Table 2.1 presents numerical values of the parameters used in the simulations, while Figures 2.11, 2.12, and 2.13 show the response of the quadrotor.

	initial values	Gains	Desired Trajectory
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$K_p^z = 12.34$ $K_d^z = 5.67$	$z_d = 5$ (m)
Roll	$\phi(0) = \frac{\pi}{4}$ $\dot{\phi}(0) = 0$	$K_p^\phi = 11.2$ $K_d^\phi = 4.2$	$\phi_d = 0$ (rad)
Pitch	$\theta(0) = \frac{\pi}{4}$ $\dot{\theta}(0) = 0$	$K_p^\theta = 9.7$ $K_d^\theta = 3.6$	$\theta_d = 0$ (rad)
Yaw	$\psi(0) = \frac{\pi}{3}$ $\dot{\psi}(0) = 0$	$K_p^\psi = 10.78$ $K_d^\psi = 5.2$	$\psi_d = 0$ (rad)

Table 2.1: PD controller parameters for the attitude regulation problem

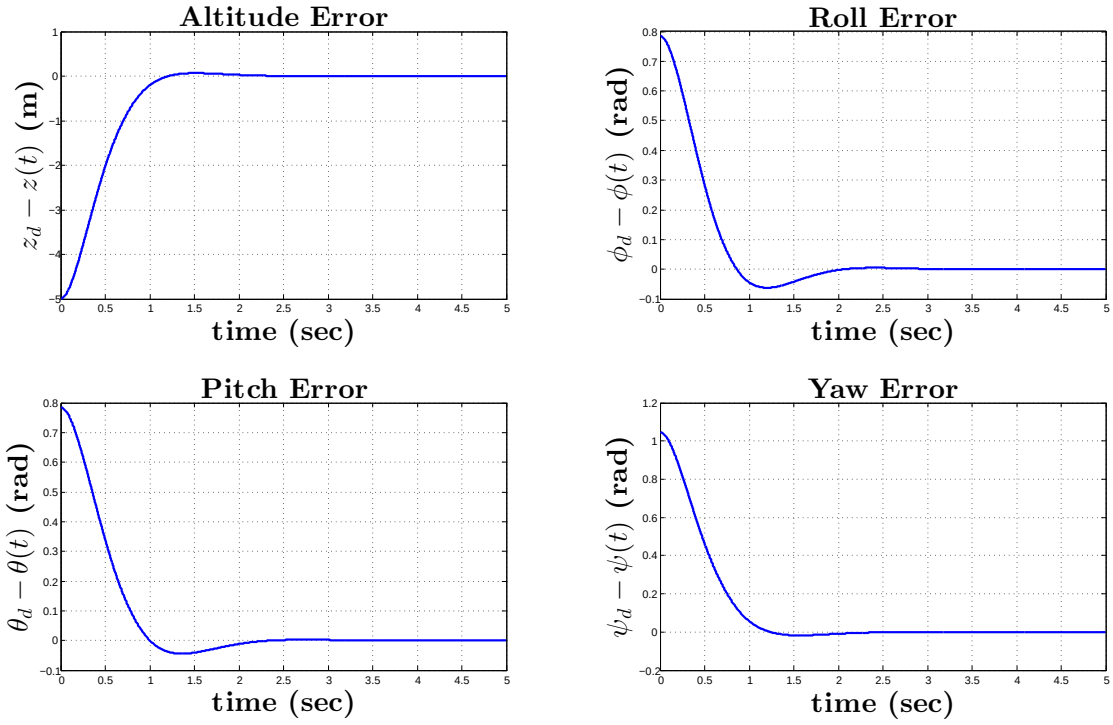


Figure 2.11: Attitude regulation problem: altitude and attitude errors for PD controller.

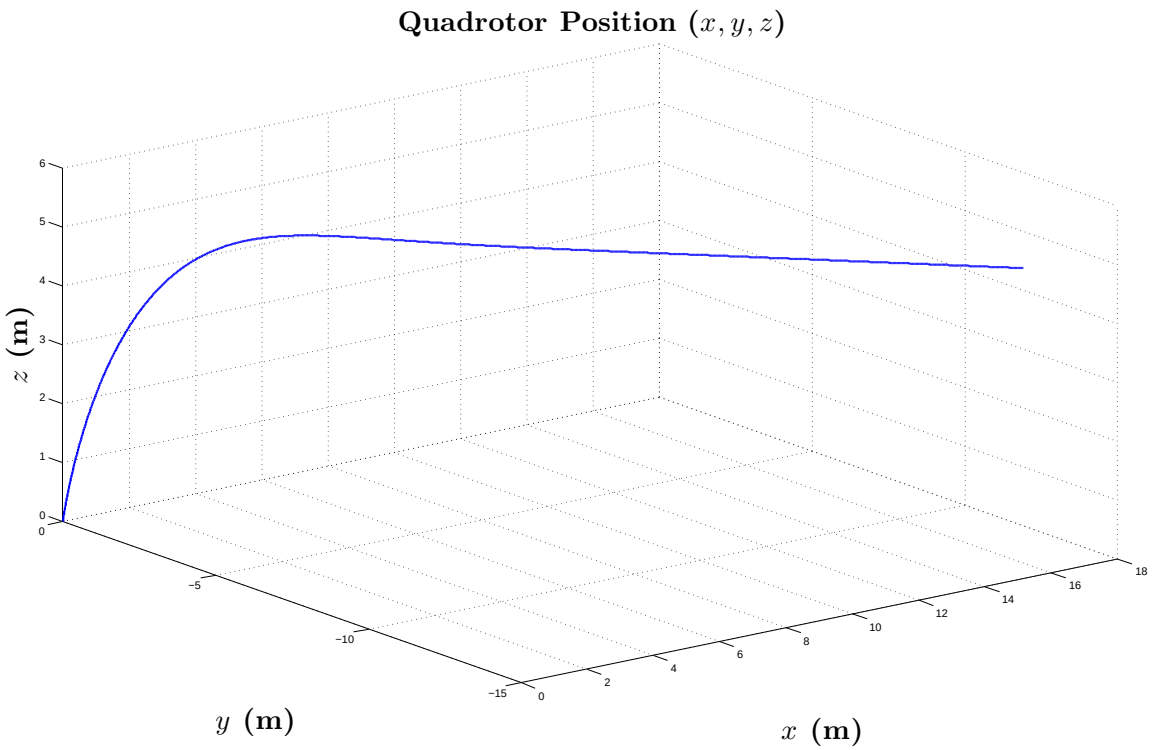


Figure 2.12: Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PD controller.

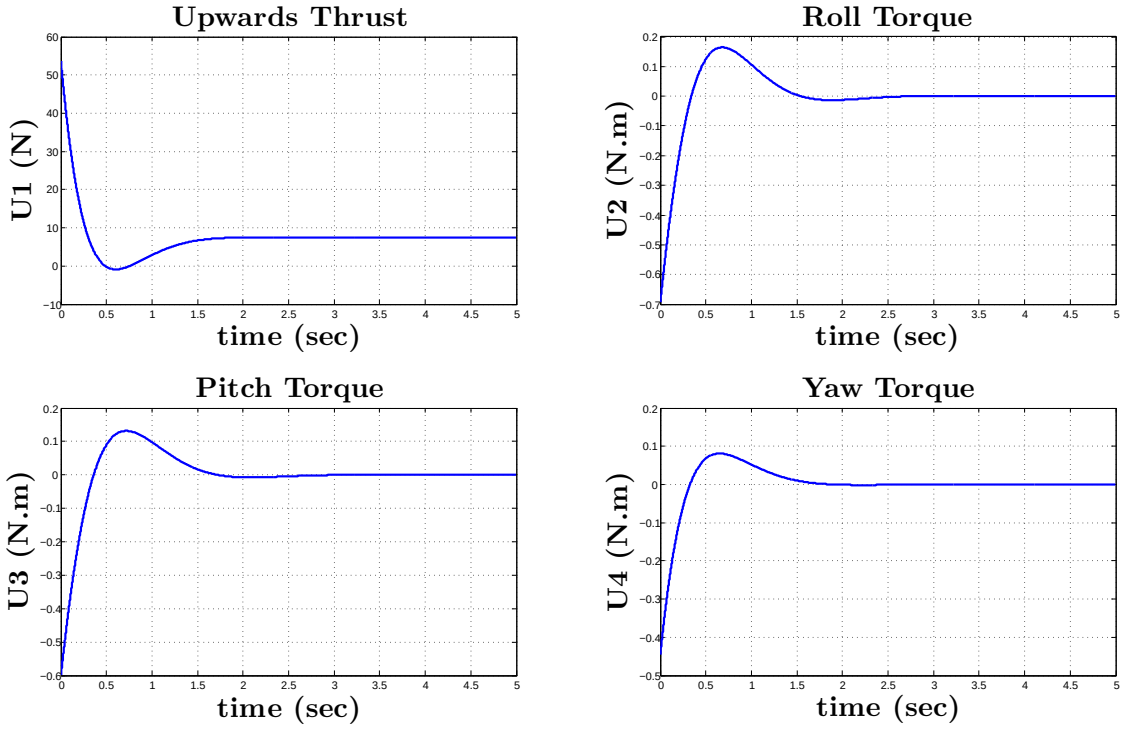


Figure 2.13: Attitude regulation problem: magnitude of the control inputs for PD controller.

In PID controller, a new term (*i.e.*, the integral term) is added to the PD controller, therefore the control inputs can be specified as follows

$$\mathbf{U}_1 = m \cdot \left(g + K_p^z(z_d - z) + K_d^z \frac{d}{dt}(z_d - z) + K_i^z \int_0^t (z_d - z(\tau))d\tau \right), \tag{2.29}$$

$$\mathbf{U}_2 = K_p^\phi(\phi_d - \phi) + K_d^\phi \frac{d}{dt}(\phi_d - \phi) + K_i^\phi \int_0^t (\phi_d - \phi(\tau))d\tau, \tag{2.30}$$

$$\mathbf{U}_3 = K_p^\theta(\theta_d - \theta) + K_d^\theta \frac{d}{dt}(\theta_d - \theta) + K_i^\theta \int_0^t (\theta_d - \theta(\tau))d\tau, \tag{2.31}$$

$$\mathbf{U}_4 = K_p^\psi(\psi_d - \psi) + K_d^\psi \frac{d}{dt}(\psi_d - \psi) + K_i^\psi \int_0^t (\psi_d - \psi(\tau))d\tau, \tag{2.32}$$

where $K_i^{(\cdot)}$ are the integral gains. Again, the system (2.19)-(2.24), (2.29)-(2.32) was simulated in Matlab in order to illustrate the performance of the PID controller in the attitude regulation problem. Table 2.2 shows the numerical values of the parameters for these simulations, and Figures 2.14, 2.15, and 2.16 show the resulting behavior of the quadrotor.

	initial values	Gains	Desired Trajectory
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$K_p^z = 11.5$ $K_d^z = 6.2$ $K_i^z = 0.0001$	$z_d = 7$ (m)
Roll	$\phi(0) = \frac{\pi}{9}$ $\dot{\phi}(0) = 0$	$K_p^\phi = 13.22$ $K_d^\phi = 10.01$ $K_i^\phi = 0.0075$	$\phi_d = 0$ (rad)
Pitch	$\theta(0) = \frac{\pi}{7}$ $\dot{\theta}(0) = 0$	$K_p^\theta = 12.63$ $K_d^\theta = 9.1$ $K_i^\theta = 0.007$	$\theta_d = 0$ (rad)
Yaw	$\psi(0) = \frac{\pi}{12}$ $\dot{\psi}(0) = 0$	$K_p^\psi = 13.58$ $K_d^\psi = 8.2$ $K_i^\psi = 0.008$	$\psi_d = 0$ (rad)

Table 2.2: PID controller parameters for the attitude regulation problem

Position Control for the Linear Model

It can be noticed from equations (2.19)-(2.20) that the linear accelerations $\ddot{x}$ and $\ddot{y}$ can not be controlled directly. However, the aforementioned linear accelerations depend on the roll and pitch angles. One can exploit this fact to control x and y indirectly through appropriately designed desired (reference) trajectories for roll $\phi_d(t)$ and pitch $\theta_d(t)$ angles [50]. Specifically, let the desired pitch and roll angles be defined as follows,

$$\theta_d := \frac{\mathbf{U_x}}{g}, \tag{2.33}$$

$$\phi_d := -\frac{\mathbf{U_y}}{g}, \tag{2.34}$$

where $\mathbf{U}_x$ and $\mathbf{U}_y$ are virtual control inputs for x and y linear positions, respectively. Assuming $\theta = \theta_d$, $\phi = \phi_d$, one sees from (2.19), (2.20) that

$$\ddot{x} = \mathbf{U_x},$$

$$\ddot{y} = \mathbf{U_y}.$$

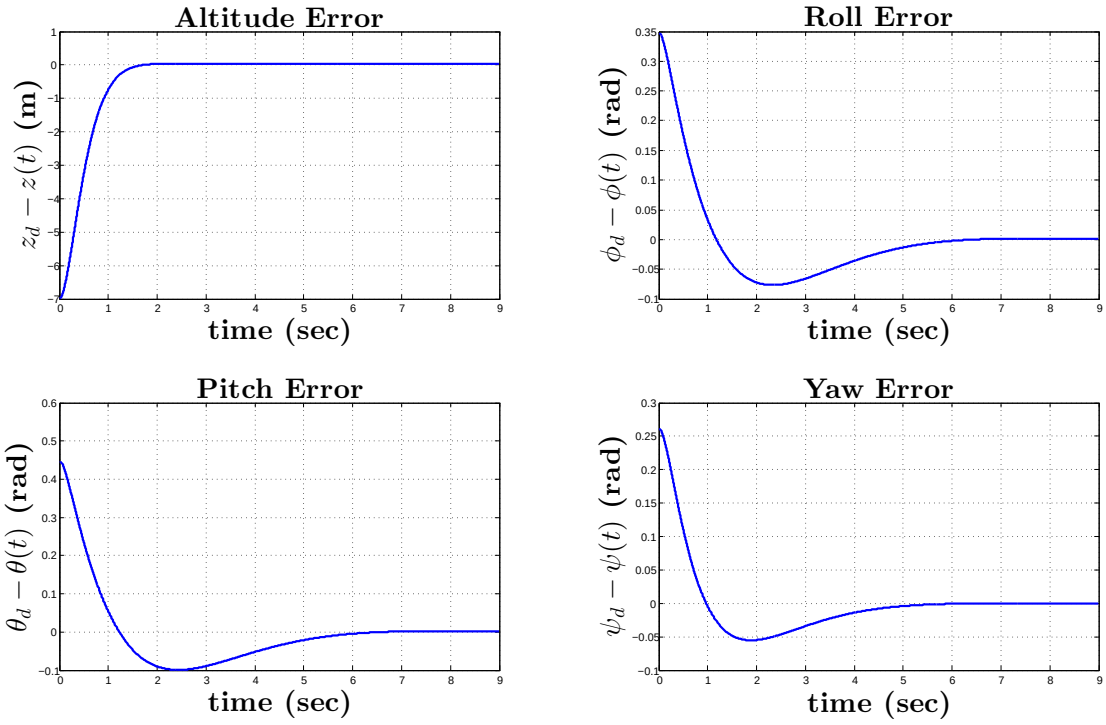


Figure 2.14: Attitude regulation problem: altitude and attitude errors for PID controller.

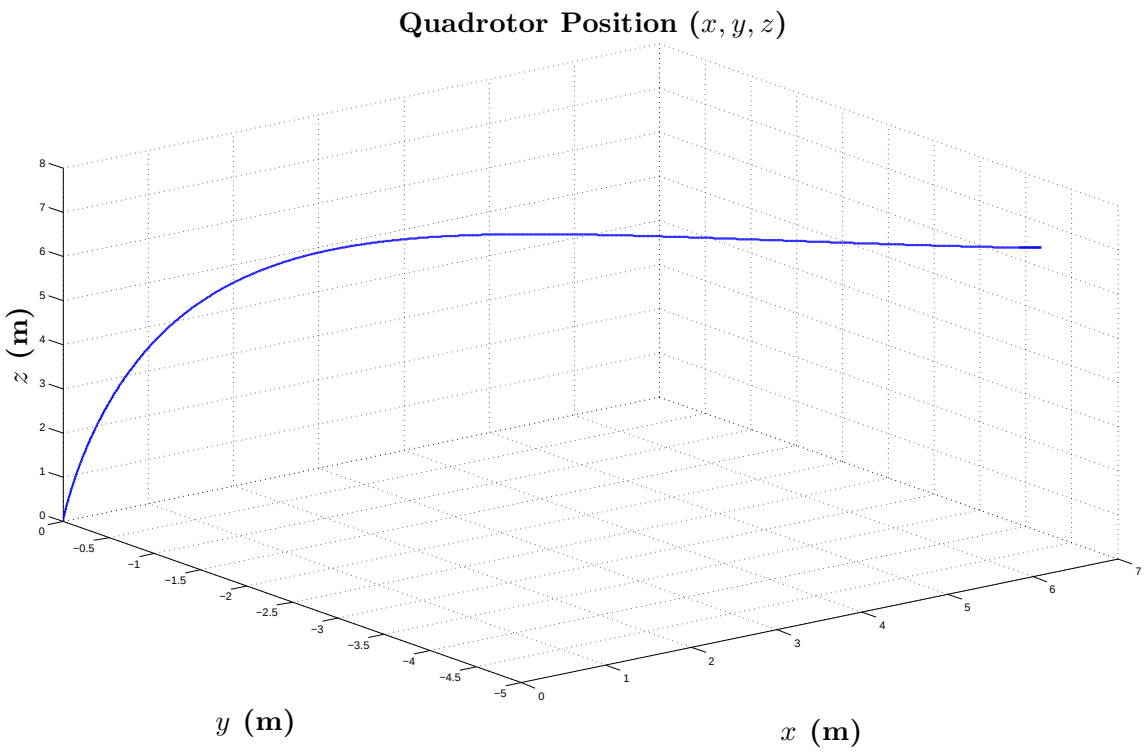


Figure 2.15: Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PID controller.

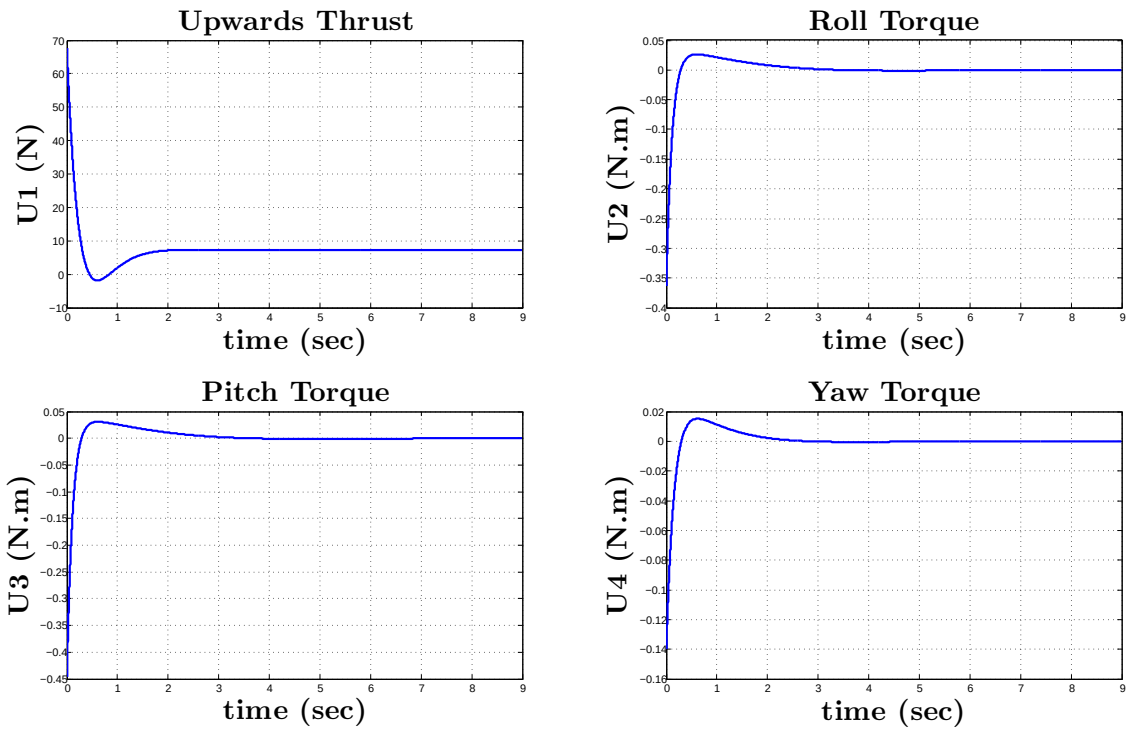


Figure 2.16: Attitude regulation problem: the magnitude of the control inputs for the PID control.

The virtual control inputs $\mathbf{U}_x$ and $\mathbf{U}_y$ can now be defined as follows,

$$\mathbf{U}_x = K_p^x(x_d - x) + K_d^x \frac{d}{dt}(x_d - x), \tag{2.35}$$

$$\mathbf{U}_y = K_p^y(y_d - y) + K_d^y \frac{d}{dt}(y_d - y). \tag{2.36}$$

In Matlab, simulations have been carried out to illustrate the performance of the PD controller. Table 2.3 shows the numerical values of the parameters for the simulations and Figures 2.17, 2.18, and 2.19 show the response of the quadrotor.

	initial values	Gains	Desired Value
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$K_p^z = 14$ $K_d^z = 7$	$z_d = 2.5$ (m)
x-Position	$x(0) = 0$ $\dot{x}(0) = 0$	$K_p^x = 10$ $K_d^x = 5$	$x_d = 2.5$ (m)
y-Position	$y(0) = 0$ $\dot{y}(0) = 0$	$K_p^y = 10$ $K_d^y = 5$	$y_d = 2.5$ (m)
Yaw	$\psi(0) = \frac{\pi}{3}$ $\dot{\psi}(0) = 0$	$K_p^\psi = 25$ $K_d^\psi = 10$	$\psi_d = 0$ (rad)
Roll	$\phi(0) = 0$ $\dot{\phi}(0) = 0$	$K_p^\phi = 21.3$ $K_d^\phi = 12.72$	
Pitch	$\theta(0) = 0$ $\dot{\theta}(0) = 0$	$K_p^\theta = 19.5$ $K_d^\theta = 9.7$	

Table 2.3: PD controller parameters for position regulation problem

2.4.2 Control Strategy for Nonlinear Quadrotor Model

In the literature, a number of nonlinear controllers are described that successfully stabilize position and attitude of the quadrotor. For example, the Lyapunov-based backstepping control is one of the most common approaches for stabilizing the quadrotor. It is has been used to yield satisfactory results in [61, 55, 62, 63]. Another common nonlinear control approach is the sliding mode control which is based on the variable structure control (VSC) principle. VSC modifies the system's structure through switching control to keep the system's trajectory on a

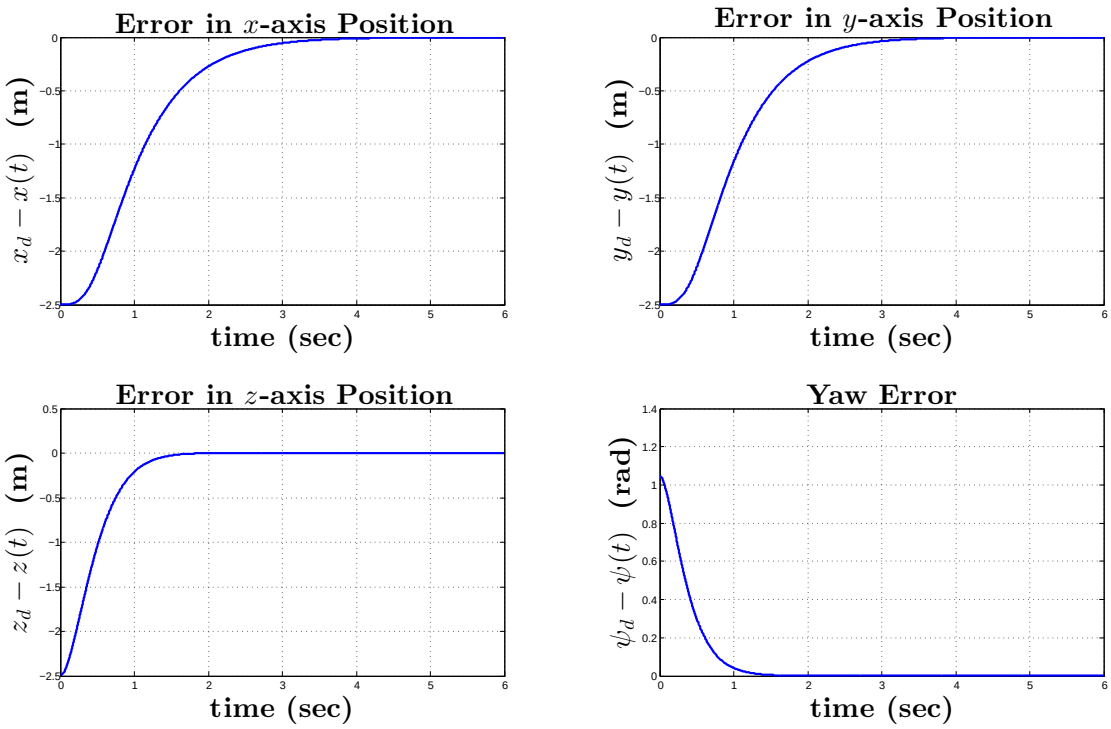


Figure 2.17: Position regulation problem: errors of linear positions and yaw angle for PD controller.

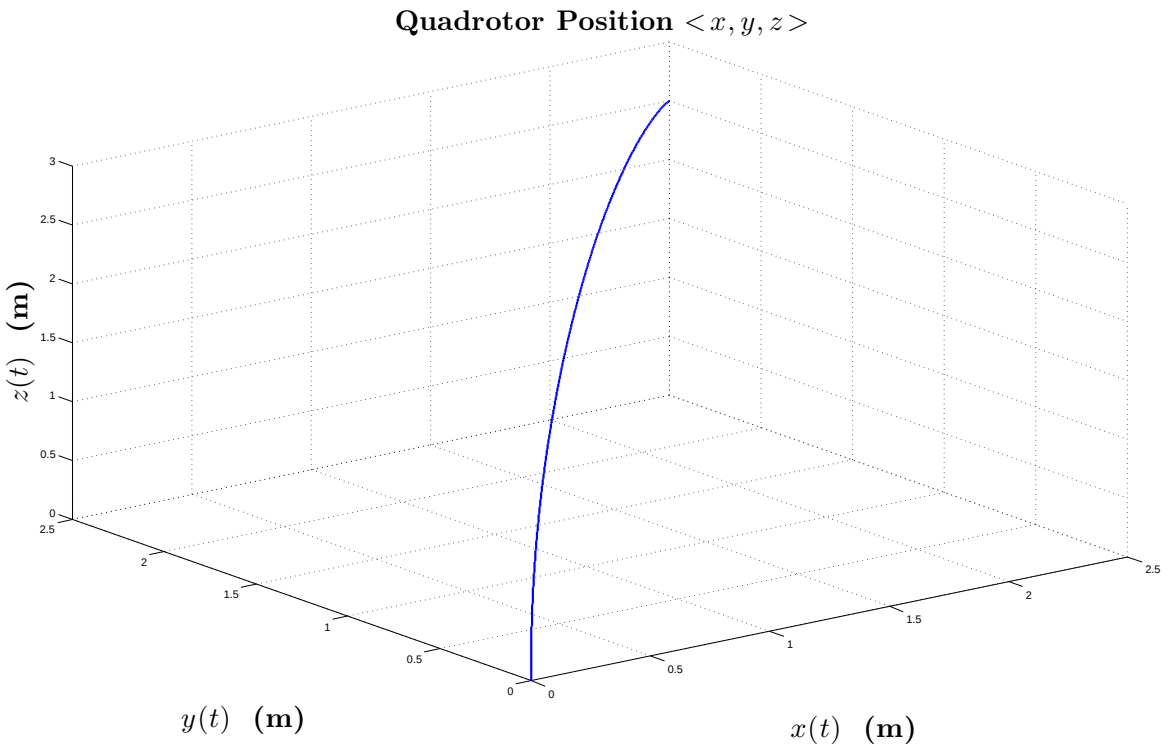


Figure 2.18: Position regulation problem: the position (x, y, z) of the center of the quadrotor's mass for PD controller.

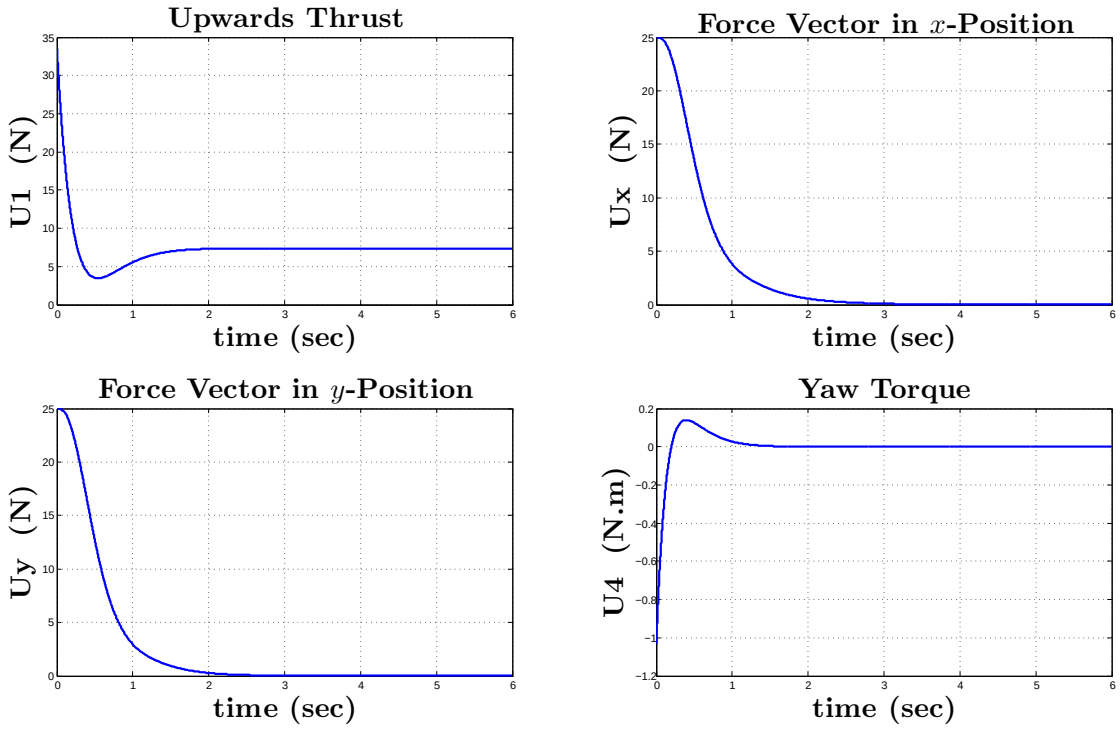


Figure 2.19: Position regulation problem: the magnitude of the control inputs for PD controller.

predetermined surface in the state space called the sliding surface [64, 65]. In [66, 61, 67, 55], the sliding mode controller has been used for quadrotor control. Feedback linearization controllers are another class of controllers that target nonlinear systems by cancelling all substantial nonlinearties; as a result, the closed-loop system becomes linear and therefore can be controlled using linear methods. In [68, 69], the feedback linearization approach has been used to stabilize the quadrotor. In this thesis, we concentrate on the Lyapunov-based backstepping control method, therefore the following sections deal with control algorithms that are based on the backstepping approach.

2.4.3 Backstepping Controller

In [70], which is currently the most comprehensive book that deals with backstepping control methods, the backstepping is defined as an approach ***"to design a controller recursively by considering some of the state variables as "virtual controls" and designing for them intermediate control laws"***[70]. The word "backstepping" comes from the fact that it is essentially a step-by-step design approach, where at each step the control algorithm propagates through one integrator back to the control input [71]. At each step, the control input is designed in such a way that makes the time derivative of a given Lyapunov function negative definite, which leads to stability of the system in the sense of Lyapunov. It is worth to mention that Lyapunov-based backstepping is different from the feedback linearization methodology. The latter approach eliminates all nonlinearities in the system, so that the closed-loop system becomes linear [70]. Canceling all the nonlinearies requires an accurate description of the mathematical model which may be hard to achieve in practice [72]. Moreover, the feedback linearization approach does not distinguish between "good" and "harm" nonlinearities [70], therefore all nonlinearies are treated equally in the sense that they all are considered destabilizing which may not always be the case. On the other hand, the Lyapunov-based backstepping approach is more flexible than the feedback linearization in that the "good" nonlinearies are retained while the "harm" ones are eliminated.

2.4.4 Attitude Control for Nonlinear Model

Design of Upwards Thrust Control Input $\mathbf{U}_1$

In this section, the control input $\mathbf{U}_1$ is derived according to the step-by-step backstepping procedure [71, 73], as follows.

Step one. Let a_1 define the altitude tracking error, as follows,

$$a_1 := z_d(t) - z(t), \tag{2.37}$$

where $z_d(t)$ and $z(t)$ are the desired and the actual altitudes, respectively. From now on, the independent time variable will be omitted for simplicity of notation. Let a Lyapunov function $V(a_1)$ be defined to be a positive definite function of a_1, as follows,

$$V(a_1) = \frac{1}{2}a_1^2. \tag{2.38}$$

Taking into account (2.37), the time derivative of Lyapunov function (2.38) is

$$\dot{V}(a_1) = a_1(\dot{z}_d - \dot{z}). \tag{2.39}$$

To make the expression (2.39) bounded from above by some negative definite function, we would like to guarantee that $\dot{V}(a_1)$ satisfies the following inequality,

$$\dot{V}(a_1) \leq -\kappa_1 a_1^2, \tag{2.40}$$

where $\kappa_1 > 0$ is a constant. Using (2.39), inequality (2.40) becomes

$$a_1(\dot{z}_d - \dot{z}) \leq -\kappa_1 a_1^2. \tag{2.41}$$

Inequality (2.41) is guaranteed if the state variable $\dot{z}$ satisfies

$$\dot{z} = \dot{z}_d + \kappa_1 a_1. \tag{2.42}$$

According to the basic ideas of the backstepping procedure [70], the state variable $\dot{z}$ can be

used as a virtual control for the purpose of stabilization. To this end, define

$$z_{vir} := \dot{z}_d + \kappa_1 a_1. \tag{2.43}$$

The above defined variable z_{vir} represents the desired value of the state variable $\dot{z}$, *i.e.*, the value of $\dot{z}$ that would guarantee the exponential convergence $a_1(t) \to 0$.

Step two. Let a new variable a_2 be defined as the difference between the state variable $\dot{z}$ and the virtual control input z_{vir}, as follows,

$$a_2 := \dot{z} - z_{vir}. \tag{2.44}$$

Using (2.44), the expression (2.39) for $\dot{V}(a_1)$ can be rewritten to yield the following formula

$$\begin{aligned}
\dot{V}(a_1) &= a_1(\dot{z}_d - \dot{z}) = a_1(\dot{z}_d - (a_2 + z_{\boldsymbol{vir}})) \\
&= a_1(\dot{z}_d - (a_2 + \dot{z}_d + \kappa_1 a_1)) = \cancel{a_1\dot{z}_d} - a_1 a_2 - \cancel{a_1\dot{z}_d} - \kappa_1 a_1^2 \\
&= -a_1 a_2 - \kappa_1 a_1^2.
\end{aligned} \tag{2.45}$$

Let us now augment the Lyapunov function (2.38) with an additional term $V(a_2) := \frac{1}{2}a_2^2$ which accounts for the dynamics of variable a_2. The total Lyapunov function $V(a_1, a_2)$ has a form

$$\begin{aligned}
V(a_1, a_2) := V(a_1) + V(a_2) &= \frac{1}{2}a_1^2 + \frac{1}{2}a_2^2 \\
&= \frac{1}{2}(z_d - z)^2 + \frac{1}{2}(\dot{z} - z_{\boldsymbol{vir}})^2 = \frac{1}{2}(z_d^2 - 2zz_d + z^2) + \frac{1}{2}(\dot{z}^2 - 2\dot{z}z_{vir} + z_{vir}^2) \\
&= \frac{1}{2}(z_d^2 - 2zz_d + z^2) + \frac{1}{2}\dot{z}^2 - \dot{z}(\dot{z}_d + \kappa_1 z_d - \kappa_1 z) \\
&\quad + \frac{1}{2}(\dot{z}_d^2 + 2\kappa_1 \dot{z}_d z_d - 2\kappa_1 \dot{z} + \kappa_1^2 z_d^2 - 2\kappa_1^2 z_d z + \kappa_1^2 z^2)
\end{aligned} \tag{2.46}$$

The time derivative of the total Lyapunov function $V(a_1, a_2)$ is given by the following expression

$$\dot{V}(a_1, a_2) = (z_d - z - \kappa_1 \dot{z} + \kappa_1 \dot{z}_d + \kappa_1^2 z_d - \kappa_1^2 z)\dot{z}_d$$

$$
\begin{aligned}
&+ (-z_d + z + \kappa_1 \dot{z} - \kappa_1 \dot{z}_d - \kappa_1^2 z_d + \kappa_1^2 z)\dot{z} \\
&+ (-\dot{z} + \dot{z}_d + \kappa_1 z_d - \kappa_1 z)\ddot{z}_d \\
&+ (\dot{z} - \dot{z}_d - \kappa_1 z_d + \kappa_1 z)\ddot{z}.
\end{aligned} \tag{2.47}
$$

By rearranging equation (2.47), one obtains

$$
\dot{V}(a_1, a_2) = a_2 \ddot{z} - a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - a_1 a_2 - \kappa_1 a_1^2. \tag{2.48}
$$

The exponential convergence $a_1(t)$ and $a_2(t)$ to zero will be guaranteed if the time derivative of the total Lyapunov function $V(a_1, a_2)$ satisfies the inequality

$$
\dot{V}(a_1, a_2) \leq -\kappa_1 a_1^2 - \kappa_2 a_2^2, \tag{2.49}
$$

where $\kappa_1, \kappa_2 > 0$. Taking into account (2.48), inequality (2.49) becomes

$$
a_2 \ddot{z} - a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - a_1 a_2 \cancel{- \kappa_1 a_1^2} \leq \cancel{-\kappa_1 a_1^2} - \kappa_2 a_2^2,
$$

or

$$
a_2 \ddot{z} - a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - a_1 a_2 \leq -\kappa_2 a_2^2. \tag{2.50}
$$

The last inequality (2.50) is satisfied if the following relation holds

$$
\ddot{z} = a_1 + (\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - \kappa_2 a_2. \tag{2.51}
$$

Indeed, substituting (2.51) into (2.50), one obtains

$$
\begin{aligned}
&a_2[a_1 + (\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - \kappa_2 a_2) - a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - a_1 a_2 \\
&= (\cancel{a_2 a_1} + \cancel{a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1))} - a_2 \kappa_2 a_2) - \cancel{a_2(\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1))} - \cancel{a_1 a_2} \\
&= -a_2 \kappa_2 a_2 = -\kappa_2 a_2^2.
\end{aligned} \tag{2.52}
$$

Finally, combining (2.51) and (2.15), formula for $\mathbf{U}_1$ is obtained as follows:

$$
\frac{\mathbf{U}_1}{m}(\cos\phi \cos\theta) - g = a_1 + (\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - \kappa_2 a_2 \tag{2.53}
$$

$$\Rightarrow \quad \frac{\mathbf{U}_1}{m}(\cos\phi\cos\theta) = a_1 + g + (\ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1)) - \kappa_2 a_2 \tag{2.54}$$

$$\Rightarrow \quad \mathbf{U}_1 = \frac{m}{\cos\phi\cos\theta}\left(a_1 + g + \ddot{z}_d - \kappa_1(a_2 + \kappa_1 a_1) - \kappa_2 a_2\right). \tag{2.55}$$

Setting $\ddot{z}_d = 0$ [61], the altitude control input that guarantees the asymptotic stability in the sense of Lyapunov is yielded, as follows

$$\boxed{\mathbf{U}_1 = \frac{m}{\cos\phi\cos\theta}\left(a_1 + g - \kappa_1(a_2 + \kappa_1 a_1) - \kappa_2 a_2\right).} \tag{2.56}$$

Design of Roll Control Input $\mathbf{U}_2$

The the control algorithm for roll input $\mathbf{U}_2$ can be derived using the same line of reasoning as the control input $\mathbf{U}_1$ in the previous section. We first define the roll tracking error a_3, as follows:

$$a_3 := \phi_d - \phi, \tag{2.57}$$

where ϕ_d and ϕ are the desired and actual roll angles, respectively. Next, define the Lyapunov function candidate $V(a_3)$ of the roll tracking error to have the following form:

$$V(a_3) = \frac{1}{2}a_3^2. \tag{2.58}$$

The time derivative of Lyapunov function (2.58) is

$$\dot{V}(a_3) = a_3(\dot{\phi}_d - \dot{\phi}). \tag{2.59}$$

One would like to make the expression (2.59) bounded from above by a quadratic negative definite function of a_3:

$$\dot{V}(a_3) \leq -\kappa_3 a_3^2, \tag{2.60}$$

where $\kappa_3 > 0$ is a constant. By substituting (2.59) into (2.60), the following inequality is yielded,

$$a_3(\dot{\phi}_d - \dot{\phi}) \leq -\kappa_3 a_3^2. \tag{2.61}$$

The inequality (2.61) is guaranteed if the state variable $\dot{\phi}$ satisfies the following relation

$$\dot{\phi} = \dot{\phi}_d + \kappa_3 a_3. \tag{2.62}$$

At this stage of the backstepping procedure, the virtual control for the state variable $\dot{\phi}$ is defined as follows,

$$\phi_{vir} := \dot{\phi}_d + \kappa_3 a_3. \tag{2.63}$$

This virtual variable ϕ_{vir} determines the desired value of the state variable $\dot{\phi}$. The next step of the backstepping procedure is to define a new variable a_4 that represents the difference between the state variable $\dot{\phi}$ and the virtual control ϕ_{vir}, hence,

$$a_4 := \dot{\phi} - \phi_{vir}. \tag{2.64}$$

Using the above equation, the time derivative of Lyapunov function $\dot{V}(a_3)$ (2.59) is calculated, as follows,

$$\begin{aligned}
\dot{V}(a_3) &= a_3(\dot{\phi}_d - \dot{\phi}) = a_3(\dot{\phi}_d - (a_4 + \phi_{vir})) \\
&= a_3(\dot{\phi}_d - (a_4 + \dot{\phi}_d + \kappa_3 a_3)) = \cancel{a_3\dot{\phi}_d} - a_3 a_4 - \cancel{a_3\dot{\phi}_d} - \kappa_3 a_3^2 \\
&= -a_3 a_4 - \kappa_3 a_3^2.
\end{aligned} \tag{2.65}$$

After augmenting the Lyapunov function (2.58) with a new term $V(a_4) := \frac{1}{2}a_4^2$, the total Lyapunov function $V(a_3, a_4)$ becomes

$$\begin{aligned}
V(a_3, a_4) &= V(a_3) + V(a_4) \\
&= \frac{1}{2}a_3^2 + \frac{1}{2}a_4^2
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}(\phi_d - \phi)^2 + \frac{1}{2}(\dot{\phi} - \phi_{vir})^2 \\
&= \frac{1}{2}(\phi_d^2 - 2\phi\phi_d + \phi^2) + \frac{1}{2}(\dot{\phi}^2 - 2\dot{\phi}\phi_{vir} + \phi_{vir}^2) \\
&= \frac{1}{2}(\phi_d^2 - 2\phi\phi_d + \phi^2) + \frac{1}{2}\dot{\phi}^2 - \dot{\phi}(\dot{\phi}_d + \kappa_3\phi_d - \kappa_3\phi) \\
&\quad + \frac{1}{2}(\dot{\phi}_d^2 + 2\kappa_3\dot{\phi}_d\phi_d - 2\kappa_3\dot{\phi} + \kappa_3^2\phi_d^2 - 2\kappa_3^2\phi_d\phi + \kappa_3^2\phi^2)
\end{aligned} \tag{2.66}
$$

The time derivative of the total Lyapunov function $V(a_3, a_4)$ is

$$
\begin{aligned}
\dot{V}(a_3, a_4) = \quad & (\phi_d - \phi - \kappa_3\dot{\phi} + \kappa_3\dot{\phi}_d + \kappa_3^2\phi_d - \kappa_3^2\phi)\dot{\phi}_d \\
& + (-\phi_d + \phi + \kappa_3\dot{\phi} - \kappa_3\dot{\phi}_d - \kappa_3^2\phi_d + \kappa_3^2\phi)\dot{\phi} \\
& + (-\dot{\phi} + \dot{\phi}_d + \kappa_3\phi_d - \kappa_3\phi)\ddot{\phi}_d \\
& + (\dot{\phi} - \dot{\phi}_d - \kappa_3 z_d + \kappa_3\phi)\ddot{\phi}.
\end{aligned} \tag{2.67}
$$

By rearranging (2.67), we get

$$
\dot{V}(a_3, a_4) = a_4\ddot{\phi} - a_4(\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3)) - a_3 a_4 - \kappa_3 a_3^2. \tag{2.68}
$$

The exponential convergence $a_3(t), a_4(t) \to 0$ will be guaranteed if the time derivative of the total Lyapunov function $V(a_3, a_4)$ satisfies the inequality

$$
\dot{V}(a_3, a_4) \leq -\kappa_3 a_3^2 - \kappa_4 a_4^2 \tag{2.69}
$$

where $\kappa_3, \kappa_4 > 0$. Taking into account (2.68), inequality (2.69) becomes

$$
a_4\ddot{\phi} - a_4(\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3)) - a_3 a_4 \cancel{- \kappa_3 a_3^2} \leq \cancel{-\kappa_3 a_3^2} - \kappa_4 a_4^2
$$

or

$$
a_4\ddot{\phi} - a_4(\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3)) - a_3 a_4 \leq -\kappa_4 a_4^2. \tag{2.70}
$$

The last inequality (2.70) is satisfied if the following relation holds

$$
\ddot{\phi} = a_3 + [\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3)] - \kappa_4 a_4 \tag{2.71}
$$

Substituting (2.16) into (2.71), the following formula is yielded,

$$\left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} + \frac{l}{J_x}\mathbf{U}_2 = a_3 + (\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3)) - \kappa_4 a_4 \tag{2.72}$$

$$\Rightarrow \quad \frac{l}{J_x}\mathbf{U}_2 = a_3 - \left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} + (\ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_4)) - \kappa_4 a_4 \tag{2.73}$$

$$\Rightarrow \quad \mathbf{U}_2 = \frac{J_x}{l}\left(a_3 - \left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} + \ddot{\phi}_d - \kappa_3(a_4 + \kappa_3 a_3) - \kappa_4 a_4\right) \tag{2.74}$$

Setting $\ddot{\phi}_d = 0$ [61], the roll angle control input $\mathbf{U}_2$ is defined according to the following formula,

$$\boxed{\mathbf{U}_2 = \frac{J_x}{l}\left(a_3 - \left(\frac{J_y - J_z}{J_x}\right)\dot{\theta}\dot{\psi} - \kappa_3(a_4 + \kappa_3 a_3) - \kappa_4 a_4\right).} \tag{2.75}$$

Design of Pitch Control Input $\mathbf{U}_3$

The pitch control input $\mathbf{U}_3$ is derived in the same way as $\mathbf{U}_1$ and $\mathbf{U}_2$ in the previous sections. The first step of the backstepping procedure for the pitch control input is to define the pitch tracking error a_5, hence

$$a_5 := \theta_d - \theta, \tag{2.76}$$

where θ_d and θ are the desired and actual pitch angles, respectively. The Lyapunov function candidate of the pitch tracking error is defined according to the following formula,

$$V(a_5) = \frac{1}{2}a_5^2. \tag{2.77}$$

The time derivative of (2.77) is

$$\dot{V}(a_5) = a_5(\dot{\theta}_d - \dot{\theta}). \tag{2.78}$$

Our goal is to guarantee that $\dot{V}(a_5)$ satisfies the following inequality,

$$\dot{V}(a_5) \leq -\kappa_5 a_5^2, \tag{2.79}$$

where $\kappa_3 > 0$ is a constant. Substituting (2.78) into (2.79), one obtains the following inequality

$$a_5(\dot{\theta}_d - \dot{\theta}) \leq -\kappa_5 a_5^2. \tag{2.80}$$

The inequality (2.80) holds if the state variable $\dot{\theta}$ satisfies the following relation

$$\dot{\theta} = \dot{\theta}_d + \kappa_5 a_5. \tag{2.81}$$

The virtual control θ_{vir} which represents the desired value of the state variable $\dot{\theta}$ is, therefore, defined as follows

$$\theta_{vir} := \dot{\theta}_d + \kappa_5 a_5. \tag{2.82}$$

The next step is to define a new variable a_6 which represents the difference between the state variable $\dot{\theta}$ and the virtual control θ_{vir}; hence,

$$a_6 := \dot{\theta} - \theta_{vir}. \tag{2.83}$$

Combining (2.82) and (2.83), one gets

$$\dot{V}(a_5) = -a_5 a_6 - \kappa_5 a_5^2. \tag{2.84}$$

The total Lyapunov function $V(a_5, a_6)$ is formed by augmenting (2.77) with an additional term $V(a_6) := \frac{1}{2}a_4^2$, which gives

$$\begin{aligned} V(a_5, a_6) &= V(a_5) + V(a_6) \\ &= \frac{1}{2}a_5^2 + \frac{1}{2}a_6^2 \\ &= \frac{1}{2}(\theta_d - \theta)^2 + \frac{1}{2}(\dot{\theta} - \theta_{\boldsymbol{vir}})^2 \\ &= \frac{1}{2}(\theta_d^2 - 2\theta\theta_d + z^2) + \frac{1}{2}(\dot{\theta}^2 - 2\dot{\theta}\theta_{vir} + \theta_{vir}^2) \\ &= \frac{1}{2}(\theta_d^2 - 2\theta\theta_d + \theta^2) + \frac{1}{2}\dot{\theta}^2 - \dot{\theta}(\dot{\theta}_d + \kappa_5\theta_d - \kappa_5\theta) \\ &\quad + \frac{1}{2}(\dot{\theta}_d^2 + 2\kappa_5\dot{\theta}_d\theta_d - 2\kappa_5\dot{\theta} + \kappa_5^2\theta_d^2 - 2\kappa_5^2\theta_d\theta + \kappa_5^2\theta^2). \end{aligned} \tag{2.85}$$

The time derivative of the total Lyapunov function $V(a_5, a_6)$ is

$$\begin{aligned}\dot{V}(a_5, a_6) = &(\theta_d - \theta - \kappa_5\dot{\theta} + \kappa_5\dot{\theta}_d + \kappa_5^2\theta_d - \kappa_5^2\theta)\dot{\theta}_d \\ &+ (-\theta_d + \theta + \kappa_5\dot{\theta} - \kappa_5\dot{\theta}_d - \kappa_5^2\theta_d + \kappa_5^2\theta)\dot{\theta} \\ &+ (-\dot{\theta} + \dot{\theta}_d + \kappa_5\theta_d - \kappa_5\theta)\ddot{\theta}_d \\ &+ (\dot{\theta} - \dot{\theta}_d - \kappa_5\theta_d + \kappa_5\theta)\ddot{\theta}.\end{aligned} \tag{2.86}$$

By rearranging (2.86), we get

$$\dot{V}(a_5, a_6) = a_6\ddot{\theta} - a_6(\ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_5)) - a_5 a_6 - \kappa_5 a_5^2. \tag{2.87}$$

The exponential convergence $a_5(t), a_6(t) \to 0$ will be guaranteed if the time derivative of the total Lyapunov function $V(a_5, a_6)$ satisfies the inequality

$$\dot{V}(a_5, a_6) \leq -\kappa_5 a_5^2 - \kappa_6 a_6^2, \tag{2.88}$$

where $\kappa_5, \kappa_6 > 0$. From (2.87) and (2.88), we get

$$a_6\ddot{\theta} - a_6(\ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_5)) - a_5 a_6 \leq -\kappa_6 a_6^2. \tag{2.89}$$

The last inequality holds if $\ddot{\theta}$ satisfies the following relation:

$$\ddot{\theta} = a_5 + [\ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_5)] - \kappa_6 a_6. \tag{2.90}$$

Combining (2.90) and (2.17), the formula for $\mathbf{U}_3$ is obtained as follows:

$$\left(\frac{J_z - J_x}{J_y}\right)\dot{\theta}\dot{\psi} + \frac{l}{J_y}\mathbf{U}_3 = a_5 + (\ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_5)) - \kappa_6 a_6 \tag{2.91}$$

$$\Rightarrow \quad \frac{l}{J_y}\mathbf{U}_3 = a_5 - \left(\frac{J_z - J_x}{J_y}\right)\dot{\theta}\dot{\psi} + (\ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_6)) - \kappa_6 a_6 \tag{2.92}$$

$$\Rightarrow \quad \mathbf{U}_3 = \frac{J_y}{l}\left(a_5 - \left(\frac{J_z - J_x}{J_y}\right)\dot{\theta}\dot{\psi} + \ddot{\theta}_d - \kappa_5(a_6 + \kappa_5 a_5) - \kappa_6 a_6\right) \tag{2.93}$$

Finally, by setting $\ddot{\theta}_d = 0$ [61], the pitch angle control input is obtained:

$$\boxed{\mathbf{U}_3 = \frac{J_y}{l}\left(a_5 - \left(\frac{J_z - J_x}{J_y}\right)\dot{\theta}\dot{\psi} - \kappa_5(a_6 + \kappa_5 a_5) - \kappa_6 a_6\right).} \tag{2.94}$$

Design of Yaw Control Input $\mathbf{U}_4$

In this section, the yaw control input $\mathbf{U}_4$ is designed. The derivation follows the same line of reasoning as those for $\mathbf{U}_1, \mathbf{U}_2$, and $\mathbf{U}_3$ in the previous sections. Let us carry out the first step of the backstepping procedure by defining the yaw tracking error as follows,

$$a_7 := \psi_d - \psi, \tag{2.95}$$

where ψ_d and ψ are the desired and actual yaw angles, respectively. Define a Lyapunov function candidate $V(a_7)$ as follows

$$V(a_7) = \frac{1}{2}a_7^2. \tag{2.96}$$

We seek to determine the time derivative of Lyapunov function (2.96), hence,

$$\dot{V}(a_7) = a_7(\dot{\psi}_d - \dot{\psi}). \tag{2.97}$$

Again, we aim to guarantee that $\dot{V}(a_7)$ is bounded from above by a negative definite function of a_7, such as

$$\dot{V}(a_7) \leq -\kappa_7 a_7^2, \tag{2.98}$$

where $\kappa_7 > 0$ is a constant. Combining (2.97) and (2.98), the following inequality is obtained

$$a_7(\dot{\psi}_d - \dot{\psi}) \leq -\kappa_7 a_7^2. \tag{2.99}$$

Inequality (2.99) is guaranteed if the state variable $\dot{\psi}$ satisfies the following relation,

$$\dot{\psi} = \dot{\psi}_d + \kappa_7 a_7. \tag{2.100}$$

The virtual control ψ_{vir} that corresponds to the state variable $\dot{\psi}$ is therefore defined as follows:

$$\psi_{vir} := \dot{\psi}_d + \kappa_7 a_7. \tag{2.101}$$

The second step of the backstepping procedure is to introduce a new variable a_8 which represents the difference between the state variable $\dot{\psi}$ and the virtual control ψ_{vir},

$$a_8 := \dot{\psi} - \psi_{vir}. \tag{2.102}$$

Combining (2.102) and (2.97), we get the following expression

$$\dot{V}(a_7) = -a_7 a_8 - \kappa_7 a_7^2. \tag{2.103}$$

At this stage, we form a total Lyapunov function $V(a_7, a_8)$ by adding a term $V(a_8) := \frac{1}{2}a_8^2$ to (2.96) to get

$$\begin{aligned} V(a_7, a_8) &= V(a_7) + V(a_8) \\ &= \frac{1}{2}a_7^2 + \frac{1}{2}a_8^2 \\ &= \frac{1}{2}(\psi_d^2 - 2\psi\psi_d + \psi^2) + \frac{1}{2}\dot{\psi}^2 - \dot{\psi}(\dot{\psi}_d + \kappa_7\psi_d - \kappa_7\psi) \\ &\quad + \frac{1}{2}(\dot{\psi}_d^2 + 2\kappa_7\dot{\psi}_d\psi_d - 2\kappa_7\dot{\psi} + \kappa_7^2\psi_d^2 - 2\kappa_7^2\psi_d\psi + \kappa_7^2\psi^2). \end{aligned} \tag{2.104}$$

The time derivative of the total Lyapunov function $V(a_7, a_8)$ is

$$\begin{aligned} \dot{V}(a_7, a_8) = \quad & (\psi_d - \psi - \kappa_7\dot{\psi} + \kappa_7\dot{\psi}_d + \kappa_7^2\psi_d - \kappa_7^2\psi)\dot{\psi}_d \\ & + (-\psi_d + \psi + \kappa_7\dot{\psi} - \kappa_7\dot{\psi}_d - \kappa_7^2\psi_d + \kappa_7^2\psi)\dot{\psi} \\ & + (-\dot{\psi} + \dot{\psi}_d + \kappa_7\psi_d - \kappa_7\psi)\ddot{\psi}_d \\ & + (\dot{\psi} - \dot{\psi}_d - \kappa_7\psi_d + \kappa_7\psi)\ddot{\psi}, \end{aligned} \tag{2.105}$$

or

$$\dot{V}(a_7, a_8) = a_8\ddot{\psi} - a_8(\ddot{\psi}_d - \kappa_7(a_8 + \kappa_7 a_7)) - a_7 a_8 - \kappa_7 a_7^2 \tag{2.106}$$

Again, our goal is to guarantee that

$$\dot{V}(a_7, a_8) \leq -\kappa_7 a_7^2 - \kappa_8 a_8^2, \tag{2.107}$$

where $\kappa_7, \kappa_8 > 0$, which implies the exponential convergence $a_7(t), a_8(t) \to 0$. From (2.106) and (2.107), we get

$$a_8\ddot{\psi} - a_8(\ddot{\psi}_d - \kappa_7(a_8 + \kappa_7 a_7)) - a_7 a_8 \leq -\kappa_8 a_8^2 \tag{2.108}$$

The above inequality (2.108) holds if the state variable $\ddot{\psi}$ satisfies the following relation

$$\ddot{\psi} = a_7 + (\ddot{\psi}_d - \kappa_7[a_8 + \kappa_7 a_7)] - \kappa_8 a_8. \tag{2.109}$$

From (2.109) and (2.18), the following expression is yielded,

$$\begin{aligned}
\left(\frac{J_x - J_y}{J_z}\right)\dot{\phi}\dot{\theta} + \frac{1}{J_z}\mathbf{U}_4 &= a_7 + (\ddot{\psi}_d - \kappa_7(a_8 + \kappa_7 a_7)) - \kappa_8 a_8 \\
\Rightarrow \quad \frac{1}{J_z}\mathbf{U}_4 &= a_7 - \left(\frac{J_x - J_y}{J_z}\right)\dot{\phi}\dot{\theta} + (\ddot{\psi}_d - \kappa_7(a_8 + \kappa_7 a_8)) - \kappa_8 a_8 \\
\Rightarrow \quad \mathbf{U}_4 &= \frac{J_y}{l}\left(a_5 - \left(\frac{J_x - J_y}{J_z}\right)\dot{\phi}\dot{\theta} + \ddot{\psi}_d - \kappa_7(a_8 + \kappa_7 a_7) - \kappa_8 a_8\right)
\end{aligned} \tag{2.110}$$

Setting $\ddot{\psi}_d = 0$ [61], we obtain the yaw angle control input $\mathbf{U}_4$ that guarantees the asymptotic stability as follows,

$$\boxed{\mathbf{U}_4 = \frac{J_y}{l}\left(a_5 - \left(\frac{J_x - J_y}{J_z}\right)\dot{\phi}\dot{\theta} - \kappa_7(a_8 + \kappa_7 a_7) - \kappa_8 a_8\right).} \tag{2.111}$$

Simulation Results for Attitude Backstepping Control

Using odeint C++ library, simulating the system has been carried out to illustrate the performance of the backstepping controller. Table 2.4 shows the numerical values of the parameters for the experiment and Fig. 2.20, 2.21, and 2.22 show the response of the quadrotor.

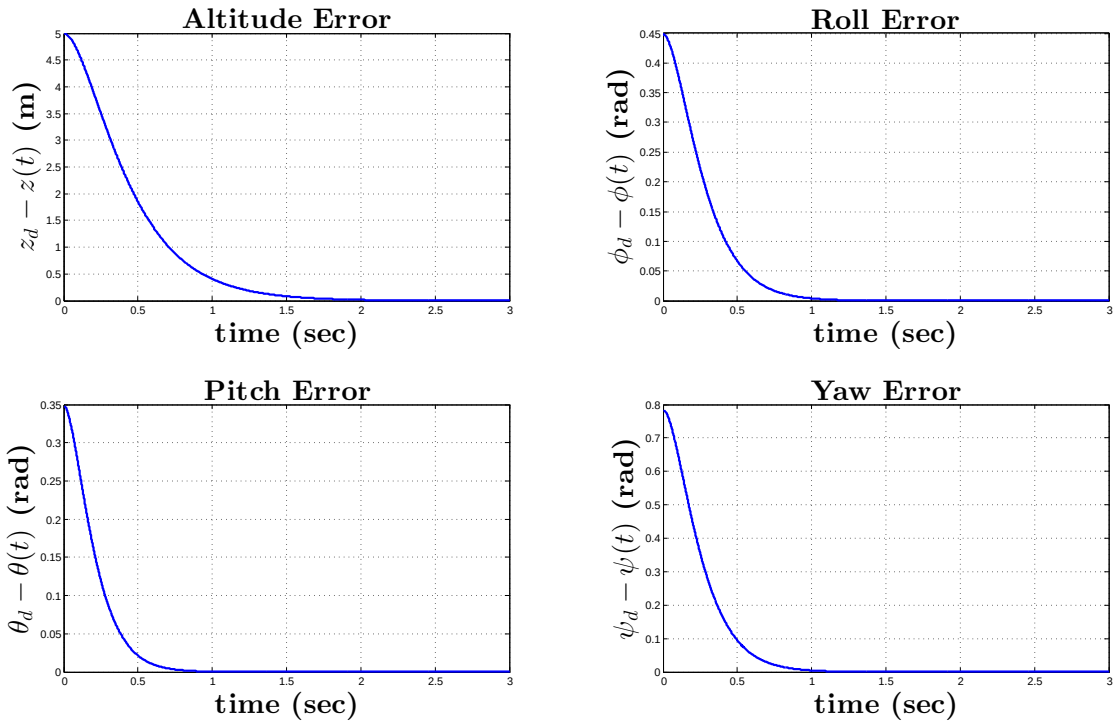


Figure 2.20: Attitude regulation problem: altitude and attitude errors for backstepping controller.

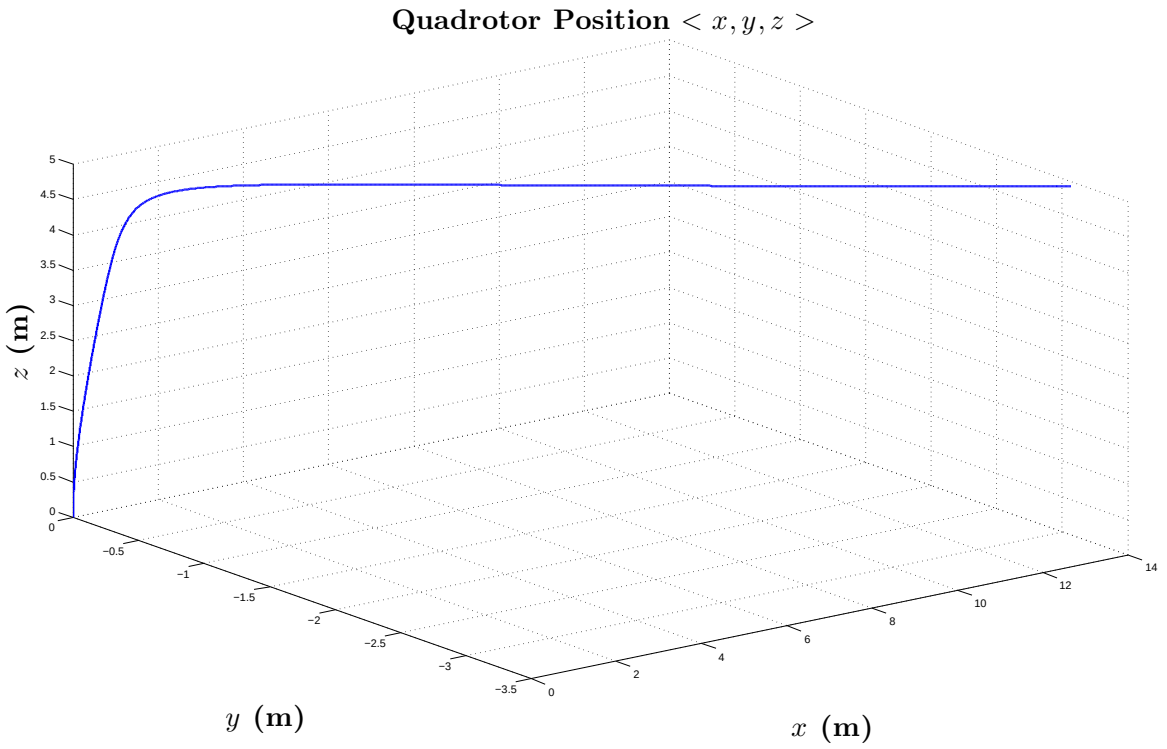


Figure 2.21: Attitude regulation problem: the position (x, y, z) of the center of the quadrotor's mass using backstepping controller.

	initial values	Gains	Desired Trajectory	Unit
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$\kappa_1 = 6.35$ $\kappa_2 = 2.91$	$z_d = 5$	(m)
Roll	$\phi(0) = 0$ $\dot{\phi}(0) = 0$	$\kappa_3 = 7.32$ $\kappa_4 = 5.99$	$\phi_d = \frac{\pi}{7}$	(rad)
Pitch	$\theta(0) = 0$ $\dot{\theta}(0) = 0$	$\kappa_5 = 9.6184$ $\kappa_6 = 8.11$	$\theta_d = \frac{\pi}{9}$	(rad)
Yaw	$\psi(0) = 0$ $\dot{\psi}(0) = 0$	$\kappa_7 = 8.412$ $\kappa_8 = 6.21$	$\psi_d = \frac{\pi}{4}$	(rad)

Table 2.4: Backstepping controller parameters for attitude control (regulation problem)

2.4.5 Position Control for Nonlinear Model

From (2.13) and (2.14), we can see that $\ddot{x}$ and $\ddot{y}$ can not be controlled directly due to the fact that the system is underactuated. However, one can notice that the vertical force $\mathbf{U}_1$ enters the aforementioned equations. The control input $\mathbf{U}_1$ in (2.13) is multiplied by the term $(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)$, while in (2.14) it is multiplied by the term $(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)$. One can exploit this fact to control the roll and pitch angles that form the new force vectors in $x-y$ plane. Consequently, let $\mathbf{U}_x$ and $\mathbf{U}_y$ be virtual control inputs for x and y linear positions, respectively, and ψ_T be the yaw angle. We need to extract ϕ_d and θ_d from (2.13) and (2.14), therefore, the relationship between $\mathbf{U}_x$, $\mathbf{U}_y$, ϕ_d and θ_d is given by the following expressions,

$$\mathbf{U}_x = (\cos\phi_d \sin\theta_d \cos\psi_T + \sin\phi_d \sin\psi_T)\,, \tag{2.112}$$

$$\mathbf{U}_y = (\cos\phi_d \sin\theta_d \sin\psi_T - \sin\phi_d \cos\psi_T)\,. \tag{2.113}$$

Multiply both sides of (2.112) by $(\sin\psi_T)$, and those of (2.113) by $(\cos\psi_T)$, we obtain

$$\mathbf{U}_x \sin\psi_T = \left(\cos\phi_d \sin\theta_d \sin\psi_T \cos\psi_T + \sin\phi_d \sin^2\psi_T\right), \tag{2.114}$$

$$\mathbf{U}_y \cos\psi_T = \left(\cos\phi_d \sin\theta_d \sin\psi_T \cos\psi_T - \sin\phi_d \cos^2\psi_T\right). \tag{2.115}$$

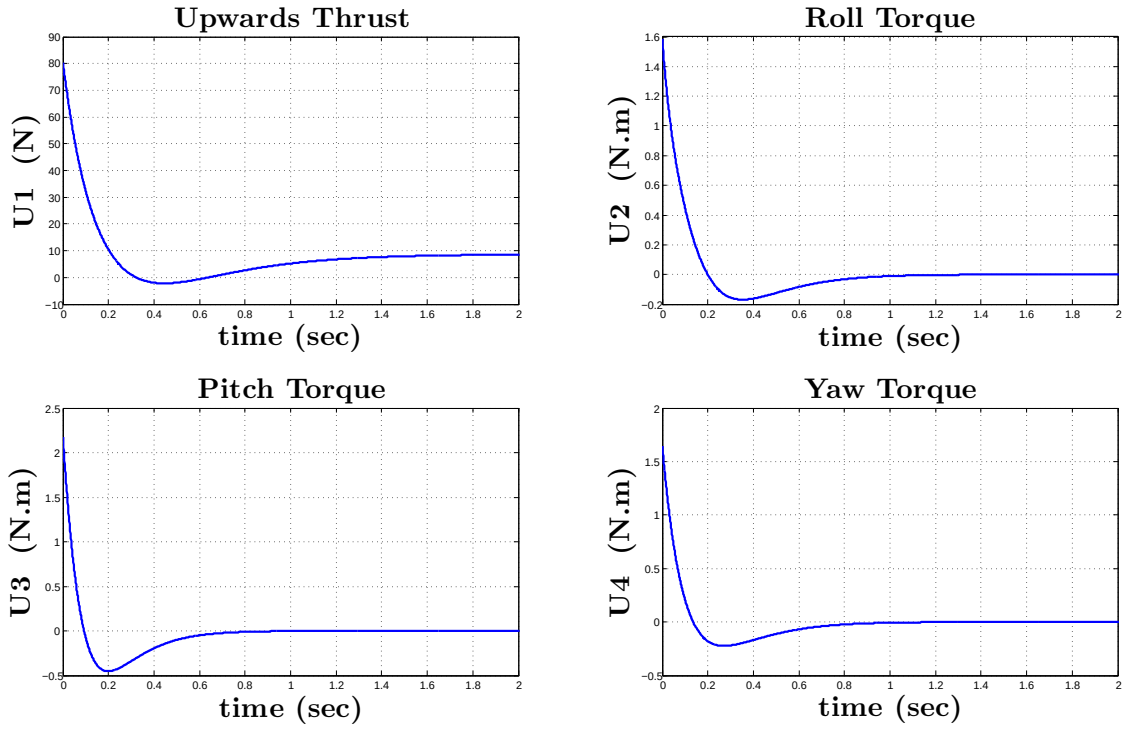


Figure 2.22: Attitude regulation problem: magnitude of the control inputs for backstepping controller .

By subtracting (2.115) from (2.114), we obtain

$$(\mathbf{U}_x \sin\psi_T) - (\mathbf{U}_y \cos\psi_T) = \sin\phi_d \left(\sin^2\psi_T + \cos^2\psi_T\right) \quad (2.116)$$

By rearranging (2.116), we obtain a closed form for ϕ_d, as follows

$$\boxed{\phi_d = \sin^{-1}(\mathbf{U}_x \sin\psi_T - \mathbf{U}_y \cos\psi_T)} \quad (2.117)$$

Again, multiply both sides of (2.112) by $(\cos\psi_T)$ and (2.113) by $(\sin\psi_T)$, we obtain,

$$\mathbf{U}_x \cos\psi_T = \left(\cos\phi_d \sin\theta_d \cos^2\psi_T + \sin\phi_d \sin\psi_T \cos\psi_T\right) \quad (2.118)$$

$$\mathbf{U}_y \sin\psi_T = \left(\cos\phi_d \sin\theta_d \sin^2\psi_T - \sin\phi_d \sin\psi_T \cos\psi_T\right) \quad (2.119)$$

By adding (2.119)) to (2.118), we obtain,

$$\mathbf{U}_x \cos\psi_T + \mathbf{U}_y \sin\psi_T = \cos\phi_d \sin\theta_d \underbrace{\left(\cos^2\psi_T + \sin^2\psi_T\right)}_{1} \tag{2.120}$$

By rearranging (2.120), a closed form for θ_d is yielded, hence

$$\boxed{\theta_d = \sin^{-1}\left(\frac{\mathbf{U}_x \cos\psi_T + \mathbf{U}_y \sin\psi_T}{\cos\phi_d}\right)} \tag{2.121}$$

where ϕ_d is provided by (2.117).

Design of Virtual Control Inputs $\mathbf{U}_x$, $\mathbf{U}_y$

As we have seen from the previous section, the vertical force can be used to yield new force vectors in $x-y$ plane; specifically, $\ddot{x}$ and $\ddot{y}$ can be rewritten as follows,

$$\ddot{x} = \frac{\mathbf{U}_1}{m}\mathbf{U}_x, \tag{2.122}$$

$$\ddot{y} = \frac{\mathbf{U}_1}{m}\mathbf{U}_y, \tag{2.123}$$

where $\mathbf{U}_x$ and $\mathbf{U}_y$ are

$$\mathbf{U}_x = \left(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi\right), \tag{2.124}$$

$$\mathbf{U}_y = \left(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi\right). \tag{2.125}$$

The control input $\mathbf{U}_x$ is derived in the same way as were the previous control inputs for attitude control. The first step is to define a new variable a_9 that represents the x-component of the position tracking error, as follows,

$$a_9 := x_d - x, \tag{2.126}$$

where x_d and x are the desired and actual linear position of the center of the quadrotor's mass along x-axis. Let a Lyapunov function candidate $V(a_9)$ be defined as follows,

$$V(a_9) = \frac{1}{2}a_9^2. \tag{2.127}$$

The time derivative of Lyapunov function (2.127) is

$$\dot{V}(a_9) = a_9(\dot{x}_d - \dot{x}). \tag{2.128}$$

Now we impose an upper bound on the expression (2.128) in the form of a quadratic negative definite function of a_9, as follows,

$$\dot{V}(a_9) \leq -\kappa_9 a_9^2, \tag{2.129}$$

where $\kappa_9 > 0$ is a constant. From (2.128) and (2.129), we acquire the following expression,

$$a_9(\dot{x}_d - \dot{x}) \leq -\kappa_9 a_9^2. \tag{2.130}$$

Inequality (2.130) is guaranteed if the state variable $\dot{x}$ satisfies

$$\dot{x} = \dot{x}_d + \kappa_9 a_9 \tag{2.131}$$

Let a new variable x_{vir} be defined to represent the desired value of the state variable $\dot{x}$, as follows,

$$x_{vir} := \dot{x}_d + \kappa_9 a_9 \tag{2.132}$$

The above defined variable x_{vir} represents the virtual control for the linear position of the center of the quadrotor's mass in x-axis. The second stage of the backstepping approach is to define a new variable a_{10} that represents the difference between the state variable $\dot{x}$ and the virtual control x_{vir} as follows,

$$a_{10} := \dot{x} - x_{vir}. \tag{2.133}$$

From (2.133) and (2.128), we get another expression for $\dot{V}(a_9)$, which is

$$\dot{V}(a_9) = -a_9 a_{10} - \kappa_9 a_9^2. \tag{2.134}$$

By adding a new term $V(a_{10}) := \frac{1}{2}a_{10}^2$ to the Lyapunov function (2.127), the total Lyapunov function $V(a_9, a_{10})$ is yielded as follows,

$$\begin{aligned} V(a_9, a_{10}) &= V(a_9) + V(a_{10}) \\ &= \frac{1}{2}a_9^2 + \frac{1}{2}a_{10}^2 \\ &= \frac{1}{2}(x_d^2 - 2xx_d + x^2) + \frac{1}{2}\dot{x}^2 - \dot{x}(\dot{x}_d + \kappa_9 x_d - \kappa_9 x) \\ &\quad + \frac{1}{2}(\dot{x}_d^2 + 2\kappa_9\dot{x}_d x_d - 2\kappa_9\dot{x} + \kappa_9^2 x_d^2 - 2\kappa_9^2 x_d x + \kappa_9^2 x^2). \end{aligned} \tag{2.135}$$

The time derivative of the total Lyapunov function $V(a_9, a_{10})$ is

$$\begin{aligned} \dot{V}(a_9, a_{10}) = \quad & (x_d - x - \kappa_9\dot{x} + \kappa_9\dot{x}_d + \kappa_9^2 x_d - \kappa_9^2 x)\dot{x}_d \\ & + (-x_d + x + \kappa_9\dot{x} - \kappa_9\dot{x}_d - \kappa_9^2 x_d + \kappa_9^2 x)\dot{x} \\ & + (-\dot{x} + \dot{x}_d + \kappa_9 x_d - \kappa_9 x)\ddot{x}_d \\ & + (\dot{x} - \dot{x}_d - \kappa_9 x_d + \kappa_9 x)\ddot{x}. \end{aligned} \tag{2.136}$$

By rearranging (2.136), one obtains

$$\dot{V}(a_9, a_{10}) = a_{10}\ddot{x} - a_{10}(\ddot{x}_d - \kappa_9(a_{10} + \kappa_9 a_9)) - a_9 a_{10} - \kappa_9 a_9^2 \tag{2.137}$$

The following inequality impose a bound on the derivative of the total Lyapunov function $V(a_9, a_{10})$ so that the exponential convergence $a_9(t), a_{10}(t) \to 0$ will be guaranteed:

$$\dot{V}(a_9, a_{10}) \leq -\kappa_9 a_9^2 - \kappa_{10} a_{10}^2, \tag{2.138}$$

where $\kappa_9, \kappa_{10} > 0$. From (2.137) and inequality (2.138), one gets

$$a_{10}\ddot{x} - a_{10}(\ddot{x}_d - \kappa_9(a_{10} + \kappa_9 a_9)) - a_9 a_{10} \leq -\kappa_{10} a_{10}^2. \tag{2.139}$$

The last inequality (2.139) is satisfied if the following relation holds

$$\ddot{x} = a_9 + (\ddot{x}_d - \kappa_9[a_{10} + \kappa_9 a_9)] - \kappa_{10} a_{10}. \tag{2.140}$$

Finally, combining (2.140) and (2.122), formula for $\mathbf{U}_x$ is obtained as follows,

$$\frac{\mathbf{U}_1}{m}\mathbf{U}_x = a_9 + (\ddot{x}_d - \kappa_9(a_{10} + \kappa_9 a_9)) - \kappa_{10} a_{10} \tag{2.141}$$

$$\Rightarrow \quad \mathbf{U}_x = \frac{m}{\mathbf{U}_1}\left(a_9 + (\ddot{x}_d - \kappa_9(a_{10} + \kappa_9 a_9)) - \kappa_{10} a_{10}\right). \tag{2.142}$$

Setting $\ddot{x}_d = 0$ [61], the control input $\mathbf{U}_x$ is obtained as follows,

$$\boxed{\mathbf{U}_x = \frac{m}{\mathbf{U}_1}\left(a_9 - \kappa_9(a_{10} + \kappa_9 a_9) - \kappa_{10} a_{10}\right).} \tag{2.143}$$

Using the exact same procedure, $\mathbf{U}_y$ is derived as follows,

$$\boxed{\mathbf{U}_y = \frac{m}{\mathbf{U}_1}\left(a_{11} - \kappa_{11}(a_{12} + \kappa_{11} a_{11}) - \kappa_{12} a_{12}\right)} \tag{2.144}$$

where $\kappa_{11}, \kappa_{12} > 0$ are constants, and

$$a_{11} := y_d - y, \tag{2.145}$$

$$a_{12} := \dot{y} - y_{vir}, \tag{2.146}$$

$$y_{vir} := \dot{y} + \kappa_{11} a_{11}, \tag{2.147}$$

where y_d and y are the desired and the actual linear position of the center of the quadrotor's mass along y-axis.

Simulation Results for Backstepping Position Control

Using odeint C++ library, simulations have been carried out to illustrate the performance of the backstepping controller. Table 2.5 shows the numerical values of the parameters for the simulations, and Figures 2.23, 2.24 and 2.25 for show the response of the quadrotor.

	initial values	Gains	Desired Trajectory
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$\kappa_1 = 3.6$ $\kappa_2 = 3.3$	$z_d = 2.5$ (m)
x	$x(0) = 0$ $\dot{x}(0) = 0$	$\kappa_9 = 3.4$ $\kappa_{10} = 2.0$	$x_d = 2.5$ (m)
y	$y(0) = 0$ $\dot{y}(0) = 0$	$\kappa_{11} = 3.4$ $\kappa_{12} = 2.0$	$y_d = 2.5$ (m)
Roll	$\phi(0) = 0$ $\dot{\phi}(0) = 0$	$\kappa_3 = 25.5$ $\kappa_4 = 20.0$	
Pitch	$\theta(0) = 0$ $\dot{\theta}(0) = 0$	$\kappa_5 = 25.5$ $\kappa_6 = 20.0$	
Yaw	$\psi(0) = 0$ $\dot{\psi}(0) = 0$	$\kappa_7 = 25.5$ $\kappa_8 = 20.0$	$\psi_d = 0$ (rad)

Table 2.5: Backstepping controller parameters for position control (regulation problem)

2.4.6 Spatial Velocity Control for UAV

In previous sections, backstepping controller has been utilized for attitude and position controls of nonlinear model of UAV. For position control, the objective is to force actual position variables (i.e. x, y, and z) to follow their desired trajectories (i.e. x_d, y_d and z_d). In some applications, it is preferable and convenient to control the spatial velocity of an UAV rather than its position. This is due to the limited workspace of the haptic device, which doesn't allow for position-position control when the UAV is to cover large distances. The aim of this section is to show how the backstepping controller can be modified to achieve linear velocity control. Let a_{13} define the linear velocity tracking error in z-axis, as follows

$$a_{13} := \dot{z}_d - \dot{z}, \tag{2.148}$$

where $\dot{z}_d$ and $\dot{z}$ are the desired and actual linear velocities of the center of the quadrotor's mass in z-axis, respectively. Let a Lyapunov function $V(a_{13})$ be defined to be a positive definite

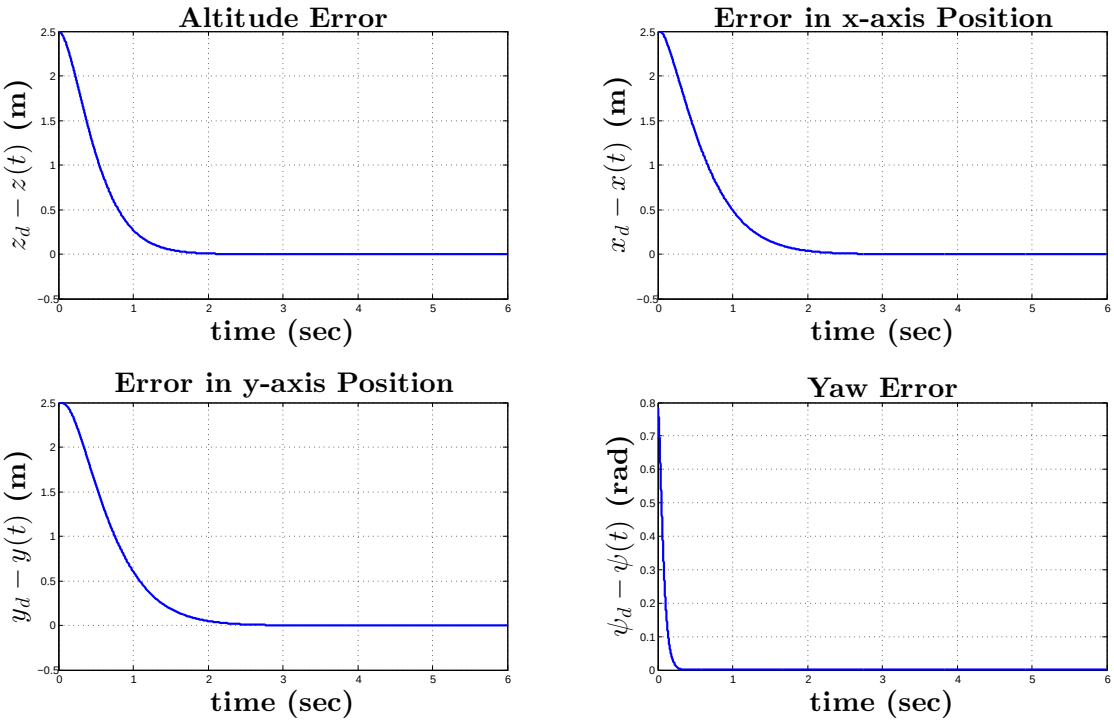


Figure 2.23: Position regulation problem: errors of linear positions and yaw angle for back-stepping controller.

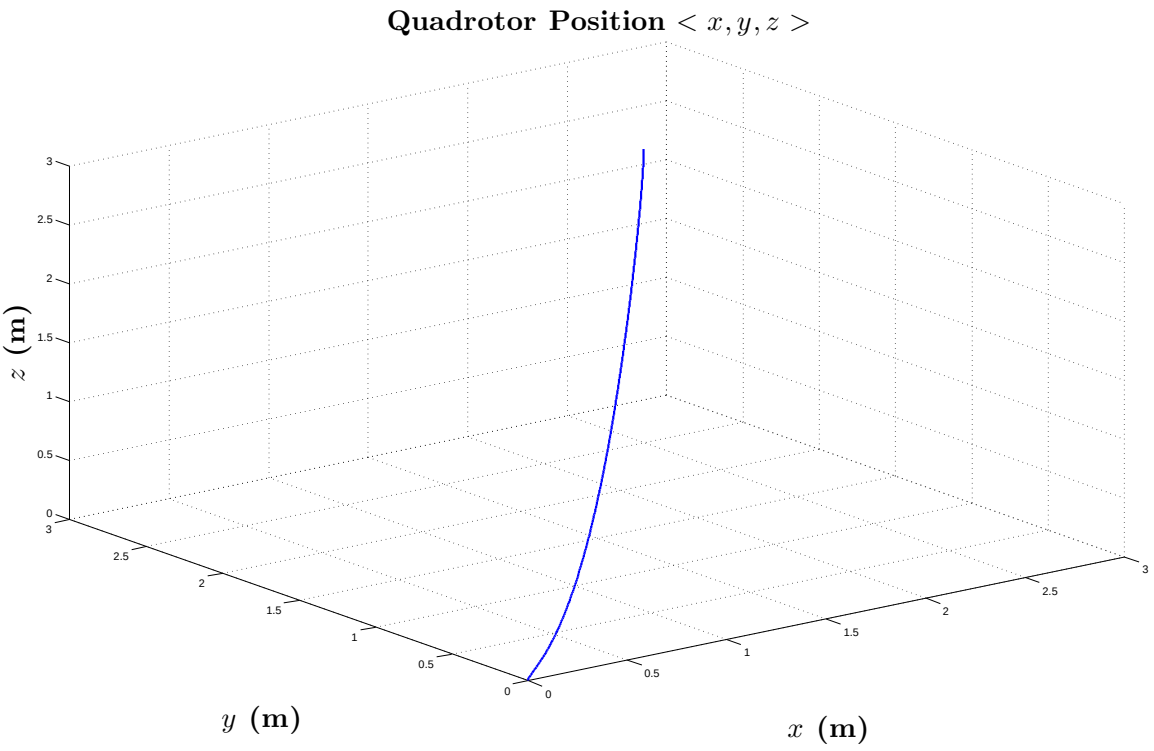


Figure 2.24: Position regulation problem: the position (x, y, z) of the center of the quadrotor's mass for backstepping controller.

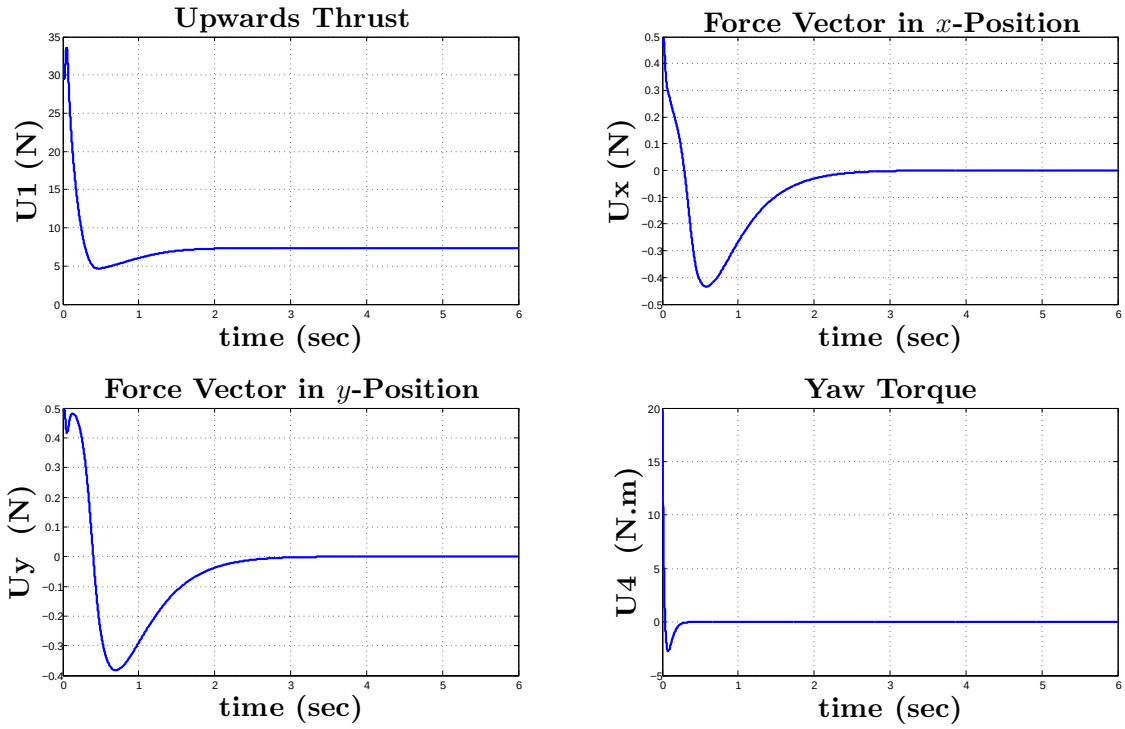


Figure 2.25: The magnitude of control inputs of backspteing controller for attitude control (regulation problem).

function of a_{13}, as follows

$$V(a_{13}) = \frac{1}{2}a_{13}^2. \tag{2.149}$$

The time derivative of Lyapunov function (2.149) is determined as follows

$$\dot{V}(a_{13}) = a_{13}(\ddot{z}_d - \ddot{z}). \tag{2.150}$$

To guarantee the exponential convergence of a_{13}, we would like to impose the following upper bound for $\dot{V}(a_{13})$,

$$\dot{V}(a_{13}) \leq -\kappa_{13}a_{13}^2, \tag{2.151}$$

where $\kappa_{13} > 0$ is a constant. Using (2.150), inequality (2.151) becomes

$$a_{13}(\ddot{z}_d - \ddot{z}) \leq -\kappa_{13} a_{13}^2. \tag{2.152}$$

Inequality (2.152) is guaranteed if the state variable $\ddot{z}$ satisfies

$$\ddot{z} = \kappa_{13} a_{13} + \ddot{z}_d. \tag{2.153}$$

Finally, combining (2.153) and (2.15), the following formula is yielded

$$\begin{aligned} & \frac{\mathbf{U}_1}{m}(\cos\phi\cos\theta) - g = \kappa_{13} a_{13} + \ddot{z}_d \\ \Rightarrow \quad & \frac{\mathbf{U}_1}{m}(\cos\phi\cos\theta) = \kappa_{13} a_{13} + \ddot{z}_d + g \\ & \Rightarrow \quad \mathbf{U}_1 = \frac{m}{\cos\phi\cos\theta}\left(\kappa_{13} a_{13} + \ddot{z}_d + g\right). \end{aligned} \tag{2.154}$$

Setting $\ddot{z}_d = 0$, the linear velocity control input in z-axis that guarantees the asymptotic stability in the sense of Lyapunov is yielded, as follows

$$\boxed{\mathbf{U}_1 = \frac{m}{\cos\phi\cos\theta}\left(\kappa_{13} a_{13} + g\right).} \tag{2.155}$$

The linear velocities control inputs in x–y axes can be acquired using the same line of reasoning as in the derivation of (2.155) , therefore, $\mathbf{U}_x$ and $\mathbf{U}_y$ are

$$\boxed{\mathbf{U}_x = \frac{m}{\mathbf{U}_1}\left(\kappa_{14} a_{14}\right),} \tag{2.156}$$

$$\boxed{\mathbf{U}_y = \frac{m}{\mathbf{U}_1}\left(\kappa_{15} a_{15}\right),} \tag{2.157}$$

where $\kappa_{14}, \kappa_{15} > 0$ are constants, and the variables a_{14} and a_{15} are defined as follows

$$a_{14} := \dot{x}_d - \dot{x}, \tag{2.158}$$

$$a_{15} := \dot{y}_d - \dot{y}, \tag{2.159}$$

where $\dot{x}_d, \dot{y}_d, \dot{x}$ and $\dot{y}$ are the desired and actual linear velocities of the center of the quadrotor's mass in $x-y$-axes, respectively.

Simulation Results for Backstepping Velocity Control

Using odeint C++ library, simulations have been carried out to illustrate the performance of the backstepping velocity controller. Table 2.6 shows the numerical values of the parameters for the simulations, and Figures 2.26, 2.27, and 2.28 show the response of the quadrotor to the regulation task of the linear velocity control.

	initial values	Gains	Desired Trajectory
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$\kappa_{13} = 6.2$	$\dot{z}_d = 2.5$ (m/s)
x	$x(0) = 0$ $\dot{x}(0) = 0$	$\kappa_{14} = 2.68$	$\dot{x}_d = 2.5$ (m/s)
y	$y(0) = 0$ $\dot{y}(0) = 0$	$\kappa_{15} = 3.73$	$\dot{y}_d = 2.5$ (m/s)
Roll	$\phi(0) = 0$ $\dot{\phi}(0) = 0$	$\kappa_3 = 8.5$ $\kappa_4 = 8.0$	
Pitch	$\theta(0) = 0$ $\dot{\theta}(0) = 0$	$\kappa_5 = 8.5$ $\kappa_6 = 8.0$	
Yaw	$\psi(0) = 0$ $\dot{\psi}(0) = 0$	$\kappa_7 = 8.5$ $\kappa_8 = 8.0$	$\psi_d = 0$ (rad)

Table 2.6: Backstepping controller parameters for linear velocity control (regulation task)

2.5 Conclusion

In this chapter, the kinematics, the dynamics, and the control strategy for quadrotor UAV systems have been addressed. Even though there are numerous approaches for kinematic description of rotation of rigid bodies in 3D space, Euler angles parametrization has been chosen due to its relative simplicity. Regarding the control strategy, the design of controllers for both linear

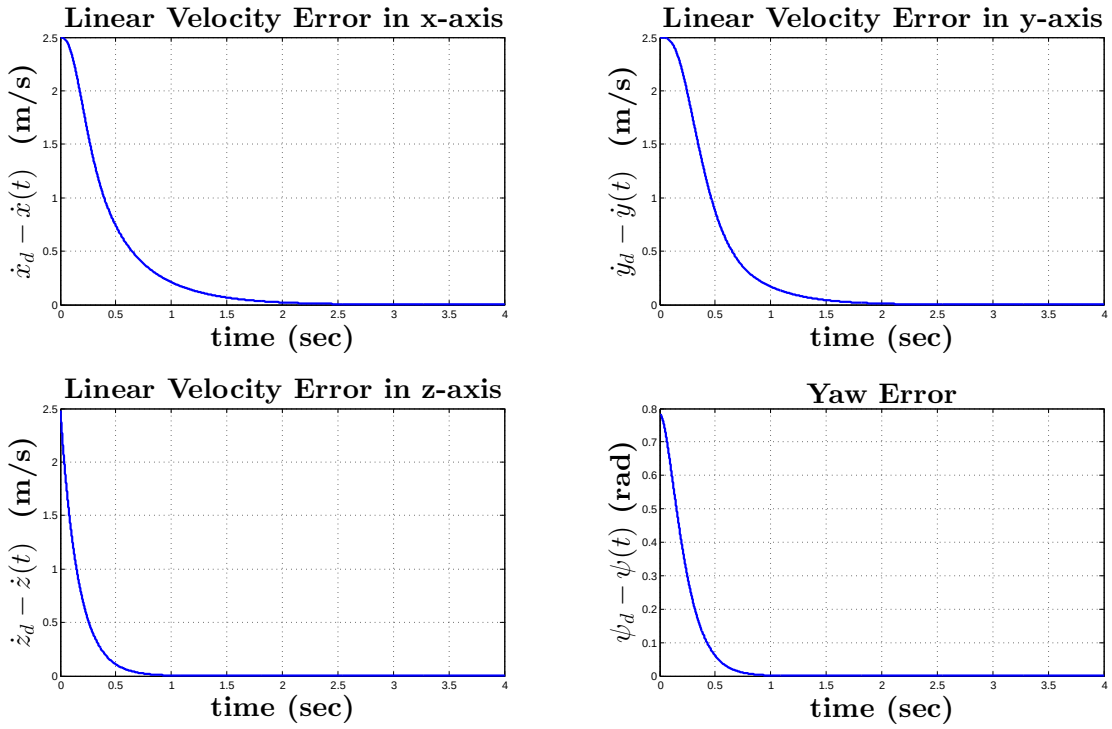


Figure 2.26: Velocity regulation problem: errors of linear velocies and yaw angle for backstepping controller.

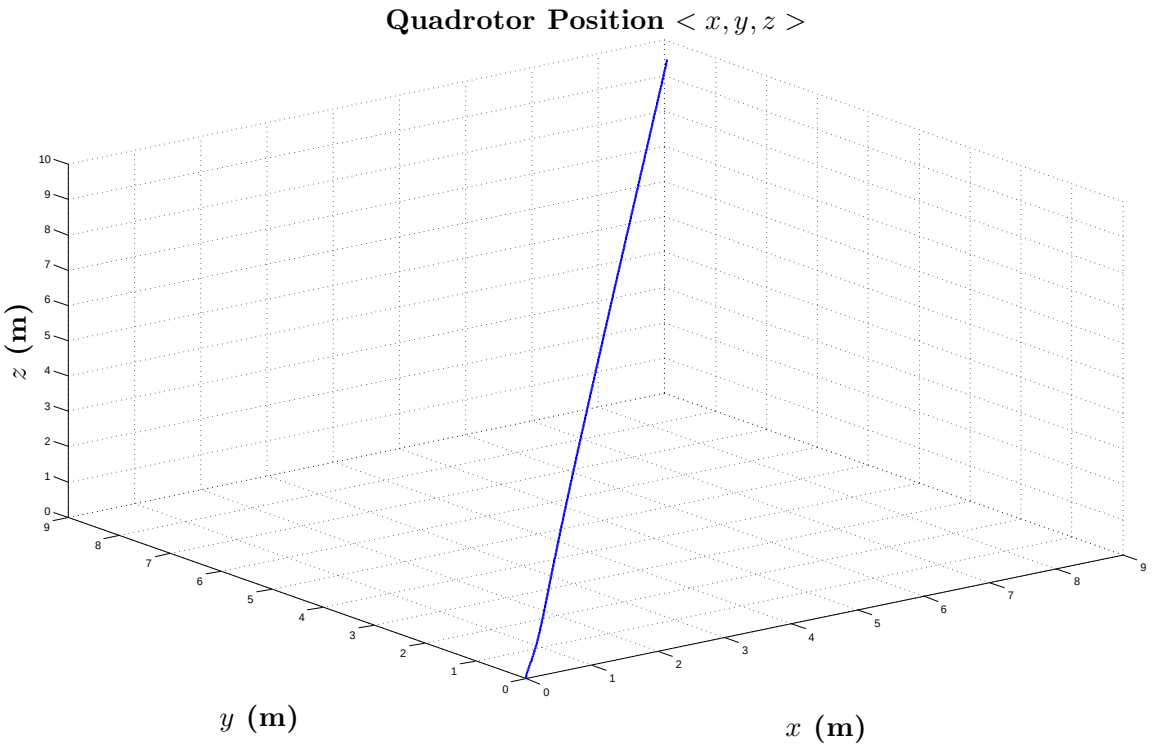


Figure 2.27: Velocity regulation problem: the position (x, y, z) of the center of the quadrotor's mass for backstepping controller.

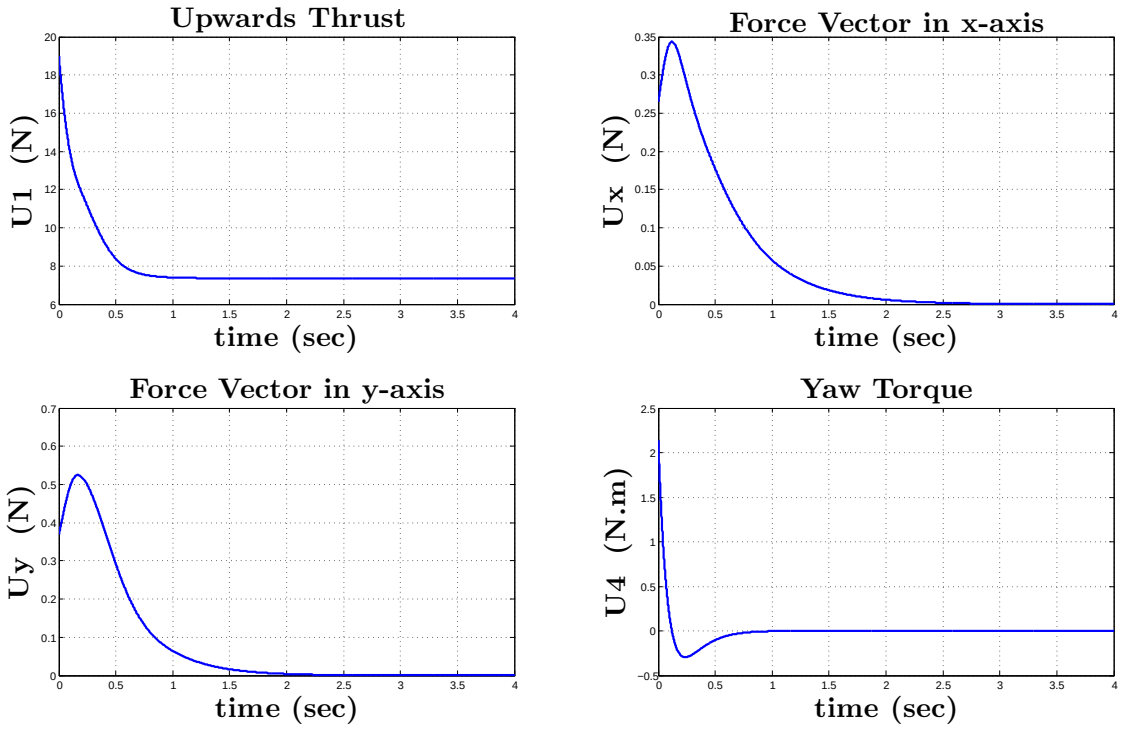


Figure 2.28: The magnitude of control inputs of backspteing controller for velocity control (regulation problem).

and nonlinear models have been described in detail. In the case of a linear model, the small angles assumption is utilized, and PD and PID controllers have been used to stabilize the linearized model of quadrotor for both attitude and position control problems. In the case of a nonlinear model, the backstepping control approach is implemented. Simulations of both attitude and position control problems have been carried out by using C++ odeint library. Moreover, the linear velocity control has been addressed as an alternative to position control. The benefits of using linear velocity control will be further clarified below in Chapter 4 where it will be used to overcome the limitation of haptics device's workspace. This allows quadrotor to cover large distances, which may not be possible using position control. The simulations demonstrated satisfactory performance of the nonlinear backstepping controller, which is therefore chosen to stabilize the quadrotor in the subsequent chapters.

Chapter 3

Simultaneous Localization and Mapping

3.1 Introduction

The problem of Simultaneous Localization and Mapping (SLAM) consists of building a map of an unknown environment while simultaneously determining the location of the robot on this map. SLAM algorithms enable a robot with a capability to estimate its pose in a variety of scenarios, both indoor and outdoor, in cases of static and dynamic environments. The ability of a robot to determine its current location opens the door to a variety of applications. In this thesis, SLAM algorithms will be used to generate haptic feedback in a virtual reality based teleoperator system for remote control of UAVs. In this chapter, SLAM problem is addressed in some detail from a probabilistic perspective. Section 3.2 provides a general description of how SLAM algorithms can be constructed probabilistically using Bayesian framework; also in this section, Section 3.2.2 describes two alternative representations of the metric maps and, in Section 3.2.4, parametric filters are briefly introduced as a means to implement Bayesian filters. Section 3.3 discusses Extended Kalman Filter (EKF) implementation of the SLAM algorithms. A complete EKF-SLAM algorithm for UAV model is presented in Section 3.4.

3.2 Structure of Probabilistic SLAM

3.2.1 Random Variables and Belief Distributions

In this subsection, some background information regarding random variables and belief distributions are given; some basic notions of probability theory are also summarized in Appendix A. A **random variable** is a variable that can be used to describe an outcome of a statistical experiment. For example, in a flipping of a coin experiment, possible outcomes are head and tail, and a random variable $\mathbf{E}$ that describes an outcome of this experiment can be defined as follows:

$$\mathbf{E} = \begin{cases} 1 & \text{if the outcome is head,} \\ 0 & \text{if the outcome is tail.} \end{cases}$$

Random variables are associated with **probabilities**. The probability of the outcome where $\mathbf{E}$ takes a particular numerical value e is denoted by $\mathbf{p}(\mathbf{E} = e)$; for simplicity, notation $\mathbf{p}(e)$ is typically used instead of $\mathbf{p}(\mathbf{E} = e)$. In the **discrete** case, where a random variable e is defined on a discrete set, the probability $\mathbf{p}(e)$ satisfies the following two properties:

1. $0 \leq \mathbf{p}(e) \leq 1$,
2. $\sum\limits_{e} \mathbf{p}(e) = 1$,

where summation in the second formula is performed over all possible values of e. In the **continuous** case, where a random variable e is defined on a continuum of values, $\mathbf{p}(e)$ is called the **probability density function** (PDF) and has the following properties:

1. $0 \leq \mathbf{p}(e)$,
2. $\int \mathbf{p}(e)\, de = 1$,

where, again, the integration in second formula is performed over all possible values of e. One of the most commonly used PDFs is the Gaussian (normal) distribution which is defined via the following formula:

$$\mathbf{p}(e) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2}\frac{(e-\mu)^2}{\sigma^2}\right)}, \tag{3.1}$$

where μ and σ^2 are mean and variance, respectively, of the normal distribution. Gaussian (normal) distribution with mean μ and variance σ^2 is denoted by $\mathcal{N}(\mu, \sigma^2)$.

In experiments that involve multiple random variables, one might be interested to find out whether a given random variable has any effect on the other random variables. The **joint probability** of two random variables **E** and **W** is defined through the following expression,

$$\mathbf{p}(\mathbf{E} = e \text{ and } \mathbf{W} = w).$$

The above expression can be written shortly as $\mathbf{p}(e, w)$. Two random variables **E** and **W** are independent if

$$\mathbf{p}(e, w) = \mathbf{p}(e) \cdot \mathbf{p}(w) \tag{3.2}$$

Frequently, one might be interested to find out the probability that **E** takes a numerical value e given the fact that **W** takes a numerical value w. Such a probability is called **conditional probability**, and denoted by

$$\mathbf{p}(\mathbf{E} = e \mid \mathbf{W} = w)$$

or shortly as $\mathbf{p}(e \mid w)$. The conditional probability $\mathbf{p}(e \mid w)$ can be found through the following formula

$$\mathbf{p}(e \mid w) = \frac{\mathbf{p}(e, w)}{\mathbf{p}(w)} \tag{3.3}$$

whenever $\mathbf{p}(w) > 0$. It follows from (3.2), (3.3) that, if **E** and **W** are independent random variables, their conditional probability $\mathbf{p}(e \mid w) = \mathbf{p}(e)$. In other words, if **E** and **W** are independent, then knowledge of **W** does not give any information regarding the value of **E**, and *vice versa*. The relation between conditional probability $\mathbf{p}(e \mid w)$ and its inverse $\mathbf{p}(w \mid e)$ is determined via the **Bayes rule**, which states

$$\mathbf{p}(e \mid w) = \frac{\mathbf{p}(w \mid e) \cdot \mathbf{p}(e)}{\mathbf{p}(w)},$$

where $\mathbf{p}(w) > 0$. The Bayes rule plays a fundamental role in **Bayes filters**; specifically, it allows for calculating the **posterior probability** $\mathbf{p}(e \mid w)$ based on **prior probability** $\mathbf{p}(e)$, **generative model** $\mathbf{p}(w \mid e)$ and **data** w. The Bayes rule can also be applied to calculate the probability of multiple random variables conditioned on other multiple random variables. For example, the probability that **E** takes a value e can be conditioned on two other randoem variables **W** and **Z**.

In this case, the Bayes rule states

$$\mathbf{p}(e \mid w, z) = \frac{\mathbf{p}(w \mid e, z) \cdot \mathbf{p}(e \mid z)}{\mathbf{p}(w \mid z)}.$$

Theorem of total probability for discrete and continuous cases is formulated as follows:

$$\mathbf{p}(e) = \sum_{w} \mathbf{p}(e \mid w) \cdot \mathbf{p}(w) \qquad \text{(discrete probabilities)},$$

$$\mathbf{p}(e) = \int_{w} \mathbf{p}(e \mid w) \cdot \mathbf{p}(w)\, dw \quad \text{(continuous probabilities)}.$$

The notion of **belief** is defined as an internal knowledge of a robot about its own state, the state of the environment, or both [74]. The belief is represented probabilistically in the form of a posterior probability over state variables conditioned on the available data; specifically, given control inputs $\mathbf{u}^*_{k+1}$ and observations $\mathbf{z}^*_{k+1}$ at a time instant $(k+1)$, the belief (**bel**) over a state variable x^*_{k+1} is described by the formula,

$$\mathbf{bel}(x^*_{k+1}) = \mathbf{p}(x^*_{k+1} \mid \mathbf{z}^*_{1:k+1}, \mathbf{u}^*_{1:k+1}),$$

where the subscript $1:k+1$ denotes the history of all control inputs and observations starting from the first instant of available data up to the current moment $k+1$. The above expression encapsulates the knowledge that the robot possesses about its current state which takes the uncertainty into account. If the PDF of the posterior probability is represented by Gaussian distribution (*i.e.*, $\mathbf{p}(e) = \mathcal{N}(\mu, \sigma^2)$), the belief that the robot possesses about its current state can be represented graphically as a bell curve. Examples of Bell curves are shown in Figure 3.1, where the red curve (**bel 3**) represents belief distribution with lowest uncertainty while the blue curve (**bel 1**) represents belief distribution with highest uncertainty among the three curves.

The robot calculates its belief from measurement and control data. A general algorithm for calculating belief is the Bayes filter algorithm represented below as Algorithm 1. The Bayes filter calculates the belief recursively through two steps which are the **prediction** step and the **measurement update** step. In the first step, the filter calculates a temporary belief $\overline{\mathbf{bel}}(x^*_{k+1})$

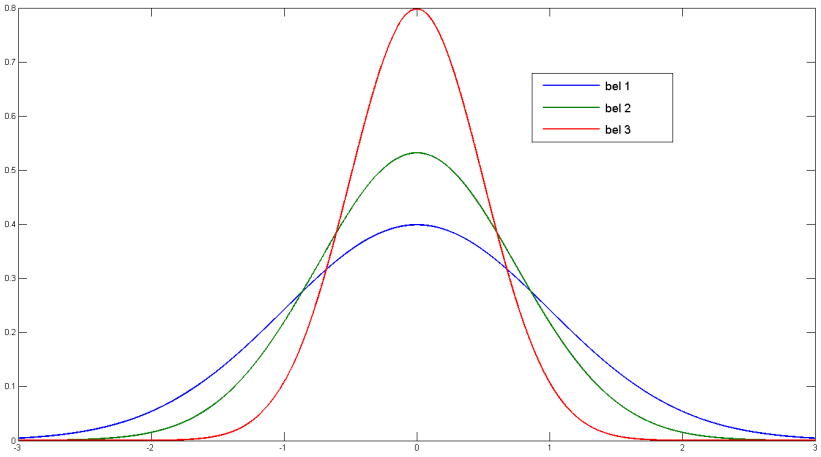


Figure 3.1: Bell curves of multiple belief distributions.

based on previous belief $\mathbf{bel}(x_k^*)$ and control input $\mathbf{u}_{k+1}^*$; the corresponding step is shown in the line 3 of Algorithm 1. Once the robot acquires new observations z_{k+1}^*, the filter carries out the measurement update in which the temporary belief $\overline{\mathbf{bel}}(x_{k+1}^*)$ and the new measurements z_{k+1}^* are utilized to calculate a new belief $\mathbf{bel}(x_{k+1}^*)$ according to line 4 of Algorithm 1.

Algorithm 1 Bayes Filter Algorithm

1: **get** $\{\mathbf{bel}(x_k^*), \mathbf{u}_k^*, \mathbf{z}_k^*\}$
2: **for all** x_{k+1}^* **do**
3: $\quad \overline{\mathbf{bel}}(x_{k+1}^*) = \sum_{x_k^*} \mathbf{p}(x_{k+1}^* \mid x_k^*, \mathbf{u}_{k+1}^*) \cdot \mathbf{bel}(x_k^*)$ (prediction)
4: $\quad \mathbf{bel}(x_{k+1}^*) = \frac{\mathbf{p}(z_{k+1}^* \mid x_{k+1}^*)}{\mathbf{p}(z_{k+1}^*)} \cdot \overline{\mathbf{bel}}(x_{k+1}^*)$ (measurement update)
5: **end for**
6: **return** $\mathbf{bel}(x_{k+1}^*)$

3.2.2 Map Representations

Maps play an essential role in localization and SLAM algorithms. As mentioned in Section 1.2, there exist two types of metric map representations which are feature-based maps and cell-based maps [75]; these are illustrated in Figure 3.2. Each representation has its own advantages and disadvantages. The feature-based map is the representation that is most commonly used in SLAM. It represents the world as a collection of geometric primitives called features (also

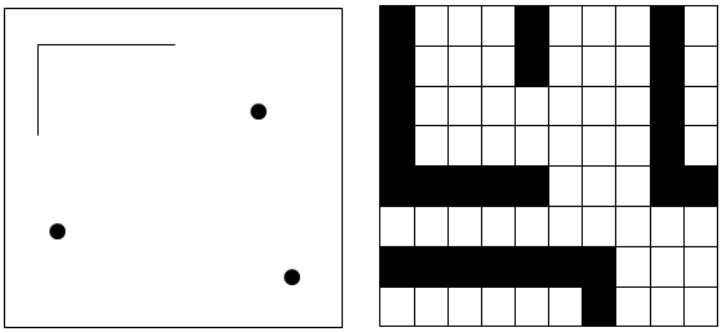


Figure 3.2: Metric map representations: a feature-based map (left); a cell-based map (right).

known as landmarks and beacons). This representation has several advantages. It allows for altering and/or relocating the features [74]; in addition, a group of features can be combined together to form a single feature and subsequently treated as such [76]. Another appealing advantage of using feature-based maps is their ability for encapsulating complex physical objects into geometric ones which leads to lower memory usage [77] and faster computational process [75]. On the other hand, a cell-based map (also known as an occupancy grid) is defined as a matrix of equally spaced cells. Each element of the matrix represents a belief regarding whether the corresponding cell is occupied or empty. This representation provides rich information regarding the environment since it is a volumetric description [74]. Cell-based maps describe both occupied and free spaces which make them suitable for path planning [75]. The empty space between the source of generating sensor signal and the detected obstacle is updated in cell-based map [78] whereas in feature-based map only the location of a feature is updated. Even though cell-base map offers rich information about the environment, the consequence of increasing cell resolutions leads to complexity in the computational process [75] and higher requirements for data storage [77]. In [76] the authors performed a comparison between the two representations and concluded that, for SLAM of UAVs, the feature-based map representation is preferable over the cell-based one. In this thesis, the feature-based representation is utilized, and the map is represented as a vector that consists of Cartesian coordinates of N

obstacles. Specifically, the map is represented as a vector of the form

$$\mathbf{m} = [\underbrace{\mathbf{m}_{1,x}, \mathbf{m}_{1,y}, \mathbf{m}_{1,z}}_{\mathbf{m}_1}, \underbrace{\mathbf{m}_{1,x}, \mathbf{m}_{2,y}, \mathbf{m}_{2,z}}_{\mathbf{m}_2}, \ldots, \underbrace{\mathbf{m}_{N,x}, \mathbf{m}_{N,y}, \mathbf{m}_{N,z}}_{\mathbf{m}_N}]^T$$

Throughout the thesis, it is assumed that the features (obstacles) are stationary and can be geometrically represented as points.

3.2.3 Probabilistic SLAM

As mentioned in Section 3.1, the term "Simultaneous Localization And Mapping" (SLAM) refers to the process of building a map of an unknown environment while simultaneously determining the location of the robot on this map. The most common approaches to the SLAM problem use probabilistic descriptions and methods, as the actual map of the environment and the robot's location on it are almost never precisely known and robot's sensors data are inherently noisy. Consequently, the state of the robot ξ, the state of the map $\mathbf{m}$, the observations $\mathbf{z}$, and the control inputs $\mathbf{u}$ are considered and treated as random variables. Mathematically, SLAM problem involves estimation of the the following posterior probability

$$\mathbf{p}(\boldsymbol{\xi}_\mathbf{k}, \mathbf{m} \mid \mathbf{z}_{1:\mathbf{k}}, \mathbf{u}_{1:\mathbf{k}}, \boldsymbol{\xi}_\mathbf{0}), \tag{3.4}$$

where $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_k$ represent the initial state of the robot and its current state (*i.e.*, the state at the current instant k), respectively. Calculation of the posterior probability (3.4) corresponds to the so-called online SLAM problem, where only the current robot's state $\boldsymbol{\xi}_k$ is considered to be of interest. On the other hand, the so-called full-SLAM problem corresponds to the case where one seeks to estimate the posterior probability that involves the entire history of the robot's path, as follows

$$\mathbf{p}(\boldsymbol{\xi}_{1:\mathbf{k}}, \mathbf{m} \mid \mathbf{z}_{1:\mathbf{k}}, \mathbf{u}_{1:\mathbf{k}}, \boldsymbol{\xi}_\mathbf{0}). \tag{3.5}$$

In this thesis, only the on-line SLAM problem (3.4) is addressed. The posterior probability (3.4) can in principle be calculated recursively using Bayes filter algorithm described above

as Algorithm 1. Initially, the Bayes filter algorithm requires the initial probaility distribution (*i.e.*, the initial belief) of $\boldsymbol{\xi}_0$, $\boldsymbol{m}$. At each iteration, calculation of the posterior probability (3.4) using Bayes filter algorithm consists of two steps. The first step is the prediction, which is formulated as follows,

$$\mathbf{p}(\boldsymbol{\xi}_{\mathbf{k+1}}, \mathbf{m} \mid \mathbf{z}_{\mathbf{1:k}}, \mathbf{u}_{\mathbf{1:k+1}}, \boldsymbol{\xi}_{\mathbf{0}}) = \sum_{\boldsymbol{\xi}_{\mathbf{k}}} \mathbf{p}(\boldsymbol{\xi}_{\mathbf{k+1}} \mid \boldsymbol{\xi}_{\mathbf{k}}, \mathbf{u}_{\mathbf{k+1}}) \cdot \mathbf{p}(\boldsymbol{\xi}_{\mathbf{k}}, \mathbf{m} \mid \mathbf{z}_{\mathbf{1:k}}, \mathbf{u}_{\mathbf{1:k}}, \boldsymbol{\xi}_{\mathbf{0}}). \tag{3.6}$$

The prediction step requires knowledge of the following probability distribution

$$\mathbf{p}(\boldsymbol{\xi}_{k+1} | \boldsymbol{\xi}_k, \mathbf{u}_{k+1}), \tag{3.7}$$

which is called the **motion model** of the robot. The second step of Bayesian filter is the measurement update. This step is executed after the robot acquires observations through its sensory system. This step is formulated as follows,

$$\mathbf{p}(\boldsymbol{\xi}_{\mathbf{k+1}}, \mathbf{m} \mid \mathbf{z}_{\mathbf{1:k+1}}, \mathbf{u}_{\mathbf{1:k+1}}, \boldsymbol{\xi}_{\mathbf{0}}) = \frac{\mathbf{p}(\mathbf{z}_{\mathbf{k+1}} \mid \boldsymbol{\xi}_{\mathbf{k+1}}, \mathbf{m}) \cdot \mathbf{p}(\boldsymbol{\xi}_{\mathbf{k+1}}, \mathbf{m} \mid \mathbf{z}_{\mathbf{1:k}}, \mathbf{u}_{\mathbf{1:k}}, \boldsymbol{\xi}_{\mathbf{0}})}{\mathbf{p}(\mathbf{z}_{\mathbf{k+1}} \mid \mathbf{z}_{\mathbf{1:k}}, \mathbf{u}_{\mathbf{1:k+1}})} \tag{3.8}$$

The measurement update step utilizes the knowledge of the following probability distribution

$$\mathbf{p}(\mathbf{z}_{k+1} | \boldsymbol{\xi}_{k+1}, \mathbf{m}), \tag{3.9}$$

which is called the **observation model**. The detailed mathematical derivation of the recursive Bayesian filter can be found in [74, 79].

3.2.4 Parametric Filters

In most cases, the analytical solution for posterior probability is intractable when using Bayesian filters. Parametric and non-parametric filters approximate the Bayesian filter to provide a tractable solution for SLAM [79]. Parametric filters are the type of filters that are most commonly utilized to handle the SLAM problem. In **parametric filters**, random variables are described using an appropriate parametrization. For example, in Gaussian filters, the moment parametrization describes random variables through their mean and covariance (also known as

the first and second moments). There are numerous types of parametric filters that are used for solving the SLAM problem. The **Extended Kalman Filter** (EKF) is one of the most commonly used parametric Gaussian filter for SLAM. EKF uses the nonlinear function of the motion model to predict the mean, and the linearized version of the motion model to propagate the covariance. The linearization is performed using the first order Taylor series expansion, which may in some cases limit the applicability of EKF. The **Unscented Kalman Filter** (UKF) is another common parametric filter [80] that, similarly to EKF, utilizes the mean and the covariance of the Gaussian. The difference between EKF and UKF is in the way the nonlinearity is handled. EKF linearizes non-linear models using Taylor series whereas UKF uses Unscented Transformation (UT) technique [80]. UKF draws a finite set of points called sigma-points deterministically from Gaussian distribution. These points are propagated through the nonlinear model and, after the transformation process is done, each point is assigned a weight. The weighed sum of these points is utilized to approximate the mean and the covariance according to UKF, see [74] for details and specific algorithms. This technique may lead to improved estimates in the case of nonlinear models in comparison with EKF. Moreover, in contrast with EKF, UKF does not require to calculate derivatives and/or Jacobians. These features make UKF superior to EKF in some cases [81]. A detailed comparison between EKF and UKF can be found in [82, 83, 81]. The **Extended Information Filter** (EIF) is a parametric filter that, similarly to EKF and UKF, describes Gaussian random variables based on two parameters. The difference between EIF and EKF, however, is in the way the Gaussian model is parametrized. EIF does not utilize the mean and the covariance, but rather uses the information matrix and the information vector as a canonical parametrization [74]. The comparison between EIF and EKF can be found in [84], while the actual algorithms for EIF can be found in [74].

3.3 Extended Kalman Filter SLAM Algorithm for a Quadrotor UAV

As mentioned above, the Extended Kalman Filter (EKF) is a parametric filter which is used for SLAM. EKF relies on the assumption that the posterior probability (*i.e.* (3.4)) can be parametrized by the mean and the covariance matrix,

$$\mathbf{p}(\boldsymbol{\xi}_{\mathbf{k+1}}, \mathbf{m} \mid \mathbf{z}_{\mathbf{1:k+1}}, \mathbf{u}_{\mathbf{1:k+1}}, \boldsymbol{\xi}_{\mathbf{0}}) \in \mathcal{N}(\mu_{k+1}, \Sigma_{k+1}) \tag{3.10}$$

where $\mathcal{N}(\mu_{k+1}, \Sigma_{k+1})$ denotes a multivariate Gaussian distribution with mean μ_{k+1} and covariance matrix Σ_{k+1}. EKF recursively calculates μ_k and Σ_{k+1} under the assumption that both the **motion model** (3.7) and **observation model** (3.9) have special form, as described below.

3.3.1 Motion Model

Generally speaking, the motion model describes how the robot is being driven from one state at a discrete time instant k to another state at a discrete time instant $k + 1$. In the case of EKF SLAM, the motion model has a form

$$\boldsymbol{\xi}_{k+1} = g(\boldsymbol{\xi}_k, \mathbf{u}_{k+1}) + \mathbf{w}_{k+1}, \tag{3.11}$$

where $g(\cdot)$ is a nonlinear function that describes the robot kinematics, and $\mathbf{w}$ is the process noise which is required to be Gaussian with zero mean and known covariance (*i.e.* $\mathbf{w}_k \in \mathcal{N}(0, R_k)$). In this thesis, a simplified version of the SLAM problem is addressed where the motion model only deals with translational motion of the quadrotor. The orientation of the quadrotor, on the other hand, is assumed to be perfectly known at any instant of time. Essentially, this means that the quadrotor's local coordinate frame F^L can always be chosen such that its orientation is equal to the orientation of the stationary global frame F^G, see Figure 3.3. From dynamic equations (2.13)-(2.15), the following model is obtained that describes the translational motion

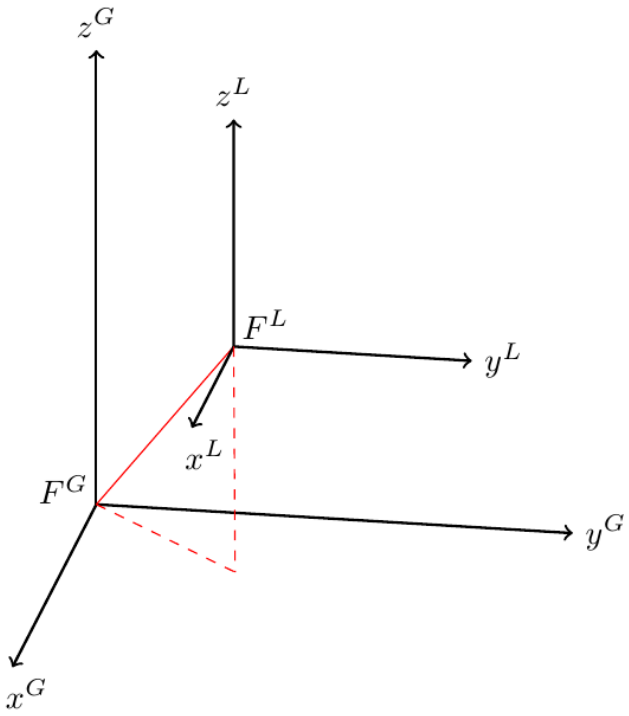


Figure 3.3: Representation of the quadrotor's frame F^L in F^G

of the quadrotor:

$$\ddot{x} = \frac{U_1}{m}(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) := f_x(\phi, \theta, \psi, U_1), \tag{3.12}$$

$$\ddot{y} = \frac{U_1}{m}(\cos\phi \sin\theta \sin\psi - \sin\phi \sin\psi) := f_y(\phi, \theta, \psi, U_1), \tag{3.13}$$

$$\ddot{z} = \frac{U_1}{m}(\cos\phi \cos\theta) - g := f_z(\phi, \theta, U_1), \tag{3.14}$$

where $f_x(\cdot)$, $f_y(\cdot)$ and $f_z(\cdot)$ are nonlinear functions that govern the translational movement of the center of quadrotor's mass in the inertial frame along x, y, and z axes, respectively. Below, the arguments of these functions will be omitted to simplify the notations. Using notation $x_1 := x$, $x_2 := \dot{x}$, $y_1 := y$, $y_2 := \dot{y}$, $z_1 := z$, $z_2 := \dot{z}$, equations (3.12) - (3.14) can be rewritten in the following state-space form

$$\dot{x}_1 = x_2, \tag{3.15}$$

$$\dot{x}_2 = f_x, \tag{3.16}$$

$$\dot{y}_1 = y_2, \tag{3.17}$$

$$\dot{y}_2 = f_y, \tag{3.18}$$

$$\dot{z}_1 = z_2, \tag{3.19}$$

$$\dot{z}_2 = f_z. \tag{3.20}$$

Equations (3.15)- (3.20) can be discretized using Euler approximation to obtain

$$x_{1(k+1)} = x_{1(k)} + x_{2(k)}\Delta t \tag{3.21}$$

$$x_{2(k+1)} = x_{2(k)} + f_x\Delta t \tag{3.22}$$

$$y_{1(k+1)} = y_{1(k)} + y_{2(k+1)}\Delta t, \tag{3.23}$$

$$y_{2(k+1)} = y_{2(k)} + f_y\Delta t, \tag{3.24}$$

$$z_{1(k+1)} = z_{1(k)} + z_{2(k+1)}\Delta t, \tag{3.25}$$

$$z_{2(k+1)} = z_{2(k)} + f_z\Delta t, \tag{3.26}$$

where $\Delta t > 0$ is a sufficiently small sampling period. From (3.21) - (3.26), the motion model (*i.e.* $g(\cdot)$) of the quadrotor's translational motion is

$$\underbrace{\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \\ z_{k+1} \\ \dot{z}_{k+1} \end{bmatrix}}_{\boldsymbol{\xi}_{k+1}} = \underbrace{\begin{bmatrix} x_k + \dot{x}_k\Delta t \\ \dot{x}_k + f_x\Delta t \\ y_k + \dot{y}_k\Delta t \\ \dot{y}_k + f_y\Delta t \\ z_k + \dot{z}_k\Delta t \\ \dot{z}_k + f_z\Delta t \end{bmatrix}}_{g(\boldsymbol{\xi}_k, \mathbf{u}_{k+1})} + \mathbf{w}_{k+1}. \tag{3.27}$$

As mentioned above, the process noise $\mathbf{w}$ is assumed to be Gaussian with zero mean and known covariance R ($\mathbf{w} \in \mathcal{N}(0, R)$), where the covariance matrix has a form

$$R = \begin{bmatrix} \sigma_{xx}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{x}\dot{x}}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{yy}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{y}\dot{y}}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{zz}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{z}\dot{z}}^2 \end{bmatrix} \tag{3.28}$$

Th implementation of EKE requires linearization of $g(\cdot)$ around the current state estimate. The Jacobian matrix of $g(\cdot)$ is

$$G = \frac{\partial g(\cdot)}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ \Delta t \frac{\partial f_x}{\partial x} & 1 + \Delta t \frac{\partial f_x}{\partial \dot{x}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t \frac{\partial f_y}{\partial y} & 1 + \Delta t \frac{\partial f_y}{\partial \dot{y}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & \Delta t \frac{\partial f_z}{\partial z} & 1 + \Delta t \frac{\partial f_z}{\partial \dot{z}} \end{bmatrix} \tag{3.29}$$

where

$$\begin{aligned} f_x &= \frac{U_1}{m} \left(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \right), \\ \frac{\partial f_x}{\partial x} &= 0, \\ \frac{\partial f_x}{\partial \dot{x}} &= 0, \\ f_y &= \frac{U_1}{m} \left(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \right), \\ \frac{\partial f_y}{\partial y} &= 0, \end{aligned}$$

$$
\begin{aligned}
\frac{\partial f_y}{\partial \dot{y}} &= 0, \\
f_z &= \frac{U_1}{m}(\cos\phi\cos\theta) - g, \\
&= \Big[(z_d - z) - \kappa_1\Big(\dot{z} - \dot{z}_d - \kappa_1(z_d - z) + \kappa_1(z_d - z)\Big) - \kappa_2(\dot{z} - \dot{z}_d - \kappa_1 z_d + \kappa_1 z)\Big], \\
\frac{\partial f_z}{\partial z} &= \Big[(0\ -1) - \kappa_1\Big(0 - 0 - \cancel{\kappa_1(z_d - z)} + \cancel{\kappa_1(z_d - z)}\Big) - \kappa_2(0 - 0 - 0 + \kappa_1)\Big], \\
&= [-1 - \kappa_2\kappa_1], \\
\frac{\partial f_z}{\partial \dot{z}} &= \Big[(0\ -0) - \kappa_1\Big(1 - 0 - \cancel{\kappa_1(z_d - z)} + \cancel{\kappa_1(z_d - z)}\Big) - \kappa_2(1 - 0 - 0 + 0)\Big] \\
&= [-\kappa_1 - \kappa_2].
\end{aligned}
$$

3.3.2 Observation Model

The observation model (3.9) describes probability distribution of observations conditioned on the robot location and the state of the map. In this thesis, the quadrotor is assumed to be equipped with a 3D sensor that provides the range r, the azimuthal angle b_θ and the polar angle b_ϕ to obstacles, and is subject to observation noise. The primary purpose of the observation model is to generate the predicted observations $\hat{\mathbf{z}}$, so that the difference between the actual $\mathbf{z}$ and the predicted $\hat{\mathbf{z}}$ observations (*i.e.* $\mathbf{z} - \hat{\mathbf{z}}$) forms an **innovation**. The observation vector $\mathbf{z}^i$ has the following form,

$$
\mathbf{z}^i = \begin{bmatrix} r^i \\ b^i_\theta \\ b^i_\phi \end{bmatrix}, \ i = 1, 2, ..., n, \tag{3.30}
$$

where

$$
q = (\mathbf{m}_{j,x} - \boldsymbol{\xi}_x)^2 + (\mathbf{m}_{j,y} - \boldsymbol{\xi}_y)^2 + (\mathbf{m}_{j,z} - \boldsymbol{\xi}_z)^2, \tag{3.31}
$$

$$
r = \sqrt{q}, \tag{3.32}
$$

$$
b_\theta = \tan^{-1}\left(\frac{\mathbf{m}_{j,y} - \boldsymbol{\xi}_y}{\mathbf{m}_{j,x} - \boldsymbol{\xi}_x}\right), \tag{3.33}
$$

$$b_\phi = \cos^{-1}\left(\frac{\mathbf{m}_{j,z} - \boldsymbol{\xi}_z}{r}\right). \tag{3.34}$$

and n is the total number of observations for a given scan. It can be seen from the above

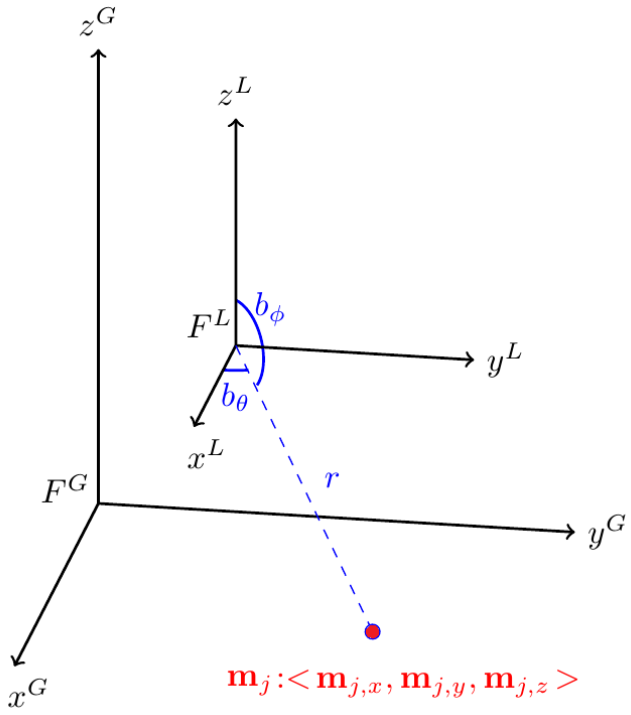


Figure 3.4: 3D sensor that provides observations to a beacon

equations that the observations depend on the quadrotor's position and the location of a beacon, therefore, a general description of the observation model is formulated as follows,

$$\mathbf{z}(k+1) = h(\boldsymbol{\xi}(k+1), \mathbf{m}_j) + \mathbf{v}(k+1) \tag{3.35}$$

where $\mathbf{v}$ is the measurement noise which is assumed to be Gaussian with zero mean and some known covariance Q (i.e. $\mathbf{v} \in \mathcal{N}(0, Q)$). EKF utilizes a linearized version of the observation model (3.35). The Jacobian H of the observation model $h(\cdot)$ has the following form,

$$H = \frac{\partial h(.)}{\partial \mathbf{x}} = \begin{bmatrix} -\frac{\mathrm{Dx}}{r} & 0 & -\frac{\mathrm{Dy}}{r} & 0 & -\frac{\mathrm{Dz}}{r} & 0 & \frac{\mathrm{Dx}}{r} & \frac{\mathrm{Dy}}{r} & \frac{\mathrm{Dz}}{r} \\ \frac{\mathrm{Dy}}{\mathrm{Dy}^2+\mathrm{Dx}^2} & 0 & -\frac{\mathrm{Dx}}{\mathrm{Dy}^2+\mathrm{Dx}^2} & 0 & 0 & 0 & -\frac{\mathrm{Dy}}{\mathrm{Dy}^2+\mathrm{Dx}^2} & \frac{\mathrm{Dx}}{\mathrm{Dy}^2+\mathrm{Dx}^2} & 0 \\ -\frac{\mathrm{DzDx}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}(q)^{3/2}} & 0 & -\frac{\mathrm{DzDy}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}(q)^{3/2}} & 0 & \frac{\frac{1}{\sqrt{q}}-\frac{\mathrm{Dz}^2}{(q)^{3/2}}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}} & 0 & \frac{\mathrm{DzDx}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}(q)^{3/2}} & \frac{\mathrm{DzDy}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}(q)^{3/2}} & -\frac{\frac{1}{\sqrt{q}}-\frac{\mathrm{Dz}^2}{(q)^{3/2}}}{\sqrt{1-\frac{\mathrm{Dz}^2}{q}}} \end{bmatrix} \tag{3.36}$$

where

$$\mathrm{Dx} = \mathbf{m}_{j,x} - \boldsymbol{\xi}_x \,,$$

$$\mathrm{Dy} = \mathbf{m}_{j,y} - \boldsymbol{\xi}_y \,,$$

$$\mathrm{Dz} = \mathbf{m}_{j,z} - \boldsymbol{\xi}_z \,.$$

The zero columns in H reflect the fact that the measurements are not affected by the translational velocities of the quadrotor. Finally, the covariance matrix Q of the observations is assumed to have the following form,

$$Q = \begin{bmatrix} \sigma_{rr}^2 & 0 & 0 \\ 0 & \sigma_{b_\theta b_\theta}^2 & 0 \\ 0 & 0 & \sigma_{b_\phi b_\phi}^2 \end{bmatrix} \tag{3.37}$$

.

3.3.3 Data Association

For each actual observation $\mathbf{z}^i$, the robot must decide whether an observation belongs to a new beacon, to the one that has been observed before, or is due to false alarm. This is done through a procedure known as **data association** which must be made before fusing the location of the beacon into the state vector. For the sake of simplicity, the case of false alarms is not addressed in this thesis.

Data association is one of the most difficult and critical parts of SLAM. Failure in data asso-

ciation step may result in adding incorrect beacons to the state vector which may make it grow unboundedly, consequently leading to divergence of the EKF-SLAM algorithm. There exist numerous techniques for data association. One of the most common methods is the nearest neighbour (NN) method. This method is applicable if the uncertainty of the location of beacons is low [74]. NN approach uses maximum likelihood (ML) estimation to determine whether an actual observation represents a new landmark or a landmark that has been observed before. The idea behind the NN technique is to compute the squared **Mahalanobis distance** [85] between the actual and the predicted observations. It is defined as follows,

$$D^2(\mathbf{z}^i, \hat{\mathbf{z}}^p) := (\mathbf{z}^i - \hat{\mathbf{z}}^p)^T \Psi^{-1} (\mathbf{z}^i - \hat{\mathbf{z}}^p), \tag{3.38}$$

where D^2 and Ψ are the squared Manalanobis distance and the innovation covariance matrix, respectively. The two indexes i and p correspond to the number of the actual observation per scan and the number of the existing beacon in the state vector. In NN method, the computed distances (3.38) are passed through a validation gate to establish the correspondence between the actual observation and the known beacons. The validation gate is implemented according to the formula

$$D^2 \leq \chi^2_{\tau,\alpha} \tag{3.39}$$

where χ^2 is the validation gate threshold (obtained from Chi-square distribution table), where τ is the rank of the predicted observation vector [86] and α is the confidence level. If the inequality (3.39) does not hold then the observation is assumed to belong to a new beacon; otherwise, the observation is associated with an existing beacon. In the latter case, the correspondence between the actual and the predicted observations is established to determine the existing beacon's index, as follows,

$$p(i) = \underset{p}{\operatorname{argmin}}(D^2(\mathbf{z}^i, \hat{\mathbf{z}}^p)). \tag{3.40}$$

3.3.4 Constructing EKF-SLAM

The EKF-SLAM approach in this thesis is adopted from [74]; the difference, however, is that in our case the SLAM for UAV is addressed, whereas in [74] the platform is a ground mobile robot that follows a circular path in a planar environment. The primary objective of EKF-SLAM algorithm is to keep the estimates of system's state and uncertainties updated based on control inputs and measurements. The estimate (or the mean) of the state vector $\hat{\mathbf{x}}$ consists of the robot's state vector $\hat{\boldsymbol{\xi}}$ and the map's state vector $\hat{\mathbf{m}}$, where the hat indicates that these quantities are estimates. More precisely, the state vector is as follows

$$\hat{\mathbf{x}} = [\hat{\boldsymbol{\xi}}\ \hat{\mathbf{m}}]^T \quad \in \mathbb{R}^{(6+3N)\times 1} \tag{3.41}$$

where

$$\hat{\boldsymbol{\xi}} = (x,\ \dot{x},\ y,\ \dot{y},\ z,\ \dot{z}) \quad \in \mathbb{R}^{6\times 1}, \tag{3.42}$$

$$\hat{\mathbf{m}} = (\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2, ..., \hat{\mathbf{m}}_N) \quad \in \mathbb{R}^{3N\times 1}. \tag{3.43}$$

The dimension of the state vector $\hat{\mathbf{x}}$ is $(6 + 3N) \times 1$, where N is the total number of beacons in the map's state vector. From Figure 3.4, $\mathbf{m}_j$ is a vector that holds the Cartesian coordinates of jth beacon, hence $\mathbf{m}_j = (\mathbf{m}_{j,x}, \mathbf{m}_{j,y}, \mathbf{m}_{j,z}) \quad \in \mathbb{R}^{3\times 1}$; its counterpart in EKF that represents its mean is $\hat{\mathbf{m}}_j$. The covariance matrix $\mathbf{P}$ reflects the uncertainty of the state vector estimate; its dimension is (6+3N)×(6+3N), and it depends quadratically on the number of the existing beacons. The covariance matrix $\mathbf{P}$ has a form

$$P = \begin{bmatrix} P_{\hat{\xi}\hat{\xi}} & P_{\hat{\xi}\hat{\mathbf{m}}_1} & \dots & P_{\hat{\xi}\hat{\mathbf{m}}_N} \\ P_{\hat{\mathbf{m}}_1\hat{\xi}} & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_1} & \dots & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{\hat{\mathbf{m}}_N\hat{\xi}} & P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_1} & \dots & P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_N} \end{bmatrix} \in \mathbb{R}^{(6+3N)\times(6+3N)}. \tag{3.44}$$

The covariance matrix (3.44) can be written in a more compact form, as follows

$$P = \begin{bmatrix} P_{\hat{\xi}\hat{\xi}} & P_{\hat{\xi}\hat{\mathbf{m}}} \\ P_{\hat{\mathbf{m}}\hat{\xi}} & P_{\hat{\mathbf{m}}\hat{\mathbf{m}}} \end{bmatrix}. \tag{3.45}$$

It can be seen from (3.45) that the covariance matrix $\mathbf{P}$ consists of four submatrices, where $P_{\hat{\xi}\hat{\xi}}$ is a covariance matrix that reflects the uncertainty in the robot's state vector and the correlation between its elements, $P_{\hat{\mathbf{m}}\hat{\mathbf{m}}}$ is a covariance matrix that reflects the uncertainty in beacons' locations and the correlation between their elements, and $P_{\hat{\xi}\hat{\mathbf{m}}} = P^T_{\hat{\mathbf{m}}\hat{\xi}}$ are covariance matrices that reflect the uncertainty between the robot's state vector and the location of existing beacons.

EKF Prediction Step

The prediction step is the first step of EKF algorithm that calculates how the estimate is driven from a given state to the next one. The estimate $\hat{\mathbf{x}}_{k+1|k}$ represents a ***prior*** state estimate of the system and $P_{k+1|k}$ represents its uncertainty. For the sake of simplicity, we assume that control inputs are not corrupted by noise, however, this is not always the case. If the control inputs themselves are corrupted by noise, then their covariance matrix estimate must be augmented, see [74]. If the ontrol inputs are not corrupted by noise, the predicting step consists of the following two equations,

$$\hat{\mathbf{x}}_{k+1|k} = g(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_{k+1}), \tag{3.46}$$

$$P_{k+1|k} = G_{k+1} P_{k|k} G^T_{k+1} + R_{k+1}, \tag{3.47}$$

where G_{k+1} and R_{k+1} are the "augmented" versions of the matrices (3.29) and (3.28), respectively. The augmentation process corresponds to adding new beacons to the map; it will be described below in "Adding a new beacon" subsection.

EKF Update Step

Update step is the second step of EKF algorithm. In this step, the filter is provided with the actual observations, and the predicted observations are generated using the observation model (3.35) using the existing state estimates. The innovation is then calculated and multiplied by the Kalman Gain K_{k+1}, the latter is calculated according to equation (3.48) below. The result is added to a prior estimate to yield a new ***posterior*** state estimate ($\hat{\mathbf{x}}_{k+1|k+1}$) in (3.49). The updated covariance is subsequently computed in (3.50). The algorithm has a form

$$K_{k+1} = P_{k+1|k} H_{k+1}^T (H_{k+1} P_{k+1|k} H_{k+1}^T + Q_{k+1})^{-1}, \tag{3.48}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + K_{k+1}(\mathbf{z} - \underbrace{h(\hat{\boldsymbol{\xi}}_{k+1|k}, \hat{\mathbf{m}}_j)}_{\hat{\mathbf{z}}}), \tag{3.49}$$

$$P_{k+1|k+1} = (I - K_{k+1} H_{k+1}) P_{k+1|k}, \tag{3.50}$$

where I is the identity matrix. Kalman gain K_{k+1} is essentially a measure to which the posterior estimate should rely on the actual observations to enhance the prior estimate.

System Initialization

The starting point from which the robot begins its operation is considered as the origin of the global frame ${}^{O}F^{G}$. The robot starts with an empty map. Therefore, the initial values of the estimate $\hat{\boldsymbol{\xi}}$ and the covariance matrix $P_{\hat{\xi}\hat{\xi}}$ of the robot are set to zero, the latter reflects the fact that the robot has zero uncertainty about its state in the beginning,

$$\hat{\boldsymbol{\xi}}(0) = 0 \quad \in \mathbb{R}^{6\times 1}, \tag{3.51}$$

$$P_{\hat{\xi}\hat{\xi}}(0) = 0 \quad \in \mathbb{R}^{6\times 6}. \tag{3.52}$$

Adding a new beacon

The robot gradually constructs the map by adding new beacons to it. Once a new beacon is recognized, an estimate of its position on the map can be obtained through the inverse mea-

surement function [74] which, for given measurements in spherical coordinates, calculates the corresponding position in Cartesian coordinates,

$$\mathbf{m}_{j(k+1)} = \boldsymbol{\Omega}(\boldsymbol{\xi}_{k+1}, \mathbf{z}^i_{k+1}). \tag{3.53}$$

The inverse measurement function $\boldsymbol{\Omega}(\cdot)$ has a form

$$\mathbf{m}_{j,x} = \boldsymbol{\xi}_x + r^i \cos(b^i_\theta) \sin(b^i_\phi), \tag{3.54}$$

$$\mathbf{m}_{j,y} = \boldsymbol{\xi}_y + r^i \sin(b^i_\theta) \sin(b^i_\phi), \tag{3.55}$$

$$\mathbf{m}_{j,z} = \boldsymbol{\xi}_z + r^i \cos(b^i_\phi). \tag{3.56}$$

In EKF-SLAM, once the robot observes a new beacon, the size of $\hat{\mathbf{x}}$ and P grows accordingly. The estimate of the new beacon's location in the global frame is acquired through the inverse measurement function:

$$\hat{\mathbf{m}}_{j(k+1)} = \boldsymbol{\Omega}(\hat{\boldsymbol{\xi}}_{k+1}, \mathbf{z}^i_{k+1}). \tag{3.57}$$

Subsequently, the estimate of the state vector $\hat{\mathbf{x}}$ is augmented as follows

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\boldsymbol{\xi}} \\ \hat{\boldsymbol{m}}_1 \\ \vdots \\ \hat{\boldsymbol{m}}_{j-1} \\ \hat{\boldsymbol{m}}_j \end{bmatrix} \tag{3.58}$$

The covariance matrix P also needs to be augmented. Before the new beacon is added, the covariance matrix has a form

$$P = \begin{bmatrix} P_{\hat{\boldsymbol{\xi}}\hat{\boldsymbol{\xi}}} & \overbrace{P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}_1} \quad \cdots \quad P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}_N}}^{P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}}} \\ P_{\hat{\mathbf{m}}_1\hat{\boldsymbol{\xi}}} & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_1} \quad \cdots \quad P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_N} \\ \vdots & \vdots \quad \ddots \quad \vdots \\ P_{\hat{\mathbf{m}}_N\hat{\boldsymbol{\xi}}} & P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_1} \quad \cdots \quad P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_N} \end{bmatrix}.$$

The dark red block represents the uncertainty of the robot's state vector, and the light red blocks represent its cross-covariance matrix with the existing beacons in the map's vector. The covariance matrix of the new beacon $P_{\hat{\mathbf{m}}_{N+1}\hat{\mathbf{m}}_{N+1}}$ is acquired as follows,

$$P_{\hat{\mathbf{m}}_{N+1}\hat{\mathbf{m}}_{N+1}} = G_{\hat{\xi}} P_{\hat{\xi}\hat{\xi}} G_{\hat{\xi}}^T + G_z Q G_z^T \tag{3.59}$$

where $G_{\hat{\xi}}$ is the Jacobian matrix of the inverse measurement function (3.53) with respect to the robot's state vector,

$$G_{\hat{\xi}} := \frac{\partial \mathbf{\Omega}(.)}{\partial \hat{\boldsymbol{\xi}}} = \begin{bmatrix} G_{\hat{\xi}11} & G_{\hat{\xi}12} & G_{\hat{\xi}13} & G_{\hat{\xi}14} & G_{\hat{\xi}15} & G_{\hat{\xi}16} \\ G_{\hat{\xi}21} & G_{\hat{\xi}22} & G_{\hat{\xi}23} & G_{\hat{\xi}24} & G_{\hat{\xi}25} & G_{\hat{\xi}26} \\ G_{\hat{\xi}31} & G_{\hat{\xi}32} & G_{\hat{\xi}33} & G_{\hat{\xi}34} & G_{\hat{\xi}35} & G_{\hat{\xi}36} \end{bmatrix}_{3\times 6}, \tag{3.60}$$

where the elements of $G_{\hat{\xi}}$ are,

$$G_{\hat{\xi}11} = \frac{r\hat{\boldsymbol{\xi}}_y \sin(b_\theta)\sin(b_\phi)}{(\hat{\boldsymbol{\xi}}_y^2 + \hat{\boldsymbol{\xi}}_x^2)} + \frac{r\hat{\boldsymbol{\xi}}_x\hat{\boldsymbol{\xi}}_z \cos(b_\theta)\cos(b_\phi)}{\left(\sqrt{1-\frac{\hat{\xi}_z^2}{q}}\right) q^{3/2}} + 1 \ ,$$

$$G_{\hat{\xi}13} = \frac{r\hat{\boldsymbol{\xi}}_y\hat{\boldsymbol{\xi}}_z \cos(b_\theta+)\cos(b_\phi)}{\left(\sqrt{1-\frac{\hat{\xi}_z^2}{q}}\right) q^{3/2}} - \frac{r\hat{\boldsymbol{\xi}}_x \sin(b_\theta)\sin(b_\phi)}{(\hat{\boldsymbol{\xi}}_y^2 + \hat{\boldsymbol{\xi}}_x^2)} \ ,$$

$$G_{\hat{\xi}51} = -\frac{r\cos(b_\theta + R_\theta)\cos(b_\phi + R_\phi)\left(\frac{1}{r} - \frac{\hat{\xi}_z^2}{q^{3/2}}\right)}{\left(\sqrt{1-\frac{\hat{\xi}_z^2}{q}}\right)} \ ,$$

$$G_{\hat{\xi}12} = G_{\hat{\xi}14} = G_{\hat{\xi}16} = 0$$

$$G_{\hat{\xi}21} = \frac{r\hat{\boldsymbol{\xi}}_x\hat{\boldsymbol{\xi}}_z \sin(b_\theta)\cos(b_\phi)}{\left(\sqrt{1-\frac{\hat{\xi}_z^2}{q}}\right) q^{3/2}} - \frac{r\hat{\boldsymbol{\xi}}_y \cos(b_\theta)\sin(b_\phi)}{(\hat{\boldsymbol{\xi}}_y^2 + \hat{\boldsymbol{\xi}}_x^2)} \ ,$$

$$G_{\hat{\xi}23} = \frac{r\hat{\boldsymbol{\xi}}_x \cos(b_\theta)\sin(b_\phi)}{(\hat{\boldsymbol{\xi}}_y^2 + \hat{\boldsymbol{\xi}}_x^2)} + \frac{r\hat{\boldsymbol{\xi}}_y\hat{\boldsymbol{\xi}}_z \sin(b_\theta)\cos(b_\phi)}{\left(\sqrt{1-\frac{\hat{\xi}_z^2}{q}}\right) q^{3/2}} + 1 \ ,$$

$$G_{\hat{\xi}25} = -\frac{r\sin(b_\theta)\cos(b_\phi)\left(\frac{1}{r} - \frac{\hat{\xi}_z^2}{q^{3/2}}\right)}{\left(\sqrt{1 - \frac{\hat{\xi}_z^2}{q}}\right)} \ ,$$

$$G_{\hat{\xi}22} = G_{\hat{\xi}24} = G_{\hat{\xi}26} = 0 \ ,$$

$$G_{\hat{\xi}31} = -\frac{r\hat{\boldsymbol{\xi}}_x\hat{\boldsymbol{\xi}}_z\sin(b_\phi)}{\left(\sqrt{1 - \frac{\hat{\xi}_z^2}{q}}\right)q^{3/2}} \ ,$$

$$G_{\hat{\xi}33} = -\frac{r\hat{\boldsymbol{\xi}}_y\hat{\boldsymbol{\xi}}_z\sin(b_\phi)}{\left(\sqrt{1 - \frac{\hat{\xi}_z^2}{q}}\right)q^{3/2}} \ ,$$

$$G_{\hat{\xi}35} = \frac{r\sin(b_\theta)\left(\frac{1}{r} - \frac{\hat{\xi}_z^2}{q^{3/2}}\right)}{\left(\sqrt{1 - \frac{\hat{\xi}_z^2}{q}}\right)} + 1 \ ,$$

$$G_{\hat{\xi}32} = G_{\hat{\xi}34} = G_{\hat{\xi}36} = 0.$$

The Jacobian matrix G_z of the inverse measurement function (3.53) with respect to the actual observation vector (*i.e.* $\mathbf{z}$) is obtained as follows:

$$G_z = \frac{\partial \boldsymbol{\Omega}(.)}{\partial \mathbf{z}} = \begin{bmatrix} G_{z11} & G_{z12} & G_{z13} \\ G_{z21} & G_{z22} & G_{z23} \\ G_{z31} & G_{z32} & G_{z33} \end{bmatrix} \tag{3.61}$$

where the parameters of G_z are,

$$G_{z11} = \cos(b_\theta)\sin(b_\phi),$$

$$G_{z12} = -r\sin(b_\theta)\sin(b_\phi),$$

$$G_{z13} = r\cos(b_\theta)\cos(b_\phi),$$

$$Gy_{21} = \sin(b_\theta)\sin(b_\phi),$$

$$G_{z22} = r\cos(b_\theta)\sin(b_\phi),$$

$$G_{z23} = r\sin(b_\theta)\cos(b_\phi),$$

$$G_{z31} = \cos(b_\phi),$$

$$G_{z32} = 0,$$

$$G_{z33} = -r\sin(b_\phi).$$

The cross-covariance matrix P_Λ of the new beacon's can now be computed as follows:

$$P_\Lambda = [G_{\hat{\boldsymbol{\xi}}} P_{\hat{\boldsymbol{\xi}}\hat{\boldsymbol{\xi}}} \vdots G_{\hat{\boldsymbol{\xi}}} P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}}] \quad \in \mathbb{R}^{3\times(6+3N)} \tag{3.62}$$

Finally, the augmented covariance matrix becomes

$$P = \begin{bmatrix} P_{\hat{\boldsymbol{\xi}}\hat{\boldsymbol{\xi}}} & P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}_1} & \cdots & P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}_N} & P_{\hat{\boldsymbol{\xi}}\hat{\mathbf{m}}_{N+1}} \\ P_{\hat{\mathbf{m}}_1\hat{\boldsymbol{\xi}}} & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_1} & \cdots & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_N} & P_{\hat{\mathbf{m}}_1\hat{\mathbf{m}}_{N+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{\hat{\mathbf{m}}_N\hat{\boldsymbol{\xi}}} & P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_1} & \cdots & P_{\hat{\mathbf{m}}_N\hat{\mathbf{m}}_N} & \vdots \\ P_{\hat{\mathbf{m}}_{N+1}\hat{\boldsymbol{\xi}}} & P_{\hat{\mathbf{m}}_{N+1}\hat{\mathbf{m}}_1} & \cdots & \cdots & P_{\hat{\mathbf{m}}_{N+1}\hat{\mathbf{m}}_{N+1}} \end{bmatrix}. \tag{3.63}$$

(P_Λ under the first block column; P_Λ^T over the last block column)

3.4 EKF-SLAM Algorithm

For simplicity, EKF-SLAM algorithm is divided below into two parts. **Algorithm 2** corresponds to the prediction step; it shows how the estimate of the robot's state vector $\hat{\boldsymbol{\xi}}$ and its uncertainty are being updated. The matrix Π is utilized to map the corresponding submatrices to higher dimensions, so that the dimensions of matrices are compatible for multiplication [74].

Algorithm 2 Prediction Step of EKF-SLAM

1: **EKF-SLAM** $(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}, \mathbf{u}_{k+1}, \mathbf{z}_{k+1})$

2: $N_{k+1} = N_k$

3: $\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0\cdots0 \end{bmatrix}_{6\times(6+3N)}$

4: $\hat{\mathbf{x}}_{k+1|k} = \hat{\mathbf{x}}_{k|k} + \Pi_x^T \begin{bmatrix} \boldsymbol{\xi}_{\dot{x}(k+1)}\Delta t \\ f_x \Delta t \\ \boldsymbol{\xi}_{\dot{y}(k+1)}\Delta t \\ f_y \Delta t \\ \boldsymbol{\xi}_{\dot{z}(k+1)}\Delta t \\ f_y \Delta t \end{bmatrix}$

5: $G_{k+1} = I + \Pi_x^T \begin{bmatrix} 0 & \Delta t & 0 & 0 & 0 & 0 \\ \Delta t \frac{\partial f_x}{\partial x} & \Delta t \frac{\partial f_x}{\partial \dot{x}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t \frac{\partial f_y}{\partial y} & \Delta t \frac{\partial f_y}{\partial \dot{y}} & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & \Delta t \frac{\partial f_z}{\partial z} & \Delta t \frac{\partial f_z}{\partial \dot{z}} \end{bmatrix} \Pi$

6: $\mathbf{P}_{k+1|k} = G_{k+1} \mathbf{P}_{k|k} G_{k+1}^T + \Pi_x^T R_{k+1} \Pi$

Algorithm 3 shows the data association and updating steps. The first part of **Algorithm 3** (lines 7-22) handles the data association by generating the predicted observations $\hat{\mathbf{z}}$ and comparing them with the actual observation to establish the correct correspondence (line 18). In line 8, it is assumed that each actual observation corresponds to a new beacon, however, the beacon will not be added until the observation passes the validation gate threshold (line 17). Line 9 starts a cycle that generates the predicted observations for the existing beacons as well as for the new one generated in line 8. The innovation covariance matrix Ψ is then computed in the line 14 and, subsequently, on line 15 the squared Manalanobis distance is determined. On line 13, Γ is $\frac{\partial h(\hat{\boldsymbol{\xi}}_{k+1|k}, \hat{\mathbf{m}}_p)}{\partial \mathbf{x}}$ and H^p is Γ after being mapped to higher dimension by $F_{x,p}$. The same operation is applied to $H^{j(i)}$ in the line 27. First, $\frac{\partial h(\hat{\boldsymbol{\xi}}_{k+1|k}, \hat{\mathbf{m}}_{j(i)})}{\partial \mathbf{x}}$ is computed and then mapped by another matrix to yield $H^{j(i)}$. Updating step starts from line 27 by computing the Kalman gain. The estimate and the covariance matrix of the system is updated on lines 28 and 29 according to EKF algorithm. Lines 30 and 31 compute the new belief of the robot which completes EKF-SLAM algorithm.

Algorithm 3 Data Association and Updating Steps of EKF-SLAM

7: **for** all actual observations $\mathbf{z}_{k+1}^i$ **do**

8: $$\begin{bmatrix} \hat{\mathbf{m}}_{N+1,x} \\ \hat{\mathbf{m}}_{N+1,y} \\ \hat{\mathbf{m}}_{N+1,z} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\xi}}_{x,(k+1|k)} \\ \hat{\boldsymbol{\xi}}_{y,(k+1|k)} \\ \hat{\boldsymbol{\xi}}_{z,(k+1|k)} \end{bmatrix} + r^i \begin{bmatrix} \cos(b^i_\theta)\sin(b^i_\phi) \\ \sin(b^i_\theta)\sin(b^i_\phi) \\ \cos(b^i_\phi) \end{bmatrix}$$

9: **for** p = 1 to $(N_k) + 1$ **do**

10: $$\delta_p = \begin{bmatrix} \delta_{p,x} \\ \delta_{p,y} \\ \delta_{p,z} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{m}}_{p,x} - \hat{\boldsymbol{\xi}}_{x,(k+1|k)} \\ \hat{\mathbf{m}}_{p,y} - \hat{\boldsymbol{\xi}}_{y,(k+1|k)} \\ \hat{\mathbf{m}}_{p,z} - \hat{\boldsymbol{\xi}}_{z,(k+1|k)} \end{bmatrix}$$

11: $$q_p = \sqrt{\delta_p^T \delta_p}$$

12: $$\hat{z}^p_{k+1} = \begin{bmatrix} q_p \\ \tan^{-1}\left(\frac{\hat{\mathbf{m}}_{p,y} - \hat{\boldsymbol{\xi}}_{y,(k+1|k)}}{\hat{\mathbf{m}}_{p,x} - \hat{\boldsymbol{\xi}}_{x,(k+1|k)}}\right) \\ \cos^{-1}\left(\frac{\hat{\mathbf{m}}_{p,z} - \hat{\boldsymbol{\xi}}_{z,(k+1|k)}}{q_p}\right) \end{bmatrix}$$

13: $$H^p_{k+1} = \Gamma^p_{k+1} F_{x,p}$$

14: $$\Psi_p = H^p_{k+1} \mathbf{P}_{k+1|k} [H^p_{k+1}]^T + Q_{k+1}$$

15: $$D^2_p = (z^i_{k+1} - \hat{z}^p_{k+1})^T \Psi^{-1}_p (z^i_{k+1} - \hat{z}^p_{k+1})$$

16: **end for**

17: $$D^2_{(N_k)+1} = \chi^2_{\tau,\alpha}$$

18: $$j(i) = \underset{p}{\operatorname{argmin}}\ D^2_p$$

19: **if** $j(i) > N_k$ **then** ▷ augment the system

20: $$N_{k+1} = j(i)$$

21: **Add** $\hat{\mathbf{m}}_{N_{k+1}}$ **To** $\hat{\mathbf{x}}_{k+1|k}$

22: **Add** $P_{\hat{\mathbf{m}}_{N_{k+1}} \hat{\mathbf{m}}_{N_{k+1}}}$ **And** P_Λ **To** $\mathbf{P}_{k+1|k}$

23: **else** ▷ don't augment the system

24: $$N_{k+1} = N_k$$

25: **end if**

26: $$K^i_{k+1} = \mathbf{P}_{k+1|k} [H^{j(i)}_{k+1}]^T \Psi^{-1}_{j(i)}$$

27: $$\hat{\mathbf{x}}_{k+1|k} = \hat{\mathbf{x}}_{k+1|k} + K^i_{k+1} (z^i_{k+1} - \hat{z}^{j(i)}_{k+1})$$

28: $\mathbf{P}_{k+1|k} = (I - K^i_{k+1} H^{j(i)}_{k+1})\mathbf{P}_{k+1|k}$

29: **end for**

30: $\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k}$

31: $\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k}$

3.5 Conclusion

In this chapter, the SLAM problem has been addressed from probabilistic point of view, and the Extended Kalman Filter (EKF) SLAM algorithm for quadrotor UAV is developed. In particular, a general overview of the recursive Bayesian framework for constructing of SLAM is given, and different parametric filters are described that approximate the solution provided by the Bayesian filter. The design procedure for EKF-SLAM algorithm for quadrotor is presented in detail. The developed EKF-SLAM algorithm will be utilized in the subsequent chapter for the purpose of generating haptic feedback in a teleoperator system for remote control of a quadrotor UAV.

Chapter 4

Teleoperation of UAVs with SLAM-based Haptic Feedback

4.1 Introduction

In the previous chapters, the kinematics, dynamics and control of a quadrotor UAV (Chapter 2) and basic formulations related to SLAM algorithms (Chapter 3) were presented. This chapter represents the main contribution of this thesis, which is development of a haptic teleoperator system for remote control of a quadrotor UAV, where the haptic feedback is generated using SLAM algorithms. Specifically, an estimate of the state of the environment obtained using SLAM algorithms will be used to approximately reconstruct a model of the remote environment in the virtual environment at the master site. The haptic feedback will be subsequently generated based on location of the obstacles in the virtual environment using potential force field methods. As a result, the human operator will haptically feel a repulsive force as the quadrotor approaches an obstacle, which increases the situational awareness for the human operator and results in improved performance and safety of UAV teleoperation.

The structure of this Chapter is as follows. Section 4.2 describes the notion of predictive displays. Section 4.3 introduces the SLAM-based haptic feedback approach. In Section 4.4,

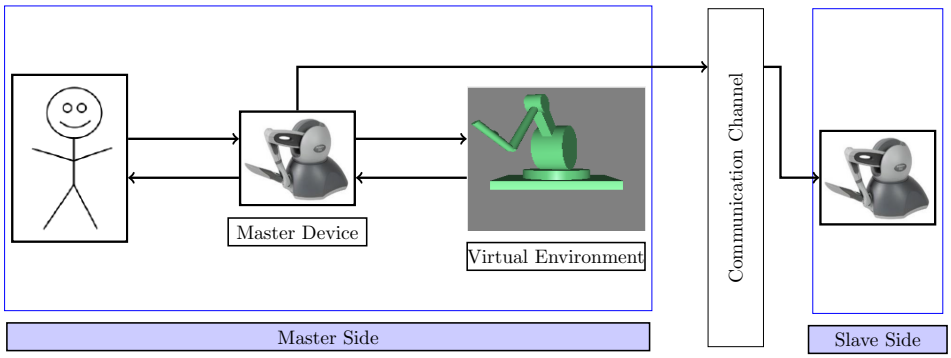


Figure 4.1: Simple predictive display model for bilateral teleoperation systems.

mathematical models for artificial potential fields (APFs) are discussed, and some experimental results are presented that demonstrate the interaction process between the human operator and the APF. The control structure of the teleoperator system with SLAM-based haptic feedback is discussed in Section 4.5. In Section 4.6, two algorithms for SLAM-based haptic feedback are developed, and the corresponding semi-experimental results are presented and discussed. Conclusions are given in Section 4.7.

4.2 Predictive Displays

In teleoperation applications, the existence of communication constraints such as time delays in the communication channel typically results in a number of problems. As mentioned in Chapter 1, time delay may destabilize haptic teleoperation system by generating energy in the communication channel. When using visual feedback, time delay is shown to substantially increase the task completion time [87]; essentially, in this case the human operator must adopt the "move-and-wait" strategy before carrying out the next set of commands. Also, long time delays increase the human operator's workload [88]. One of the most popular approaches to cope with time delay in teleoperation systems is based on the use of **predictive displays**. The idea behind the predictive display is to build a virtual model of the remote environment at the master site, and to provide the human operator with feedback from the local virtual model rather than

with the actual data from the remote site. The predictive displays were used extensively for applications that involve long time delays, such as space and undersea [89, 90, 88]. The approach based on predictive displays was shown to reduce the mental workload for the human operator and to improve the task performance in the presence of large time delays; it also removes the necessity of the "move-and-wait" strategy. The paper [91] is apparently the first work where the predictive display approach was utilized as a means to provide the human operator with the force feedback in the presence of time delay. The general structure of a teleoperator system with haptic predictive display is shown in Figure 4.1. Here, the information about motion of the master device is sent over a delayed communication channel to the remote site, where it is used as a reference trajectory for the slave device. The same reference trajectory, however, is applied to the virtual slave device without delay. The interaction forces between the virtual slave and objects in the virtual environment are calculated and subsequently applied to the master device; therefore, the delay in the haptic channel is eliminated. The central issue associated with the predictor displays approach is that building the virtual model requires detailed knowledge of the remote environment. The approach is therefore not directly applicable in the situations where the remote environment is not precisely known.

4.3 SLAM-Based Haptic Feedback

In this section, the basic idea of SLAM-based haptic feedback is described. As mentioned in the previous section, the predictive displays rely on the virtual model of a remote environment to provide the human operator with different types of feedback. The problem with the predictive displays approach, however, is that building virtual models requires precise knowledge of the remote environment. When the robot navigates unknown environments, the predictive display approach can not be directly utilized due to the fact that the virtual model of a remote environment is not available a priori. In this thesis, we propose the idea of using SLAM algorithms as a means for building the virtual model of an unknown remote environment in real

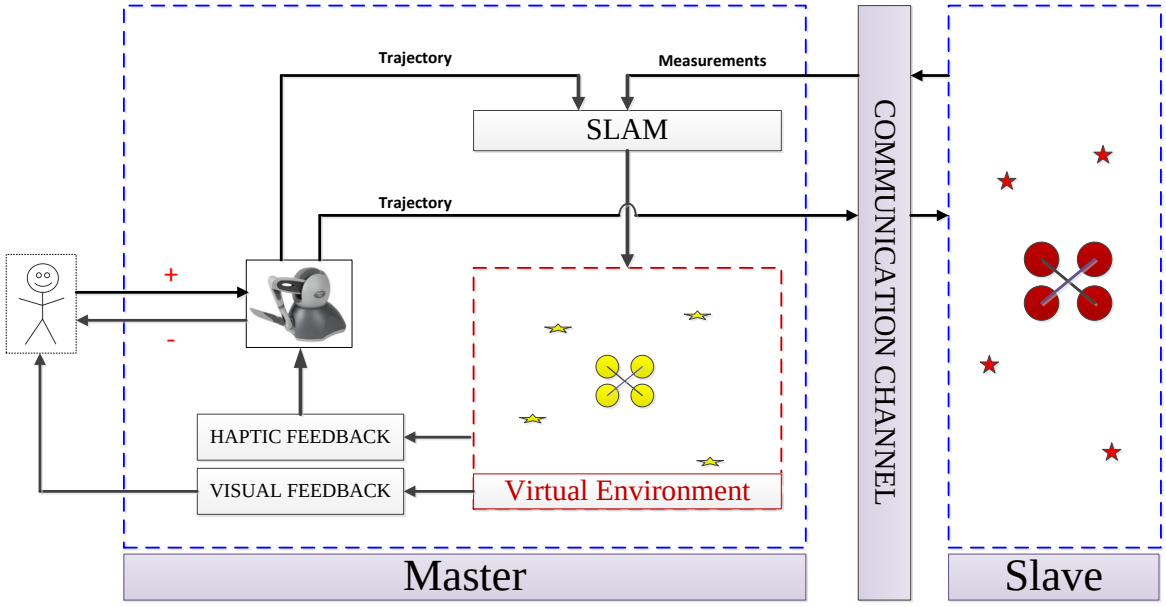


Figure 4.2: Structure of the teleoperator system with SLAM-based haptic feedback

time, and use this model in a teleoperator system to provide haptic feedback to the human operator. Figure 4.2 shows the general structure of a teleoperator system with SLAM-based haptic feedback. In this system, the human operator controls the haptic device; the position of the end-effector of the haptic device determines the desired trajectory for the remote UAV. The UAV executes the desired trajectory while simultaneously scanning the environment for obstacles and measuring distance/direction to them; in Figure 4.2, the obstacles are denoted by stars. These measurements are subsequently transmitted to the master site, where they are used as inputs to the SLAM algorithm. The SLAM algorithm determines the location of the obstacles and updates the virtual environment with a map of the obstacles as well as the position of the UAV with respect to these obstacles. The haptic feedback is generated by building an artificial potential force field around the obstacles in the virtual environment. Figure 4.3 gives a simple 2D picture which illustrates the method for generating the haptic feedback. In this Figure, the blue star and the blue triangle illustrate the actual positions of an obstacle and

the UAV, respectively; the yellow star and the yellow triangle denote estimates of the positions of the obstacle and the UAV, respectively, obtained through running SLAM algorithm. The dashed green line represents the border of the artificial potential field built around the obstacle; the human operator receives haptic feedback as the platform penetrates the artificial potential field. The red line represents the estimated distance of penetration of the UAV into the artificial potential field; the potential field subsequently generates repulsive force which is directed away from the obstacle, while the magnitude of this force is a function of the distance of penetration. As a result, the human operator feels repulsive forces as the UAV approaches the obstacle.

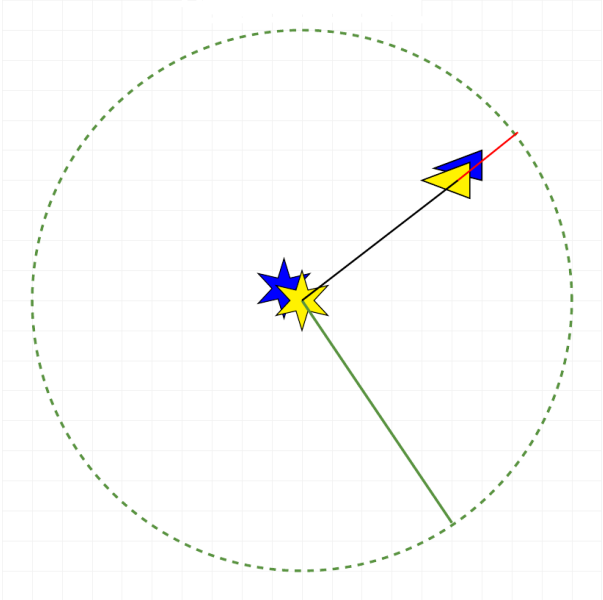


Figure 4.3: 2D illustration of the method for generating the SLAM-based haptic feedback.

4.4 The Artificial Potential Field

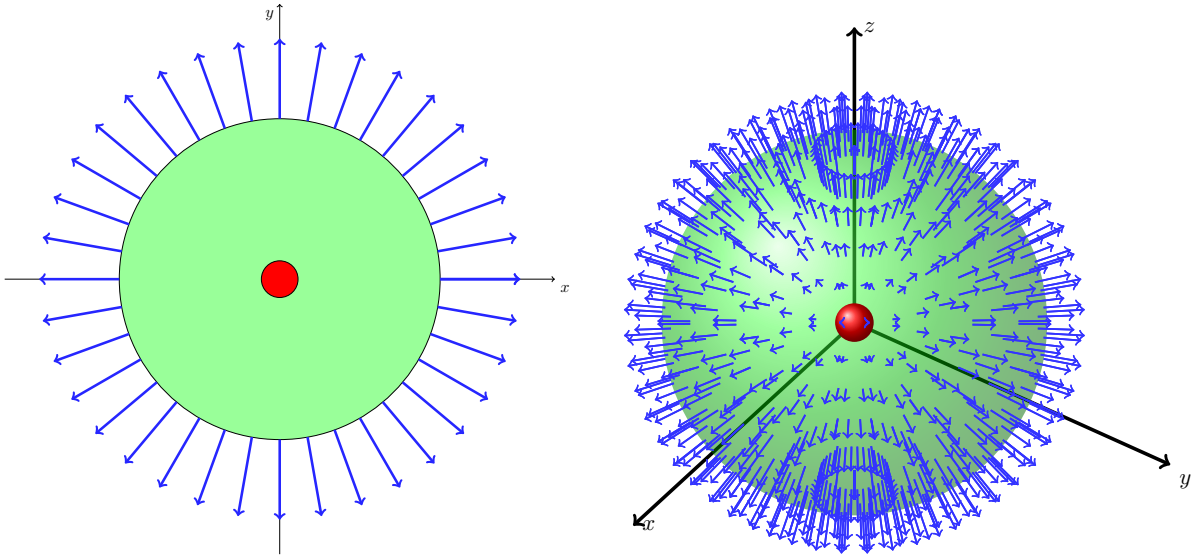
As explained in the previous section, in the proposed teleoperator structure the haptic feedback is generated using an artificial potential field; the latter is built around estimated positions of the obstacles in the virtual environment on the master side. In this section, the methods for building artificial potential fields are addressed in some detail, and the corresponding experimental

results are presented. The primary purpose of the artificial potential field (APF) approach is to transform or map the surrounding environment of a physical object into virtual forces [92]. These virtual forces can be utilized for various purposes, such as to redirect a mobile robot to avoid any potential collision [93], or to alert the human operator if an UAV is approaching a physical object [92]. The idea of an artificial potential field was originally proposed in [94] to deal with collision avoidance problem in robotics. The APF approach allowed for substantial simplification and practical real-time implementation of the obstacle avoidance algorithms, while effectively reducing the amount of higher level off-line planning required. The original paper [94] proposes several different models for APFs, including the one that relies on the shortest distance to an obstacle, as well as models that take the geometric shape of an obstacle into account. In [95], a generalized potential field (GPF) model is proposed, where the force depends not only on the distance to the obstacle but also on the direction of the relative velocity. Specifically, if the robot is moving away from the obstacle then the GPF force is set to zero; otherwise, the force is set to be inversely proportional to the difference between the so called maximum and minimum avoidance times.

In [92], the APF approach was applied to haptic teleoperation of UAVs. In contrast with the above described works where APF forces were injected into the control input to redirect the robot in order to avoid collisions, in [92] the APF forces were used to generate haptic feedback to the human operator. In addition, work [92] introduces a number of modification to the APF method, which make the latter more suitable for haptic teleoperation. The modifications are summarized in two new models for APFs, called the basic risk field (BRF) and the parametric risk field (PRF). These new modifications of the APF models take into account the hardware limitations typical for haptic devices, as well as result in decreasing the human operator workload.

4.4.1 Models for Artificial Potential Field

In this section, models for APFs used in this thesis are described. The APF is built around the obstacles and applies repulsive forces to the haptic device whenever its avatar penetrates the APF. Figures 4.4a and 4.4b show the direction of the repulsive forces; these forces always point outward from the center of an APF which coincides with an obstacle (represented by a red dot). For simplicity of implementation, it is assumed that the APF possesses a uniform geometric shape (*i.e.*, a circle in 2D case, and a sphere in 3D case) regardless of the actual shape of the object. We follow the approach of [94] in the sense that, in the algorithms utilized in our



(a) 2D uniform artificial potential field. (b) 3D uniform artificial potential field.

Figure 4.4: Direction of APF force vectors in the vicinity of obstacles.

work, the APF forces are constantly applied to the end-effector of the haptic device whenever the avatar of the device is inside the APF. Even though this may result in an increase of the human operator workload as claimed in [92], we believe this approach has certain advantages that make it beneficial in our case. Specifically, the approach makes the human operator constantly alerted while the avatar of the device is inside the potential field, thus increasing the human operator's awareness and trust in safety of the operation. Moreover, since the environment in our case is feature-based, and interaction with the obstacles can be assumed relatively

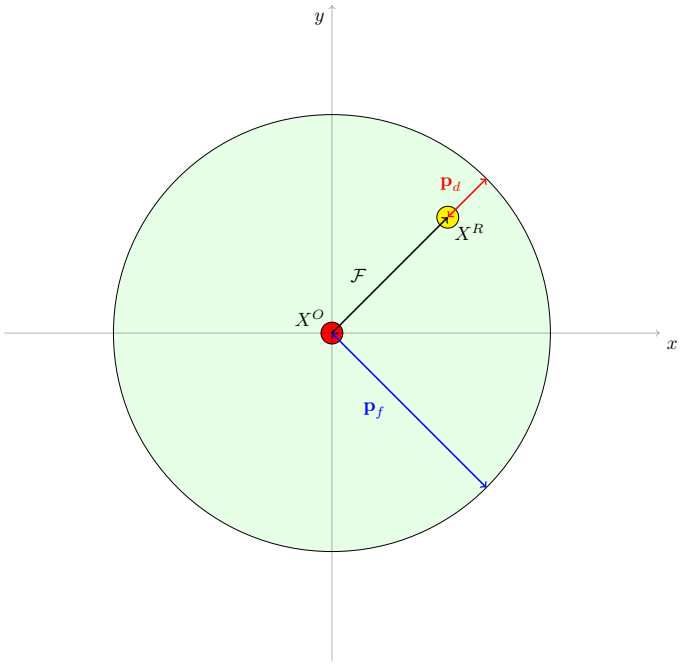


Figure 4.5: Algorithm 4: APF based on the penetration depth only

infrequent in UAV teleoperation, the corresponding increase of the human operator workload is expected to be insignificant. In addition, stiffness of APF can be adjusted to decrease the human operator's workload to an acceptable level, if necessary.

Two algorithms for APFs are addressed in this work. The first algorithm is described below as Algorithm 4. This algorithm calculates the repulsive force based on the depth of penetration only. Specifically, the APF force is directed outward from the position of the obstacle, and its magnitude is proportional to the depth of penetration $\mathbf{p}_d$ multiplied by the APF stiffness $\mathbf{K}_p$. The algorithm is illustrated in Figure 4.5, where the green, the red, and the yellow circles represent the APF, the obstacle, and the end-effector of the haptic device, respectively, while X^R and X^O are the end effector's position and the obstacle's position, respectively.

Algorithm 4 : APF based on the penetration depth

get X^R and X^O
$X^d = \|X^R - X^O\|$
if $X^d < \mathbf{p}_f$ **then** ▷ where $\mathbf{p}_f$ is APF radius
    $\mathbf{p}_d = \mathbf{p}_f - X^d$ ▷ compute penetration distance
    $\mathbf{F}_d = \frac{X^R - X^O}{\|X^R - X^O\|}$ ▷ determine force direction (i.e. unit vector)
    $\mathcal{F} = \mathbf{K}_p \cdot \mathbf{p}_d \cdot \mathbf{F}_d$.
end if

Algorithm 4 does not utilize the robot's relative velocity vector. The use of relative velocity can be beneficial in that it can be used to distinguish between the cases where the end-effector approaching the obstacle and where it is moving away from the obstacle. In the second algorithm (Algorithm 5), the APF force depends on the direction of the movement; specifically, a damping term is added to the repulsive force if the platform is approaching the obstacle. Figure 4.6 illustrates Algorithm 5. Here, $\dot{X}^R$ and $\tilde{X}$ represent the velocity of the end effector and the difference between the obstacle's position and the end effector's position, respectively. If the angle ϕ between $\dot{X}^R$ and $\tilde{X}$ is less than 90°, this implies that the end effector is approaching the obstacle, and therefore the damping term is added to the repulsive force according to Algorithm 5; otherwise, the damping term is set to zero. Both Algorithm 4 and Algorithm 5 allow for a straightforward extension to the 3D case.

Algorithm 5 : APF with velocity-dependent term.

get $\dot{X}^R$, X^R and X^O
$X^d = \|X^R - X^O\|$
if $X^d < \mathbf{p}_f$ **then**
    $\mathbf{p}_d = \mathbf{p}_f - X^d$,
    $\mathbf{F}_d = \frac{X^R - X^O}{\|X^R - X^O\|}$,
    $\tilde{X} = X^O - X^R$,
    $\phi = \cos^{-1}(\frac{(\dot{X}^R)^T \tilde{X}}{\|\dot{X}^R\|\|\tilde{X}\|})$
    if $\phi < 90°$ **then**
        $\mathcal{F} = \underbrace{\mathbf{K}_p \cdot \mathbf{p}_d \cdot \mathbf{F}_d}_{stiffness} + \underbrace{\mathbf{D}_p \cdot \|\dot{X}^R\| \cdot \cos\phi \cdot \mathbf{F}_d}_{damping}$ ▷ where $\mathbf{D}_p > 0$ is damping constant.
    else
        $\mathcal{F} = \mathbf{K}_p \cdot \mathbf{p}_d \cdot \mathbf{F}_d$
    end if
end if

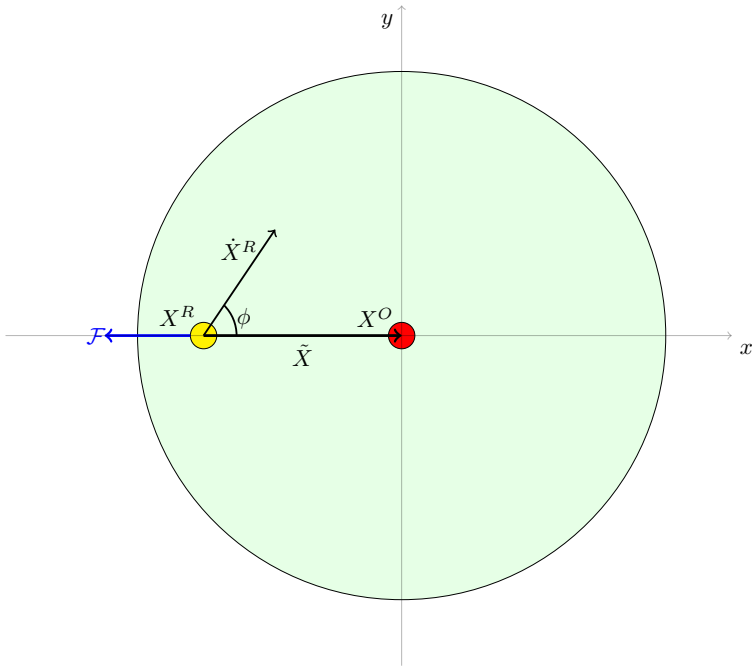


Figure 4.6: Algorithm 5: APF with velocity-dependent term.

4.4.2 Experimental results for APFs

In this section, some experimental results for APF are presented. The APF has been implemented according to Algorithm 5 using Phantom Omni as a haptic device (see Appendix B for more information about this device) and OpenGL as a Graphics library. The aim of these experiments is to demonstrate the interaction process between the human operator and the APF under different values of stiffness and damping parameters; in particular, one of the goals is to determine the values of these parameters that do not deteriorate performance of the haptic interaction. Since the quadrotor is modelled as a point in order to generate the haptic feedback as we will see subsequently, in this section, we carry out the tests considering solely the position of the end-effector of the haptic device, therefore, in these experiments, the human operator moves the end-effector of the haptic device so that the avatar penetrates the APF of a virtual object. Figure 4.7 shows the Graphical User Interface (GUI) used in these experiments for the 2D scenario. The green circle represents APF that surrounds a virtual object, the latter shown as the red point. The yellow point is the avatar of the end-effector of the Phantom Omni haptic device. The menu on the right side of GUI provides real-time data related to the experiment

Figure 4.8	stiffness $\mathbf{K}_p$ (N/mm)	damping $\mathbf{D}_p$ (N· ms/mm)	APF radius $\mathbf{p}_f$ (mm)
Figure (a)	0.25	0.01	40
Figure (b)	0.1	0.1	40
Figure (c)	0.1	0.6	40
Figure (d)	0.6	0.1	40

Table 4.1: The numerical parameters of stiffness and damping terms for 2D APF.

such as position of the end-effector, velocity of the end-effector, joint angles, angular velocities, and Gimbal angles. The update rate of the haptic rendering loop is set to 1KHz. Table 4.1 summarizes the values of the parameters of APF experiments in 2D case; the corresponding experimental results that show the stiffness and damping terms of the reflected force applied to the human operator's hand are given in Figure 4.8. In Figure 4.9, the GUI for 3D case is shown. Table 4.2 summarizes the values of the parameters for 3D APF experiment, while the experimental results are shown in Figure 4.10.

One of the observations that can be made from results of the previous experiments is that increasing the damping constant affects stability of the haptic device. Specifically, the end-effector starts oscillating when the human operator penetrates the APF; as a result, the human operator needs to make additional effort to stabilize the device. This leads to increased workload for the human operator. To overcome this problem, the constant of the damping term is decreased to satisfy $\mathbf{D}_p \leq 0.1$, at which the oscillations are not generated and the performance of the haptic interaction is not affected. On the other hand, increasing the stiffness constant does not have similar negative effect as in the case of damping constant; however, increasing the stiffness constant results in APF generating high repulsive forces; in particular, this may result in a sudden generation of high forces when the avatar of the device interacts with the APF. To avoid such a sudden generation of high forces, the stiffness constant is chosen to satisfy $\mathbf{K}_p \leq 0.2$ in the subsequent studies.

Figure 4.10	stiffness $\mathbf{K}_p$ (N/mm)	damping $\mathbf{D}_p$ (N· ms/mm)	APF radius $\mathbf{p}_f$ (mm)
Figure (a)	0.25	0.1	40
Figure (b)	0.7	0.01	40
Figure (c)	0.1	0.3	40
Figure (d)	0.4	0.01	40

Table 4.2: The numerical parameters of stiffness and damping terms for 3D APF.

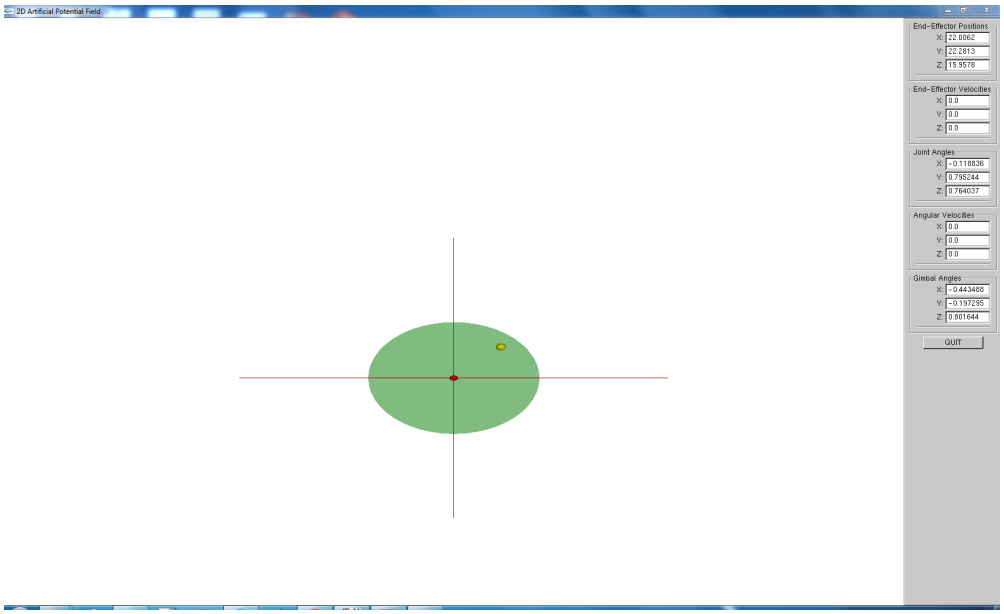
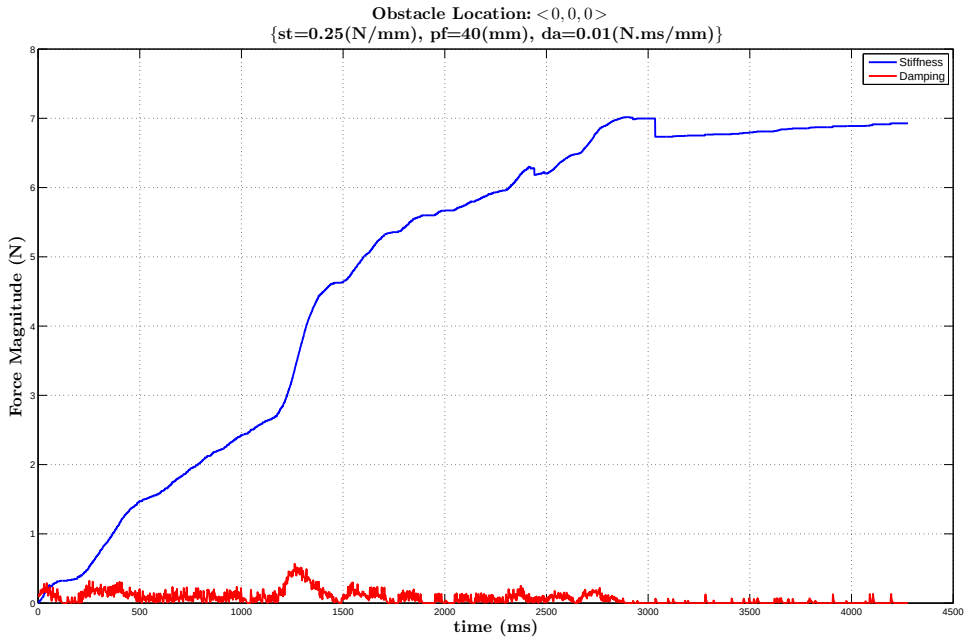
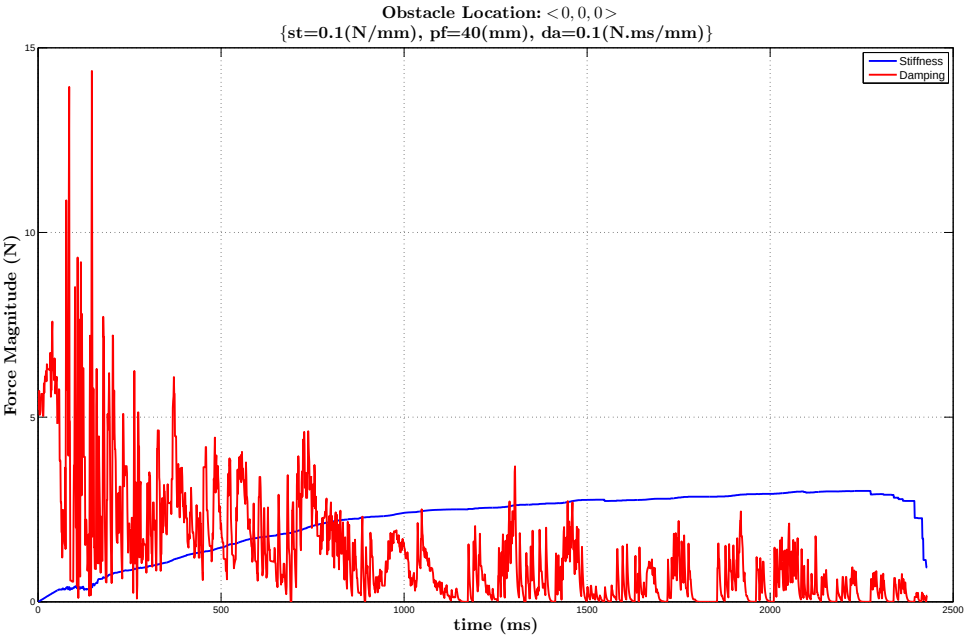


Figure 4.7: GUI used in the experiments with 2D APF.

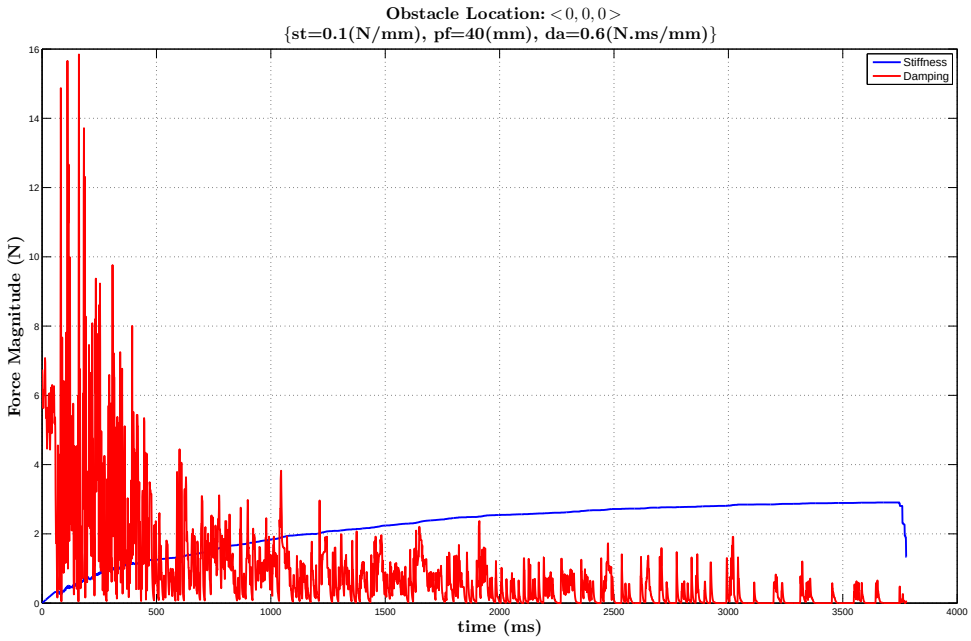


(a) $\mathbf{K}_p$ = 0.25 (N/mm), $\mathbf{D}_p$ = 0.1 N· ms/mm

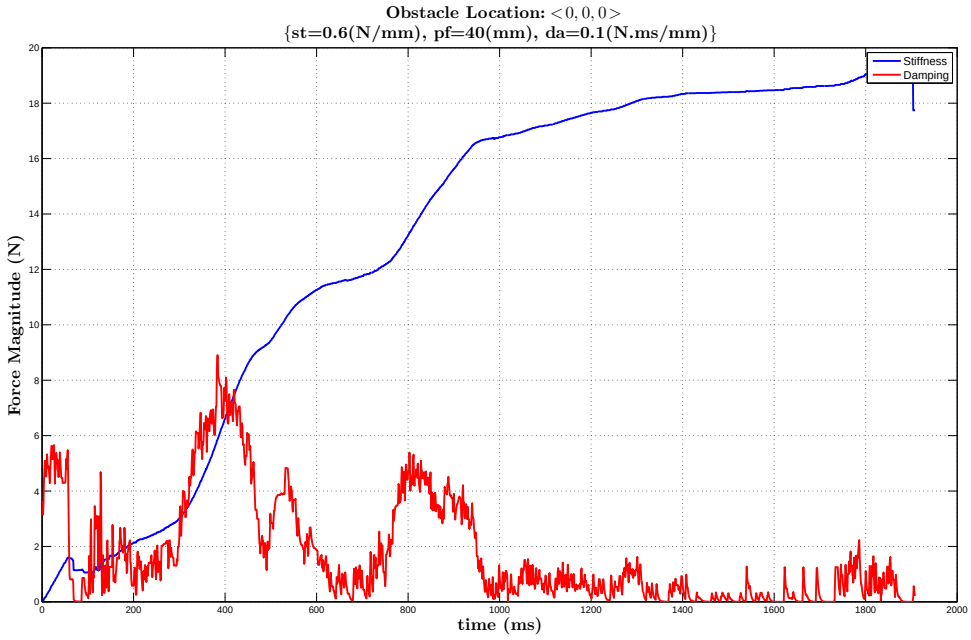


(b) $\mathbf{K}_p$ = 0.1 (N/mm), $\mathbf{D}_p$ = 0.1 N· ms/mm

Figure 4.8: Response of 2D APF: the stiffness and the damping components of the reflected force.



(c) (cont.) $\mathbf{K}_p$ = 0.1 (N/mm), $\mathbf{D}_p$ = 0.6 N· ms/mm



(d) (cont.) $\mathbf{K}_p$ = 0.6 (N/mm), $\mathbf{D}_p$ = 0.1 N· ms/mm

Figure 4.8 (cont.): Response of 2D APF: the stiffness and the damping components of the reflected force.

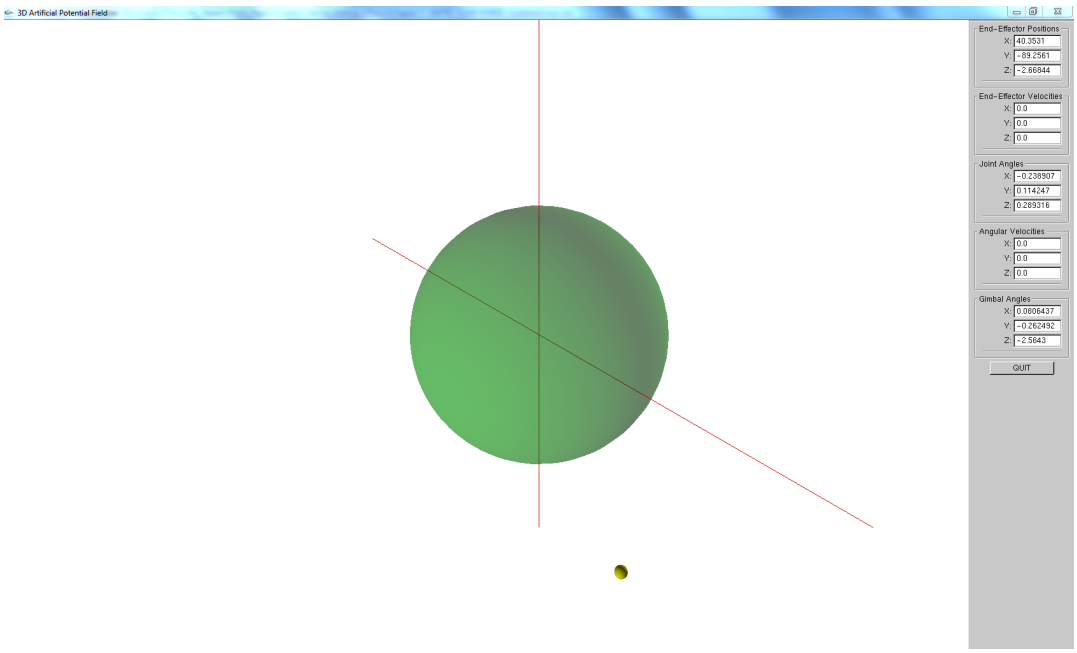
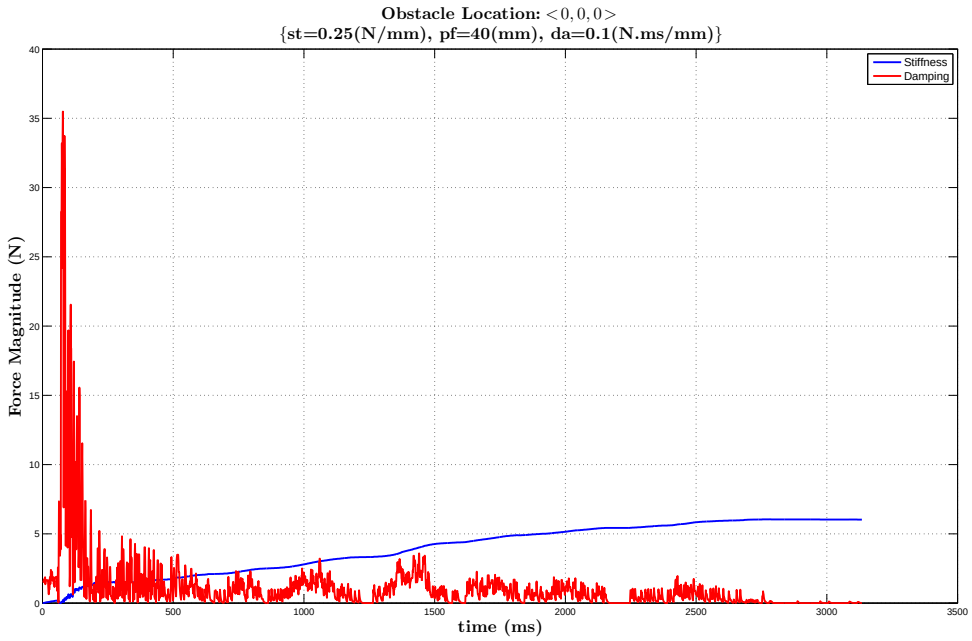
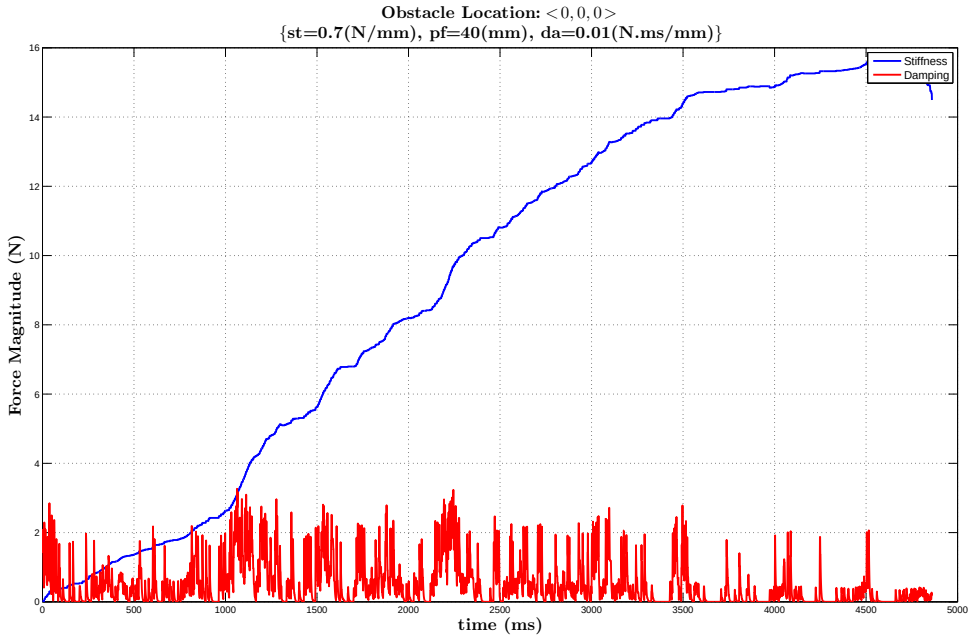


Figure 4.9: GUI used in the experiments with 3D APF.

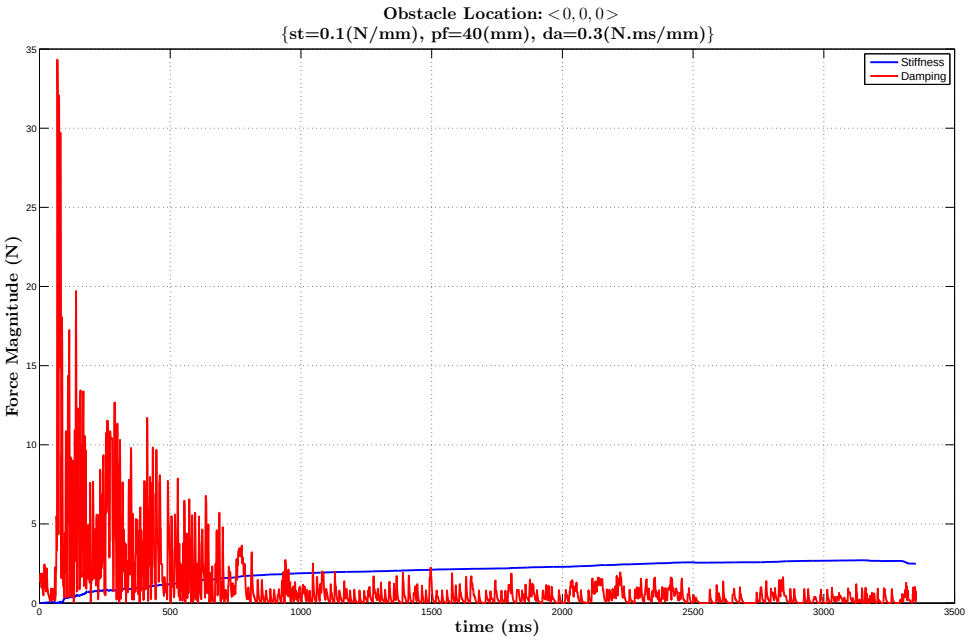


(a) $\mathbf{K}_p$ = 0.25 (N/mm), $\mathbf{D}_p$ = 0.1 N· ms/mm

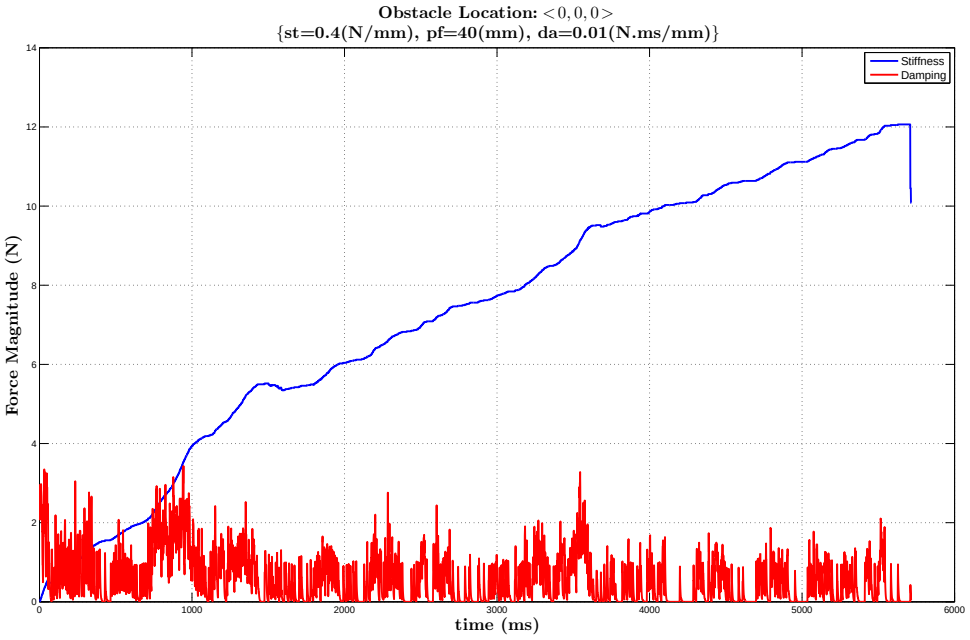


(b) $\mathbf{K}_p$ = 0.7 (N/mm), $\mathbf{D}_p$ = 0.01 N· ms/mm

Figure 4.10: Response of 3D APF: the stiffness and the damping components of the reflected force.



(c) (cont.) $\mathbf{K}_p$ = 0.1 (N/mm), $\mathbf{D}_p$ = 0.3 N· ms/mm



(d) (cont.) $\mathbf{K}_p$ = 0.4 (N/mm), $\mathbf{D}_p$ = 0.01 N· ms/mm

Figure 4.10 (cont.): Response of 3D APF: the stiffness and the damping components of the reflected force.

4.5 Control Structure of a Teleoperator System with SLAM-based Haptic Feedback

In this Section, the overall control structure of a teleoperator system with SLAM-based haptic feedback is described. The control structure is schematically shown in Figure 4.11. Two types

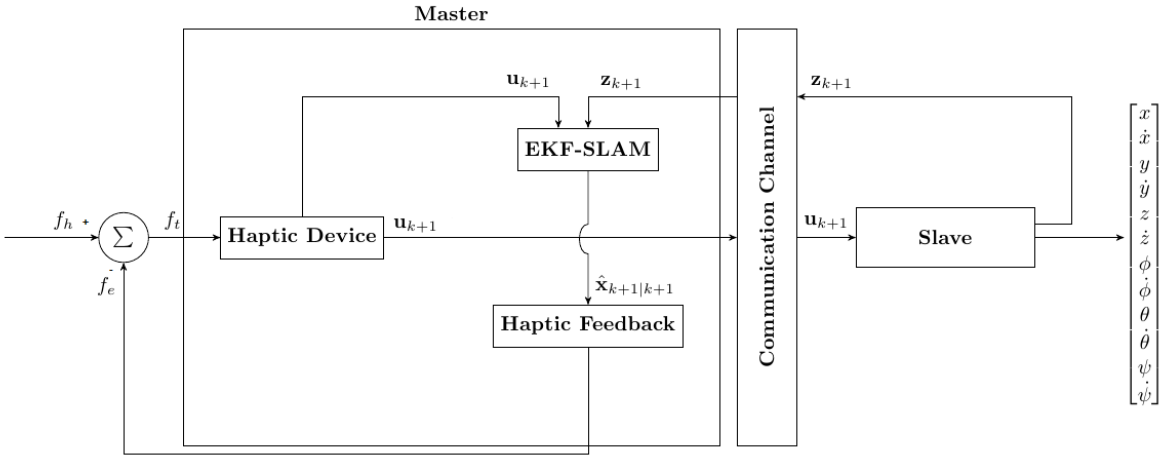


Figure 4.11: Control structure of a teleoperator system with SLAM-based haptic feedback.

of control architectures are used in this work, which are position-to-position control architecture and position-to-velocity control architecture. In position-to-position control architecture, the position of the end-effector of the haptic device plays a role of the desired position for the center of the quadrotor's mass. This type of control architecture is beneficial in applications that require the quadrotor to hover close to obstacles in order to execute tasks that require precise position control, such as lifting weights. On the other hand, if the task requires the quadrotor to fly over large distances such as in the case of exploration missions, position-to-position control may not be applicable due to the fact that the workspace of the haptic device is limited. An alternative approach that can be used in this case to overcome the limitation of the haptic device's workspace is to map the position of the end-effector into the desired linear velocity of the center of the quadrotor's mass. This type of control architecture is called position-to-velocity.

The overall structure can be described as follows. At the master side, the human operator controls the end-effector of the haptic device by applying force f_h. The control input $\mathbf{u}_m$ is the position of the end-effector where the subscript m indicates the the Master side,

$$\mathbf{u}_m := [x, y, z] \tag{4.1}$$

At the slave side, the quadrotor receives the desired trajectories denoted as $\mathbf{u}_s$ where the subscript s indicates the slave side. For position-to-position control, $\mathbf{u}_s$ is

$$\mathbf{u}_s = [x_d, y_d, z_d] \tag{4.2}$$

For position-to-velocity strategy, $\mathbf{u}_s$ is

$$\mathbf{u}_s = [\dot{x}_d, \dot{y}_d, \dot{z}_d] \tag{4.3}$$

The quadrotor follows the desired trajectory and navigates an unknown 3D environment of a feature-based type. An example of such an environment is shown in Figure 4.12 where the blue objects are the true locations of the beacons (obstacles) in the global frame. The quadrotor scans the environment through its 3D sensor that is mounted at the center of its mass. For each scan, the measurements $\mathbf{z}$ is generated and sent back to the master side. At the master side, these measurements are used as an input to the EKF-SLAM algorithm; the algorithm is used to construct a virtual environment which replicates the essential features of the actual remote environment. An example of a virtual environment constructed by the EKF-SLAM algorithm that corresponds to the actual environment of Figure 4.12 is shown in Figure 4.13. In this figure, the red dot represents the estimate of the position of the center of the quadrotor's mass. Estimates $\hat{\mathbf{x}}_{k+1|k+1}$ of the positions of obstacles are represented by the yellow dots. Also, the blue dots represent the actual position of the obstacles which are not accessible in practice but shown here for illustrative purposes. One of the obstacles in Figure 4.13 does not have its estimated counterpart; this reflects the fact that some obstacles may not be sensed by the quadrotor's sensor at all times. Once the obstacle falls within the sensor's range of the quadrotor, an

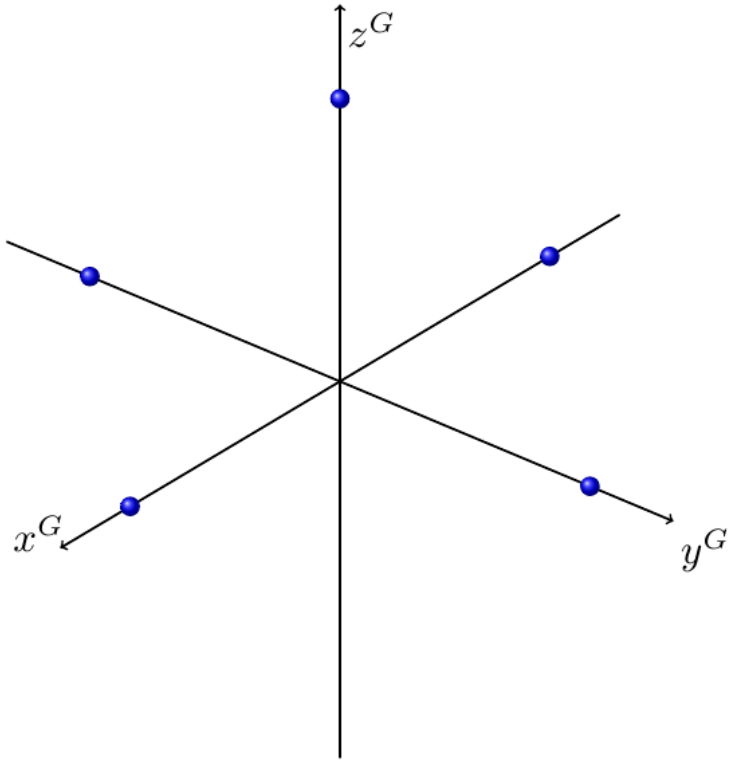


Figure 4.12: An example of a feature-based map of the environment.

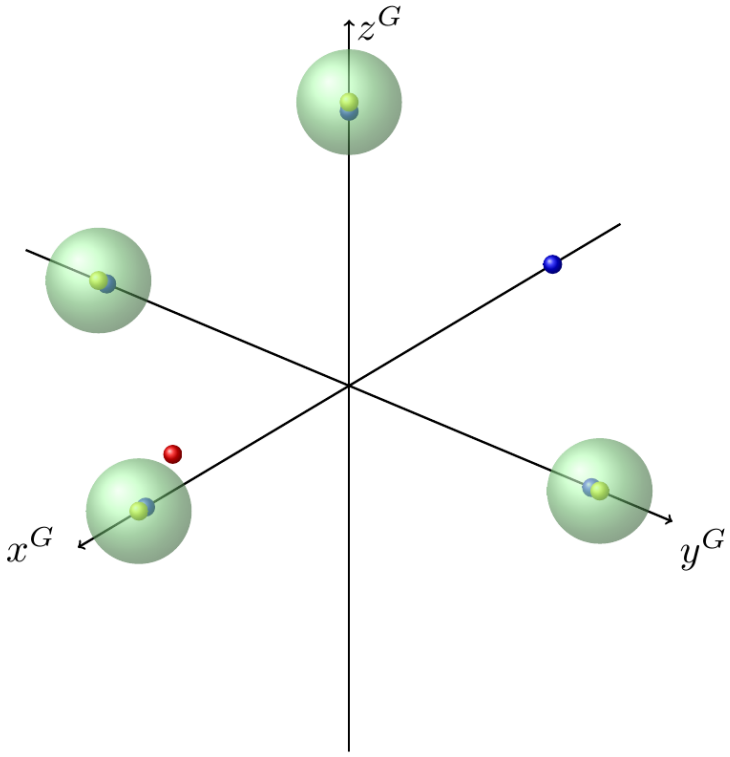


Figure 4.13: A virtual environment constructed by EKF-SLAM algorithm on the master side which corresponds to the map in Figure 4.12.

estimate of its location is added to the state vector of the SLAM, and its visual representation appears in GUI at the master side. Green shaded spheres represent the artificial potential field implemented around each obstacle in the virtual environment. Once the quadrotor penetrates the APF around an obstacle, as shown in Figure 4.13, the APF generates repulsive forces which provides the human operator with haptic feedback indicating proximity of the obstacle. The deeper the end effector of the haptic device penetrates the APF, the higher reflected force is applied to the haptic device to prevent further penetration. Figure 4.14 shows the artificial potential field is being penetrated by the platform where $\mathbf{p}_d$ is the penetration depth, $\mathcal{F}$ is the reflected force and $\mathbf{p}_f$ is the radius of the potential field where F^B is the beacon's local frame.

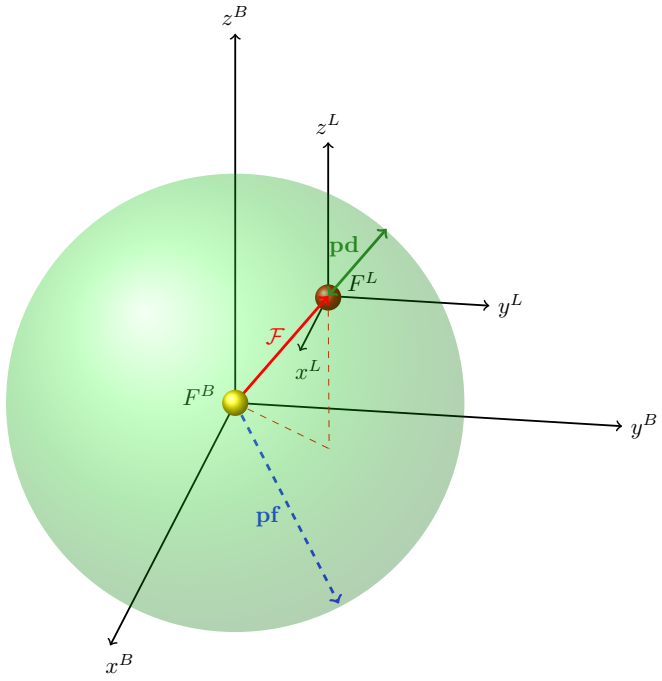


Figure 4.14: 3D potential force field around a beacon penetrated by the quadrotor.

4.6 Algorithms for SLAM-based haptic feedback

In this section, we presents results related to two types of SLAM-based algorithms for generating the artificial potential field around an estimated location of the obstacles. The first type, which is called the basic SLAM-based haptic feedback algorithm, the APF around each obstacle is characterized by the same set of parameters, in particular the size of the APF around each obstacle is equal. This type of APF is simplest in terms of implementation, however, a possible drawback of this approach is that the uncertainty of the beacon location estimates are not taken into account. The second type of algorithms proposed in this work is called the robust SLAM-based haptic feedback algorithm. This algorithms takes into account the uncertainty of the estimates of the obstacle positions which is characterized by covariance estimate of the EKF-SLAM algorithm. In this case, the size of APF around the obstacles depends on the uncertainty in the available estimates, *i.e.*, larger uncertainty result in larger size of the corresponding APF. Below, both these algorithms are theoretically described and experimentally

investigated.

4.6.1 Basic SLAM-based haptic feedback algorithm

Basic SLAM-based haptic feedback algorithm is presented below as **Algorithm** 6. This algorithm uses an estimate $\hat{\mathbf{x}}_{k+1|k+1}$ of the positions of the quadrotor and the beacons (obstacles) provided by Algorithms 2, 3 of Chapter 3. Lines 32-34 of Algorithm 6 compute the Euclidean distance between the estimates of the quadrotor's position $\hat{\boldsymbol{\xi}}_{k+1|k+1}$ and positions of each existing beacon $\hat{\boldsymbol{m}}_{k+1|k+1}$ in the state vector. The index $\lambda_{\mathbf{J}}$ of the beacon which closest to the quadrotor is determined in line 35. If the distance between the quadrotor and $\lambda_{\mathbf{J}}$-th beacon is smaller than the predefined radius of the potential field $\mathbf{p}_f$, this indicates that the platform is penetrating the corresponding APF. The depth of penetration into the APF is computed in line 39. Finally, the magnitude and the direction of the reflected force $\mathcal{F}$ is calculated in lines 37-43. These forces are subsequently applied to the motors of haptic device in order to provide the human operator with haptic feedback. **Algorithm** 6 is illustrated in Figure 4.15.

Algorithm 6 Basic SLAM-based Haptic Feedback Algorithm

32: **for** b = 1 to N_{k+1} **do**
33: $\quad \lambda_b = \sqrt{(\hat{\boldsymbol{\xi}}_x - \hat{\boldsymbol{m}}_{b,x})^2 + (\hat{\boldsymbol{\xi}}_y - \hat{\boldsymbol{m}}_{b,y})^2 + (\hat{\boldsymbol{\xi}}_z - \hat{\boldsymbol{m}}_{b,z})^2}$
34: **end for**
35: $\mathbf{J} = \underset{b}{\operatorname{argmin}}\, \lambda_b$
36: **if** $\lambda_{\mathbf{J}} < \mathbf{p}_f$ **then** ▷ where $\mathbf{p}_f$ is the radius of potential field
37: $\quad \varphi = \tan^{-1}\left(\frac{\hat{\boldsymbol{m}}_{\mathbf{J},y} - \hat{\boldsymbol{\xi}}_y}{\hat{\boldsymbol{m}}_{\mathbf{J},x} - \hat{\boldsymbol{\xi}}_x}\right)$
38: $\quad \beta = \cos^{-1}\left(\frac{\hat{\boldsymbol{m}}_{\mathbf{J},z} - \hat{\boldsymbol{\xi}}_z}{\lambda_{\mathbf{J}}}\right)$
39: $\quad \mathbf{p}_d = \mathbf{p}_f - \lambda_{\mathbf{J}}$ ▷ where $\mathbf{p}_d$ is penetration distance
40: $\quad \mathcal{F} = -\mathbf{K}_p \cdot \mathbf{p}_d$ ▷ where $\mathbf{K}_p$ is the stiffness of APF
41: $\quad \mathcal{F}_x = \mathcal{F}\cos(\varphi)\sin(\beta)$
42: $\quad \mathcal{F}_y = \mathcal{F}\sin(\varphi)\sin(\beta)$
43: $\quad \mathcal{F}_z = \mathcal{F}\cos(\beta)$
44: **end if**

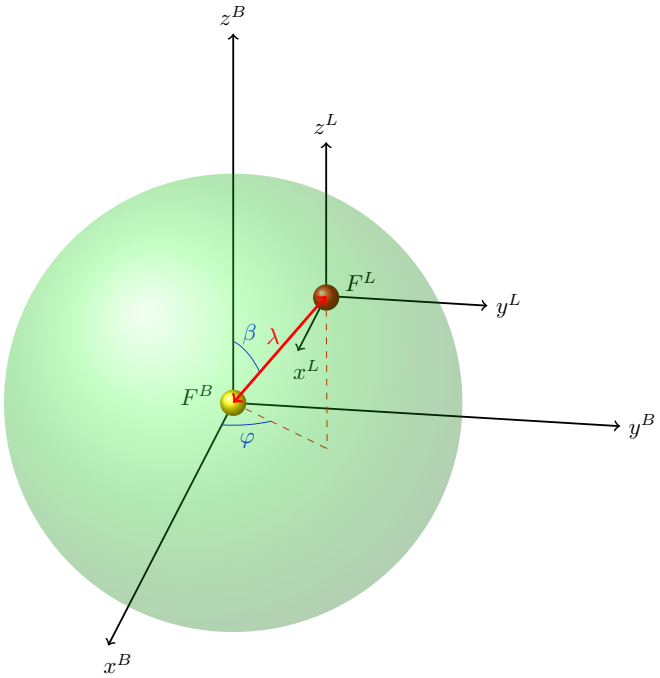


Figure 4.15: Illustration of Algorithm 6.

4.6.2 Semi-experimental results for basic SLAM-based haptic feedback algorithm

In this subsection, results of semi-experimental investigation of the basic SLAM-based haptic feedback algorithm (Algorithm 6) are presented, where the estimates of the obstacle locations are obtained using the EKF-SLAM algorithm (Algorithms 2 and 3). In this experiment, the human operator physically operates the haptic device, however, the quadrotor and the remote environment are simulated in real-time, which is the reason for the use of the term "semi-experimental." The position of the end-effector of the haptic device determines the trajectory that the quadrotor should follow. In this experiment, there are four beacons (denoted **B1**, **B2**, **B3**, **B4**, respectively) simulated at the slave side, whose actual locations are given in Table 4.3. The human operator attempts to approach each beacon as shown in Figure 4.16, and penetrates its APF. When the quadrotor's sensor detects a beacon at the slave side, the estimate of its location is added to the state vector of the SLAM algorithm and visualized in the virtual model

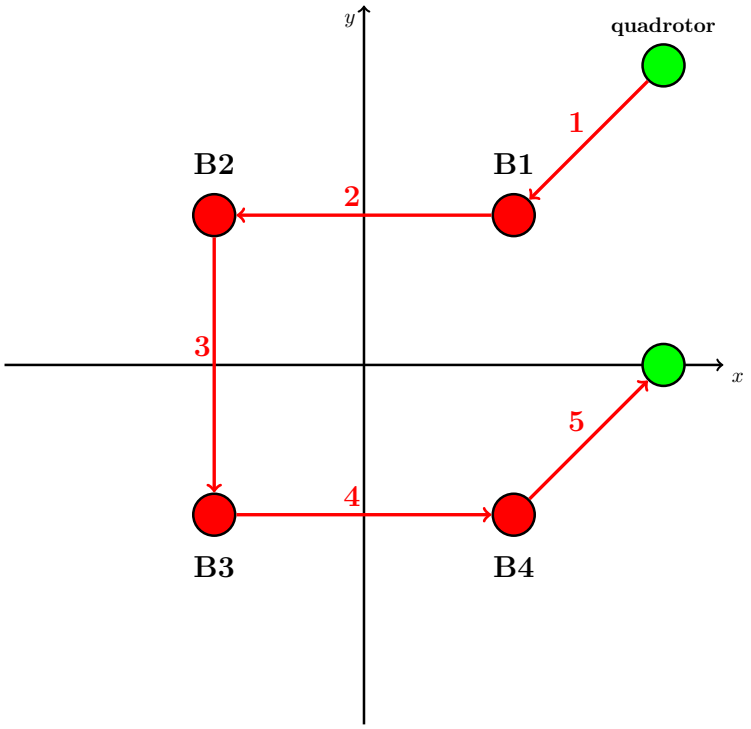


Figure 4.16: The trajectory of the quadrotor on the slave side.

of the remote environment at the master side, as shown in Figure 4.17. In this figure, the blue spheres represent estimates of the beacons' location (denoted $\widehat{\mathbf{B1}}$, $\widehat{\mathbf{B2}}$, $\widehat{\mathbf{B3}}$, and $\widehat{\mathbf{B4}}$, respectively) while the shaded red spheres represent the true location of the beacons. The true locations are shown here for illustrative purposes; however, in reality, they are not visualized in the virtual model. Once the estimate of the quadrotor's location, represented as a yellow sphere in Figure 4.17, penetrates the APF built around the estimated location of a beacon, a repulsive force is generated by the APF algorithms and is applied to the hand of the human operator via the haptic device.

The quadrotor is assumed to be equipped with a range sensor that measures distance r to an obstacle up to 4 m with uncertainty $\pm\sigma_r$ where σ_r is 1 cm. Also, the sensor is assumed to provide a bearing (*i.e.* the azimuthal polar angles) with uncertainty $\pm\sigma_{b_{\phi,\theta}}$ where $\sigma_{b_{\phi,\theta}}$ is 0.1° degree. The motion model of the quadrotor is not subjected to any external noise (*i.e.*, zero process noise is assumed). The GUI used in this experiment is programmed by using

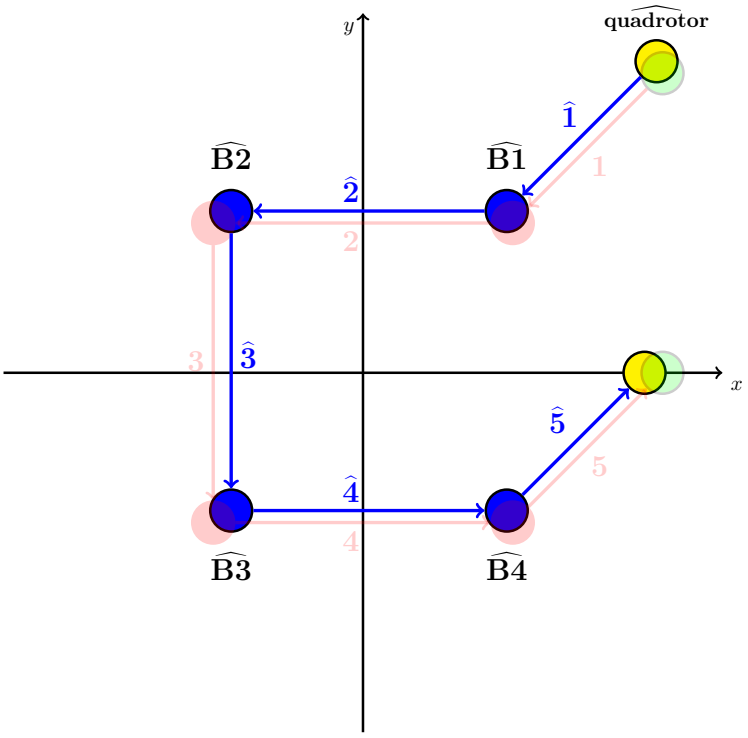


Figure 4.17: Estimates of the beacons' locations at the master side (blue spheres) vs. true locations (shaded red spheres).

C++/OpenGL, and shown in Figure 4.18.

Beacon jth	$\mathbf{m}_{j,x}$	$\mathbf{m}_{j,y}$	$\mathbf{m}_{j,z}$	Unit
1	7	7	6	(m)
2	7	-7	6	(m)
3	-7	7	6	(m)
4	-7	-7	6	(m)

Table 4.3: Basic SLAM-based haptic feedback: True locations of Beacons at the Slave side.

For the experiments with the basic SLAM-based haptic feedback algorithm, the position-to-position control strategy is chosen. Table 4.4 shows the parameters of the workspace of the Phantom Omni device (for more information about the device, see Appendix B). The workspace is too small for the direct (not scaled) position-to-position control. To overcome this spatial limitations of the device workspace, reference trajectory is scaled up so that the quadrotor can cover at least an area of a large room size. In our experiments, the position of the end-effector is scaled up 100 times; therefore, one centimeter increment of the position of

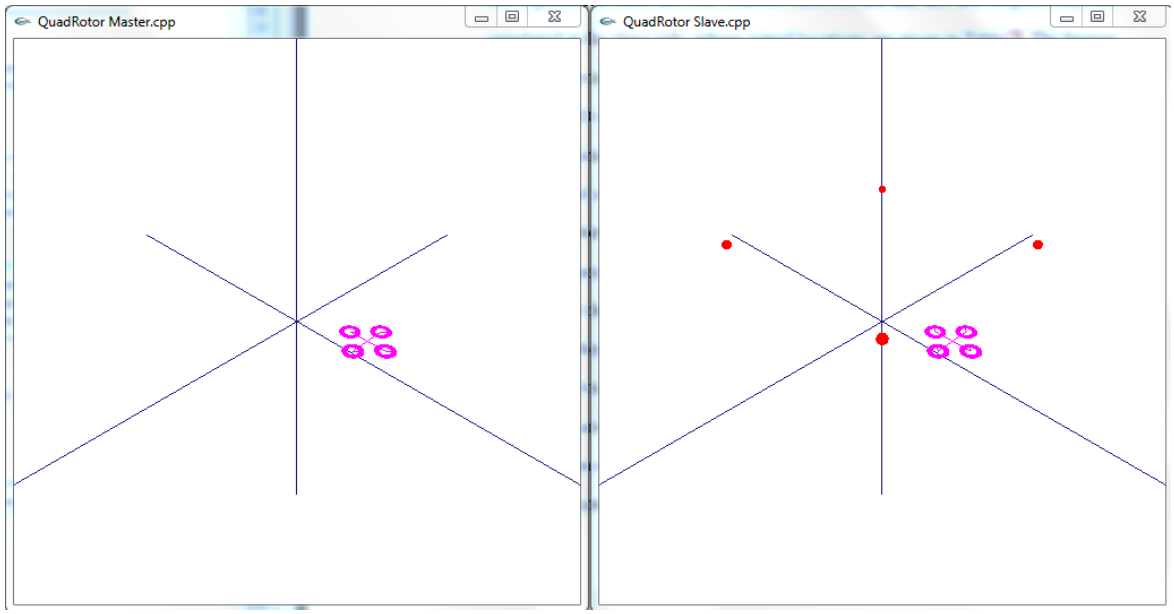


Figure 4.18: Basic SLAM-based haptic feedback: GUI which shows the master (left) and slave (right) sides.

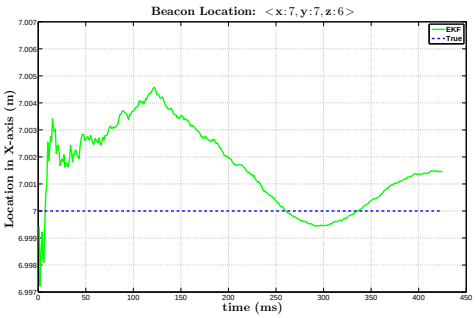
the end-effector corresponds to one meter increment of the quadrotor's reference position.

Position	Minimum	Maximum	Unit
x^h	-210	210	mm
y^h	-110	205	mm
z^h	-85	130	mm

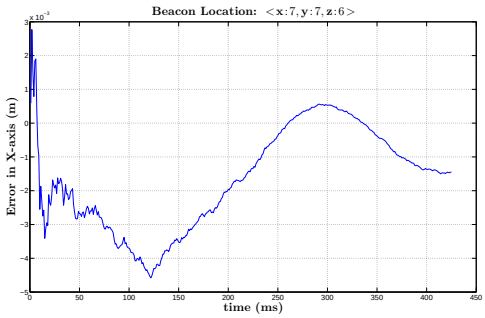
Table 4.4: The workspace of the Phantom Omni.

The APF (scaled) radius $\mathbf{p}_f$ is chosen equal to 3 m, and its stiffness $\mathbf{K}_p$ is 0.15 (N/m). The confidence level for the validation gate is chosen at 95%, which results in threshold $\chi^2_{3,0.05} = 8.0$. The experimental results for basic SLAM-based haptic feedback algorithm are shown in Figures 4.19-4.30. In these figures, estimated and true location of the beacons vs. time, estimation errors vs. time, magnitude of the reflected force vs. time, APF penetration distance vs. time, as well as the x, y, and z-components of the reflected force vs. time are shown for all four beacons that were successively approached by the quadrotor during the experiment.The results of this experiment demonstrate feasibility of the developed basic SLAM-based haptic feedback algorithm. Moreover, they show that no false beacons have been reported. This is crucial not only in SLAM but also in our case in which false beacons will eventually deteriorate

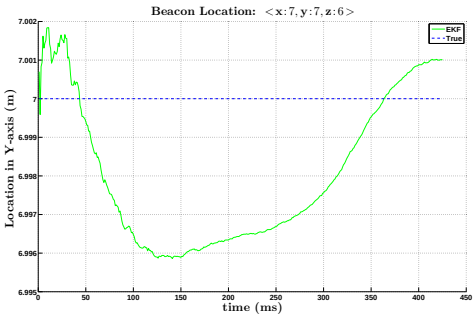
the perception of the human operator about the remote environment. Another conclusion can be made from the aforementioned results is the fact that errors in estimates of beacons' location are rather small and bounded. In other words, the radius of APF is large enough to cover the place where the true location of beacons may fall.



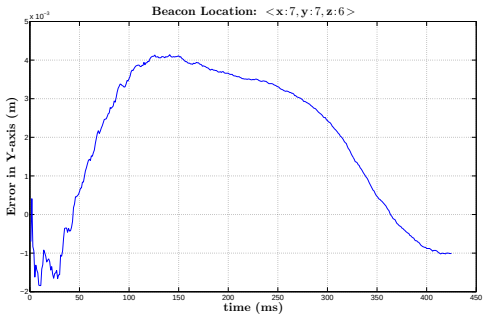
(a) The beacon's estimated vs. true location along the x-axis



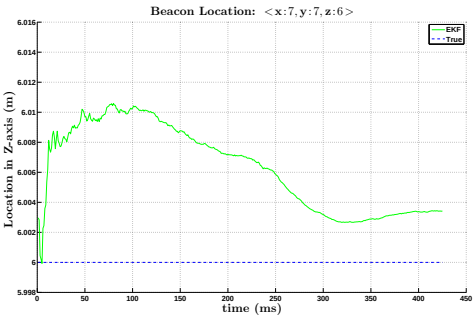
(b) Estimation error (m) along the x-axis



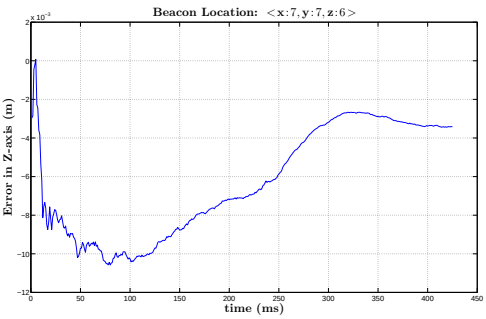
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

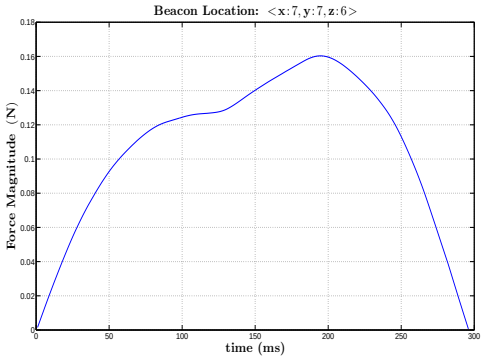


(e) The beacon's estimated vs. true location along the z-axis

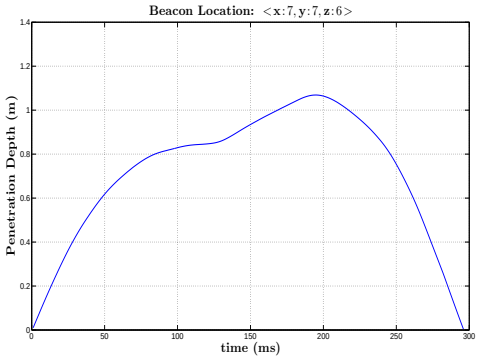


(f) Estimation error (m) along the z-axis

Figure 4.19: Basic SLAM-based haptic feedback experiment, beacon 1. Estimated and true location vs. time (left); estimation errors vs. time (right).

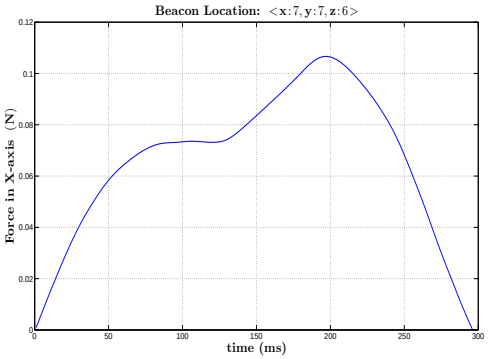


(a) Magnitude of the reflected force

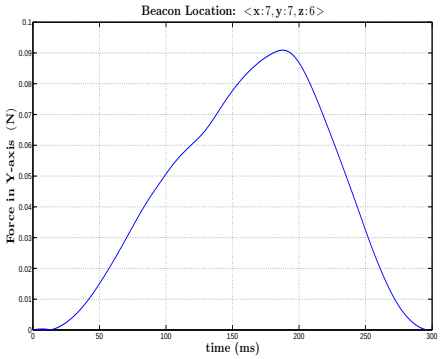


(b) The penetration distance $\mathbf{p}_d$

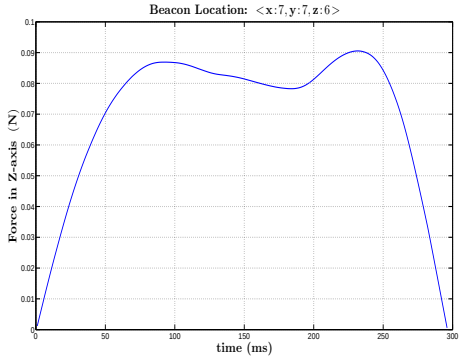
Figure 4.20: Basic SLAM-based haptic feedback experiment, beacon 1. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis

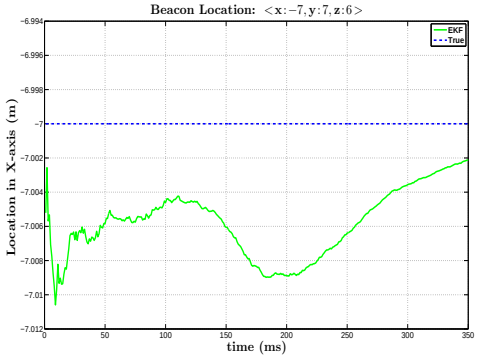


(b) Reflected force component along y-axis

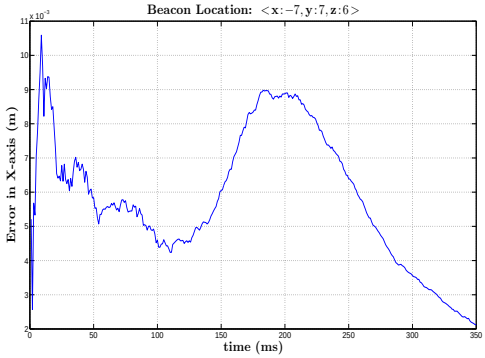


(c) Reflected force component along z-axis

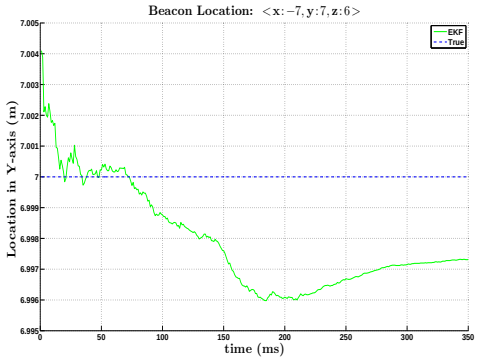
Figure 4.21: Basic SLAM-based haptic feedback experiment, beacon 1. The reflected force components along x, y, and z axes vs. time.



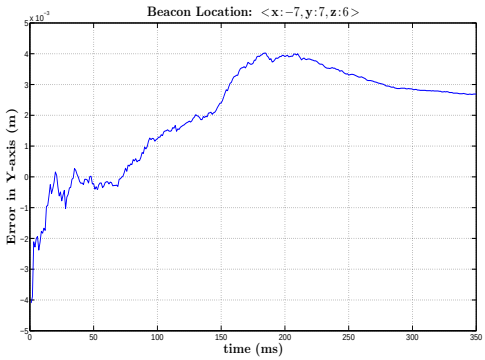
(a) The beacon's estimated vs. true location along the x-axis



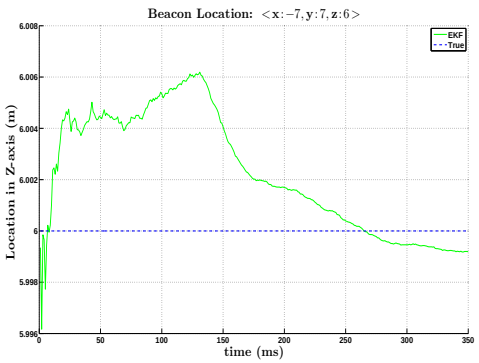
(b) Estimation error (m) along the x-axis



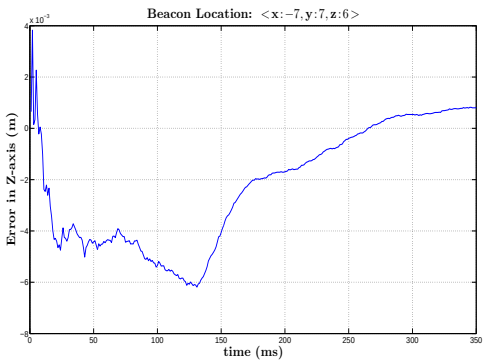
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

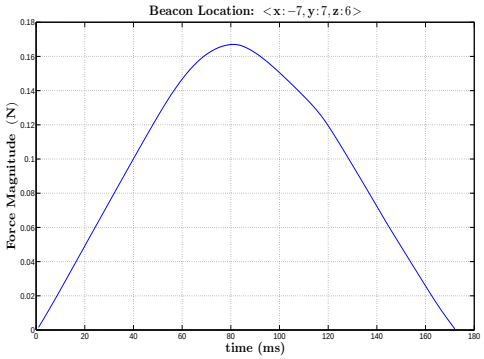


(e) The beacon's estimated vs. true location along the z-axis

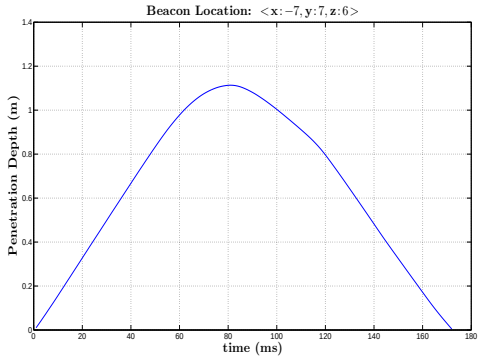


(f) Estimation error (m) along the z-axis

Figure 4.22: Basic SLAM-based haptic feedback experiment, beacon 2. Estimated and true location vs. time (left); estimation errors vs. time (right).

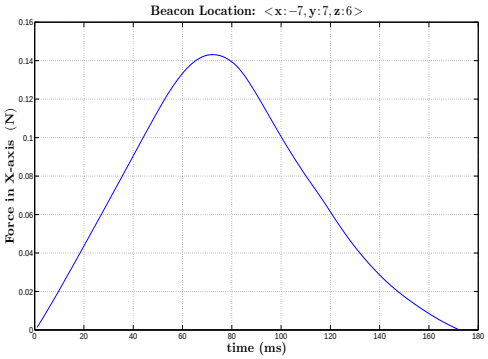


(a) Magnitude of the reflected force

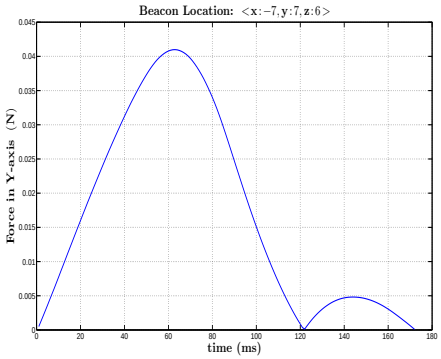


(b) The penetration distance $\mathbf{p}_d$

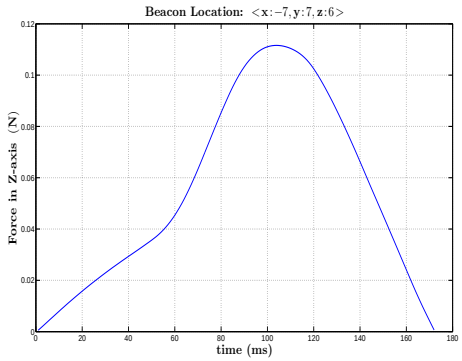
Figure 4.23: Basic SLAM-based haptic feedback experiment, beacon 2. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis

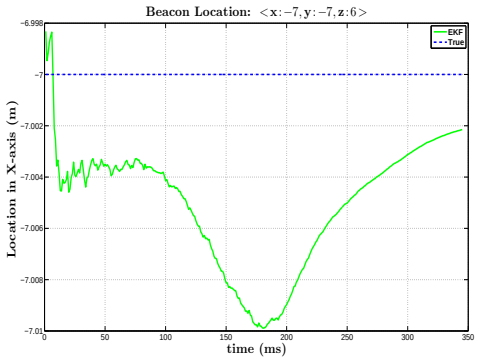


(b) Reflected force component along y-axis

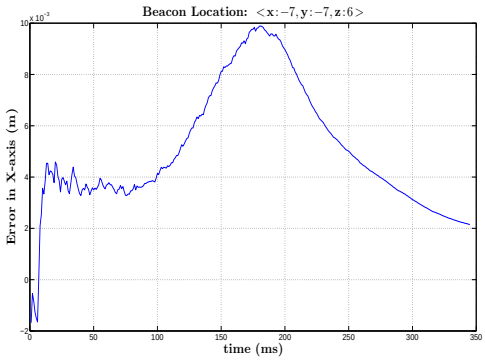


(c) Reflected force component along z-axis

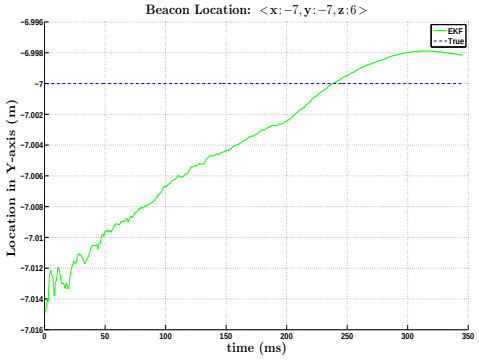
Figure 4.24: Basic SLAM-based haptic feedback experiment, beacon 2. The reflected force components along x, y, and z axes vs. time.



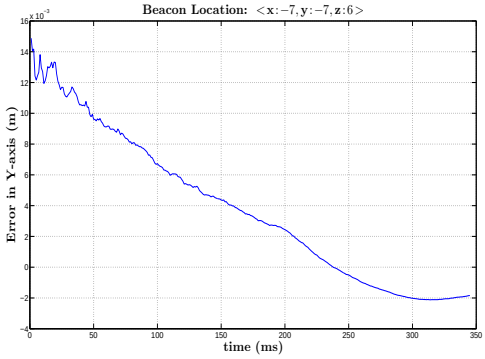
(a) The beacon's estimated vs. true location along the x-axis



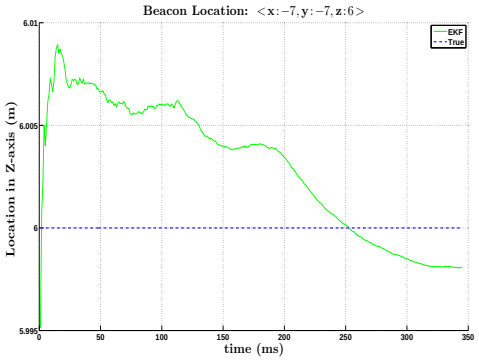
(b) Estimation error (m) along the x-axis



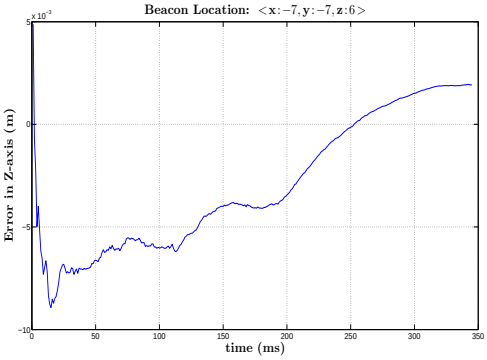
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

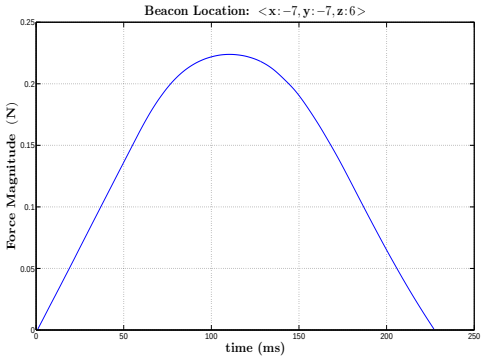


(e) The beacon's estimated vs. true location along the z-axis

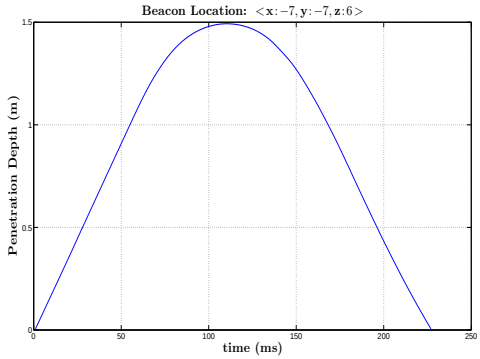


(f) Estimation error (m) along the z-axis

Figure 4.25: Basic SLAM-based haptic feedback experiment, beacon 3. Estimated and true location vs. time (left); estimation errors vs. time (right).

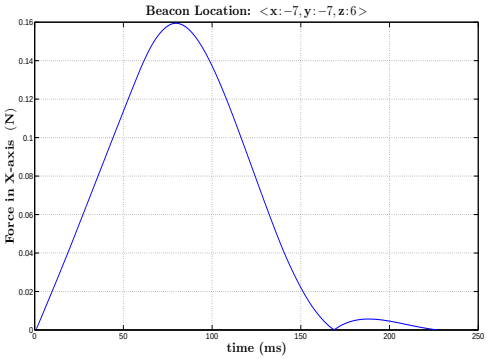


(a) Magnitude of the reflected force

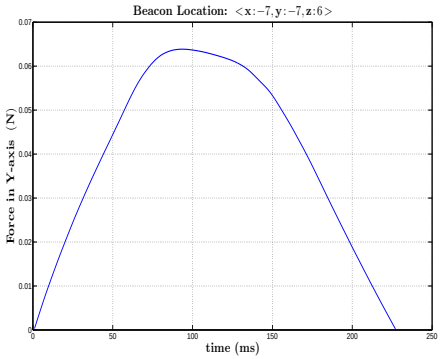


(b) The penetration distance $\mathbf{p}_d$

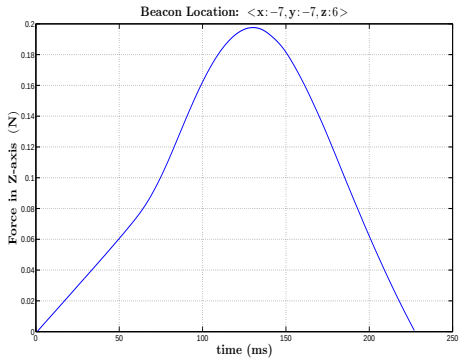
Figure 4.26: Basic SLAM-based haptic feedback experiment, beacon 3. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis

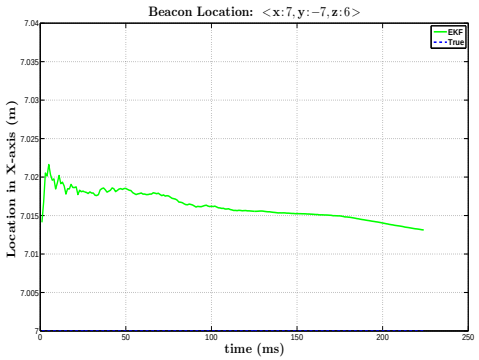


(b) Reflected force component along y-axis

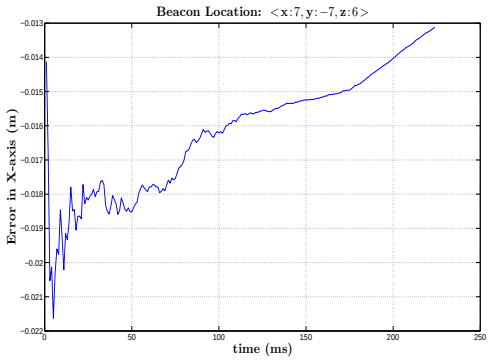


(c) Reflected force component along z-axis

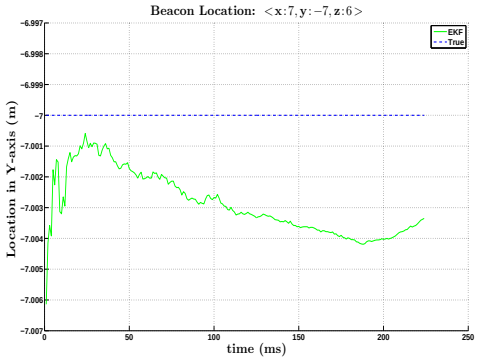
Figure 4.27: Basic SLAM-based haptic feedback experiment, beacon 3. The reflected force components along x, y, and z axes vs. time.



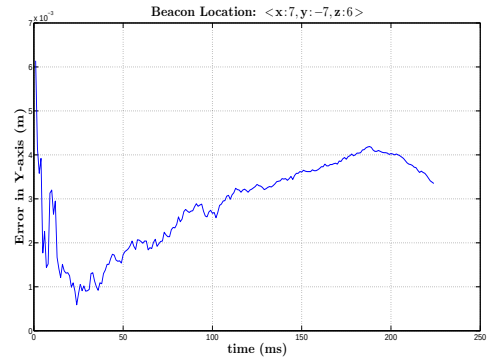
(a) The beacon's estimated vs. true location along the x-axis



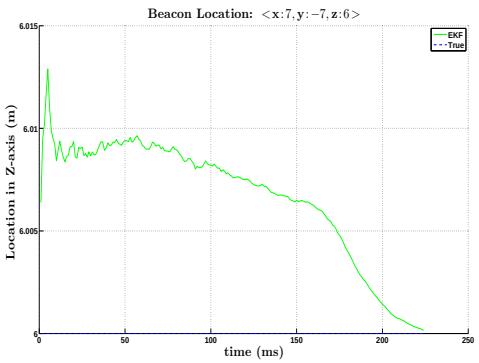
(b) Estimation error (m) along the x-axis



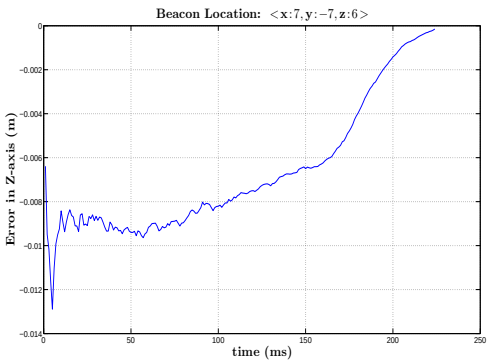
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

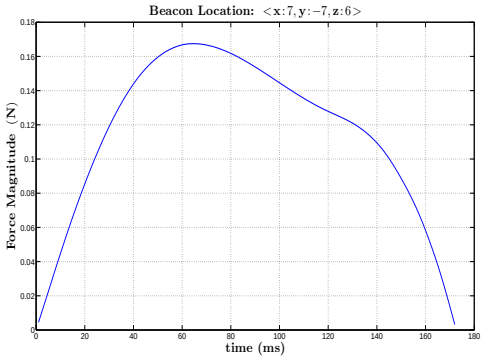


(e) The beacon's estimated vs. true location along the z-axis

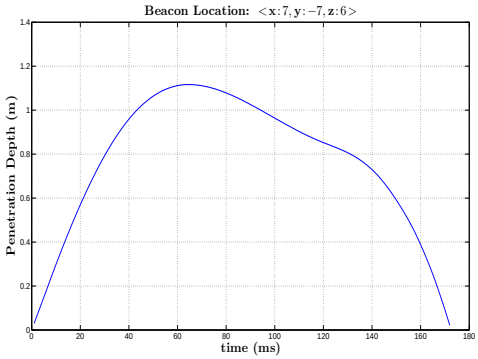


(f) Estimation error (m) along the z-axis

Figure 4.28: Basic SLAM-based haptic feedback experiment, beacon 4. Estimated and true location vs. time (left); estimation errors vs. time (right).

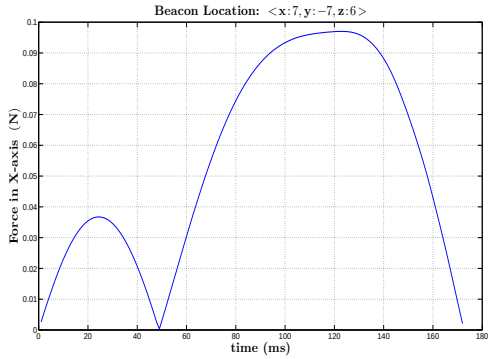


(a) Magnitude of the reflected force

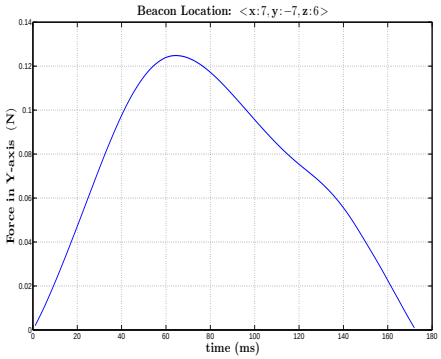


(b) The penetration distance $\mathbf{p}_d$

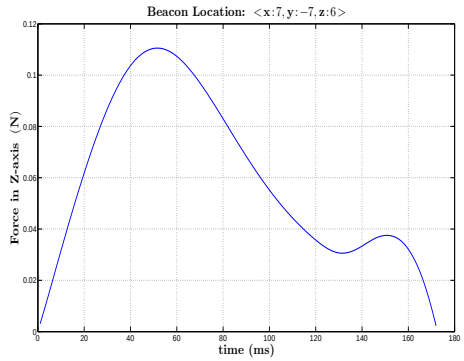
Figure 4.29: Basic SLAM-based haptic feedback experiment, beacon 4. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis



(b) Reflected force component along y-axis



(c) Reflected force component along z-axis

Figure 4.30: Basic SLAM-based haptic feedback experiment, beacon 4. The reflected force components along x, y, and z axes vs. time.

4.6.3 Robust SLAM-based haptic feedback algorithm

The basic SLAM-based haptic feedback algorithm presented in the previous section assumes that the parameters of the artificial potential field, particularly its radius, are fixed and the same for all obstacles. The major problem with this approach is that, in reality, the locations of the obstacles determined by the SLAM algorithm are not precisely known. In other words, the uncertainty of a beacon's estimate represented by the covariance matrix is not taken into account when using the basic algorithm from previous section. In 3D Gaussian distribution, the regions where the obstacle can be found with predefined confidence levels are represented by nested ellipsoids of different size as shown in Figure 4.31, where the red spheres represent possible true locations of the beacon. If the uncertainty of the beacon's location is large, then the actual location of the beacon may differ substantially from its estimated location; as a result, a fixed size APF built around the estimated location may be unable to prevent a collision between the UAV and an obstacle. This situation is illustrated in Figure 4.32, which shows the side effects of building a fixed size APF around an estimated location of the obstacle. These considerations motivate the modification to the basic SLAM-based haptic feedback algorithm where the size of the APF around an estimated location of the obstacle explicitly depends on the uncertainty of such an estimate.

Recall that, in the EKF-SLAM algorithm, an estimate of a beacon's location is characterized by its mean $\hat{\mathbf{m}}_j$ and a covariance matrix $\mathbf{P}_{\hat{\mathbf{m}}_j\hat{\mathbf{m}}_j}$,

$$\hat{\mathbf{m}}_j = \begin{bmatrix} \hat{\mathbf{m}}_{j,x} \\ \hat{\mathbf{m}}_{j,y} \\ \hat{\mathbf{m}}_{j,z} \end{bmatrix}, \quad \mathbf{P}_{\hat{\mathbf{m}}_j\hat{\mathbf{m}}_j} = \begin{bmatrix} \sigma^2_{\hat{\mathbf{m}}_{xx}} & \sigma_{\hat{\mathbf{m}}_{xy}} & \sigma_{\hat{\mathbf{m}}_{xz}} \\ \sigma_{\hat{\mathbf{m}}_{yx}} & \sigma^2_{\hat{\mathbf{m}}_{yy}} & \sigma_{\hat{\mathbf{m}}_{yz}} \\ \sigma_{\hat{\mathbf{m}}_{zx}} & \sigma_{\hat{\mathbf{m}}_{zy}} & \sigma^2_{\hat{\mathbf{m}}_{zz}} \end{bmatrix}$$

In Gaussian distribution, the mean is the centroid of the probability density function and the covariance matrix is a measure of the dispersion of possible beacon's locations around the mean. The eigenvectors and eigenvalues $\mathbf{P}_{\hat{\mathbf{m}}_j\hat{\mathbf{m}}_j}$ determine the ellipsoids that correspond to different confidence levels. Specifically, the eigenvectors represent the principal axes of the

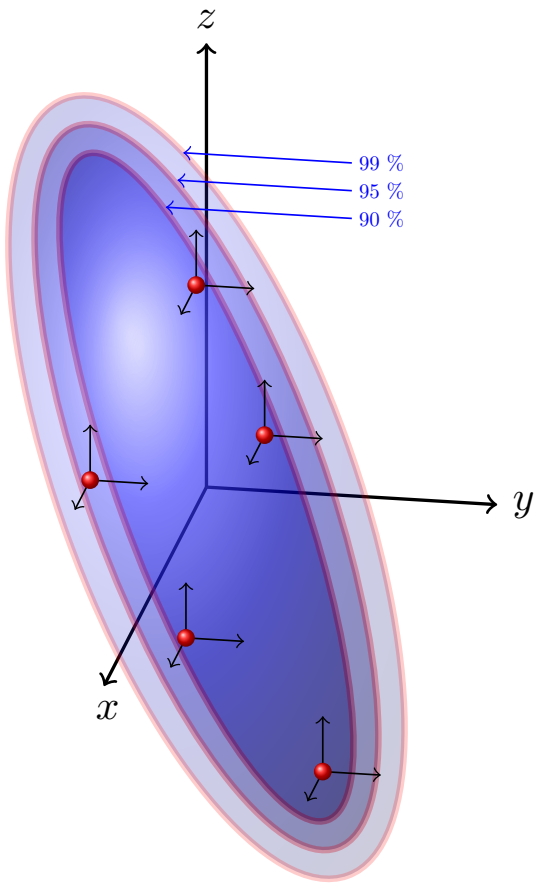


Figure 4.31: 3D Gaussian distribution represented as an ellipsoid with different confidence levels.

ellipsoid whereas the lengths of these axes are determined by the eigenvalues. We propose to build an APF of a spherical shape that would cover the ellipsoid with a predefined confidence level. This type of potential field can be called the uncertainty-dependent artificial potential field (UDAPF). Figure 4.33 illustrates the construction of UDAPF. The first step is to determine the maximum half-length of the ellipsoid's axes which is computed based on two parameters, namely the maximum eigenvalue λ_{max} of $\mathbf{P}_{\hat{\mathbf{m}}_j\hat{\mathbf{m}}_j}$ and the threshold of the validation gate χ^2 (see Chapter 3 for more information regarding the latter). The radius $\mathbf{p}_f$ of UDAPF is determined via the following formula,

$$\mathbf{p}_f := \sqrt{\chi^2 \, \lambda_{max}} + L_{pf}, \tag{4.4}$$

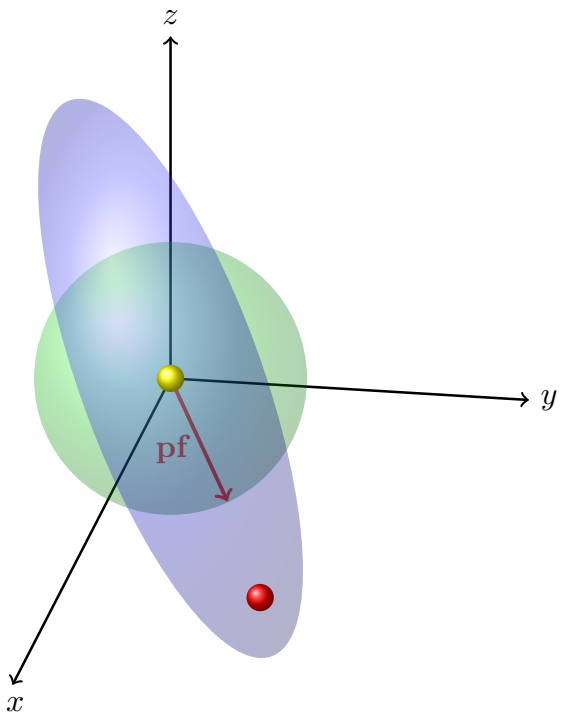


Figure 4.32: The side-effect of buidling a fixed APF where the green, red, yellow, blue shapes represent APF, a possible true location of a beacon, a beacon's estimate and the ellipsoid of the beacon's estimate.

where $L_{pf} > 0$ is a positive constant that determines the minimal distance between the ellipsoid and the border of the UDAPF, see Figure 4.33.

The Robust SLAM-based haptic feedback algorithm proposed in this section is shown below as **Algorithm 7**. The algorithm builds UDAPF around estimated location of the obstacles, and uses velocity dependent APF as described by Algorithm 5.

Algorithm 7 Robust SLAM-based Haptic Feedback Algorithm

for b = 1 to N_{k+1} **do**
  $\lambda_b = \sqrt{(\hat{\boldsymbol{\xi}}_x - \hat{\boldsymbol{m}}_{b,x})^2 + (\hat{\boldsymbol{\xi}}_y - \hat{\boldsymbol{m}}_{b,y})^2 + (\hat{\boldsymbol{\xi}}_z - \hat{\boldsymbol{m}}_{b,z})^2}$
end for
$\mathbf{J} = \underset{b}{\operatorname{argmin}}\, \lambda_b$
get $\mathbf{P}_{\hat{\mathbf{m}}_\mathbf{J}\hat{\mathbf{m}}_\mathbf{J}}$
get λ_{max} of $\mathbf{P}_{\hat{\mathbf{m}}_\mathbf{J}\hat{\mathbf{m}}_\mathbf{J}}$
$\mathbf{p}_f = \sqrt{\chi^2\, \lambda_{max}} + L_{pf}$
if $\lambda_\mathbf{J} < \mathbf{p}_f$ **then**
  $\dot{X}^R = [\hat{\boldsymbol{\xi}}_{\dot{x}}, \hat{\boldsymbol{\xi}}_{\dot{y}}, \hat{\boldsymbol{\xi}}_{\dot{z}}]$
  $X^R = [\hat{\boldsymbol{\xi}}_x, \hat{\boldsymbol{\xi}}_y, \hat{\boldsymbol{\xi}}_z]$

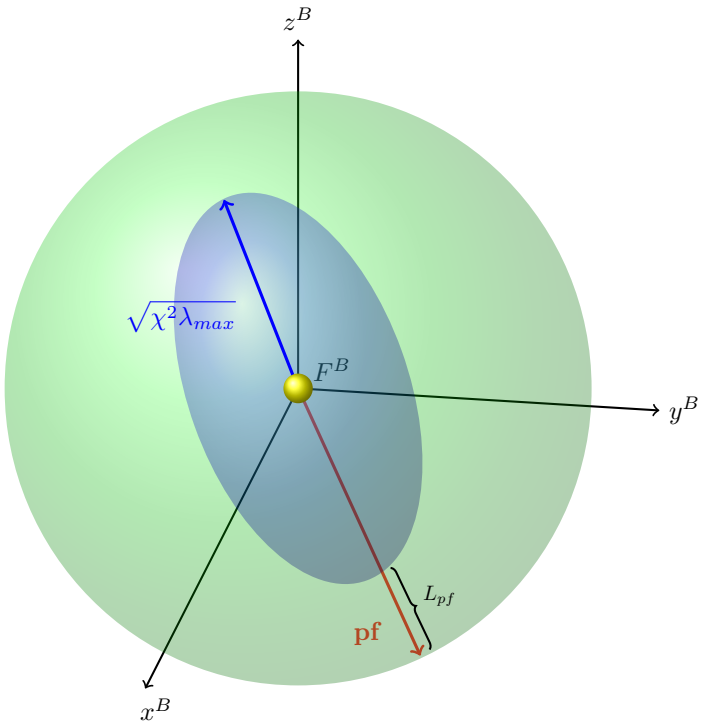


Figure 4.33: Uncertainty-dependent artificial potential field (UDAPF).

$X^O = [\hat{\mathbf{m}}_{\mathbf{J},x}, \hat{\mathbf{m}}_{\mathbf{J},y}, \hat{\mathbf{m}}_{\mathbf{J},z}]$
$\tilde{X} = X^O - X^R$
$\mathbf{p}_d = \mathbf{p}_f - \lambda_{\mathbf{J}}$
$\mathbf{F}_d = \frac{X^R - X^O}{\|X^R - X^O\|}$
$\phi = \cos^{-1}(\frac{(\dot{X}^R)^T \tilde{X}}{\|\dot{X}^R\|\|\tilde{X}\|})$
if $\phi < 90°$ **then**
    $\mathcal{F} = \underbrace{\mathbf{K}_p \cdot \mathbf{p}_d \cdot \mathbf{F}_d}_{stiffness} + \underbrace{\mathbf{D}_p \cdot \|\dot{X}^R\| \cdot \cos\phi \cdot \mathbf{F}_d}_{damping}$
else
    $\mathcal{F} = \mathbf{K}_p \cdot \mathbf{p}_d \cdot \mathbf{F}_d$
end if
end if

4.6.4 Semi-experimental results for robust SLAM-based haptic feedback algorithm

In this subsection, results of a semi-experimental investigation of the robust SLAM-based haptic feedback algorithm (**Algorithm 7**) are presented. The experiment performed is similar to the one described in Section 4.6.2, where performance of the basic SLAM-based haptic feedback algorithm was tested. In this experiment, four beacons (*i.e.* **B1, B2, B3, B4**) are simulated at the slave side, whose locations are shown in Table 4.5. Similarly to Section 4.6.2, the human operator controls the haptic device such that the quadrotor approaches the four beacons successively as shown in Figure 4.16, and penetrates its APF. In this experiment, position-to-velocity control strategy is utilized, where the position of the end-effector of the haptic device determines the reference velocity of the quadrotor. One centimeter increment of the position of the haptic device corresponds to reference velocity increment of 5 m/s. The backstepping velocity control algorithm is utilized; the control parameters and initial conditions are given in Table 4.6. Estimates of the obstacle locations are obtained using the EKF-SLAM algorithm (Algorithms 2 and 3). The quadrotor is equipped with a noisy sensor that provides a range r up to 3 m with uncertainty $\pm\sigma_r$ where σ_r is 0.8 m. Also, the sensor provides a bearing (*i.e.*, the azimuthal polar angles) with uncertainty $\pm\sigma_{b_{\phi,\theta}}$ where $\sigma_{b_{\phi,\theta}}$ is 0.8° degree. The quadrotor's model is not subject to any external noise (*i.e.*, zero process noise is assumed). The threshold of the validation gate is calculated from the chi-squared distribution table with 95% confidence level, which gives $\chi^2_{3,0.05} = 8.0$. For APF, the stiffness constant is chosen $\mathbf{K}_p = 0.1$ (N/m) and the damping constant $\mathbf{D}_p = 0.1$ (N·s/m).

Beacon jth	$\mathbf{m}_{j,x}$	$\mathbf{m}_{j,y}$	$\mathbf{m}_{j,z}$	Unit
1	1	1	1	(m)
2	1	1	12	(m)
3	1	1	22	(m)
4	1	1	32	(m)

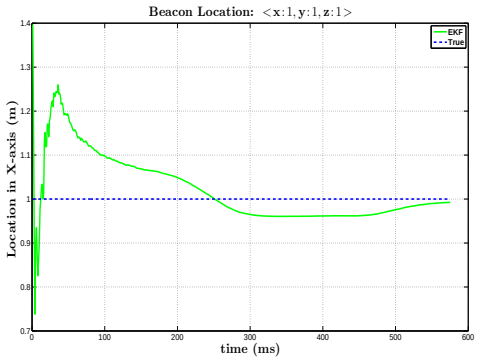
Table 4.5: Locations of beacons at the slave side

The results of the experiment are illustrated in Figures 4.34-4.45, where estimated and

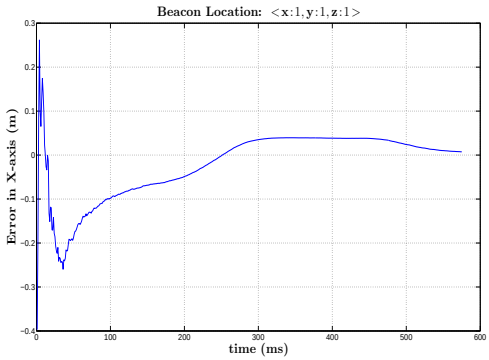
true location of the beacons vs. time, estimation errors vs. time, magnitude of the reflected force vs. time, APF penetration distance vs. time, as well as the x, y, and z-components of the reflected force vs. time are shown. The results of this experiment demonstrate feasibility of the developed robust SLAM-based haptic feedback algorithm. Similar conclusions that have been made in basic SLAM-based haptic feedback algorithm can be drawn in this section. Particularly, no false beacons have been recorded and errors in estimates of beacons' location are small and bounded.

	initial values	Gains	Unit
Altitude	$z(0) = 0$ $\dot{z}(0) = 0$	$\alpha_7 = 6.0$ $\alpha_8 = 5.1$	(m)
x	$x(0) = 0$ $\dot{x}(0) = 0$	$\alpha_9 = 3.0$ $\alpha_{10} = 3.0$	(m)
y	$y(0) = 0$ $\dot{y}(0) = 0$	$\alpha_{11} = 3.0$ $\alpha_{12} = 3.0$	(m)
Roll	$\phi(0) = 0$ $\dot{\phi}(0) = 0$	$\alpha_1 = 20.5$ $\alpha_2 = 20.0$	(rad)
Pitch	$\theta(0) = 0$ $\dot{\theta}(0) = 0$	$\alpha_3 = 20.5$ $\alpha_4 = 20.0$	(rad)
Yaw	$\psi(0) = 0$ $\dot{\psi}(0) = 0$	$\alpha_5 = 20.5$ $\alpha_6 = 20.0$	(rad)

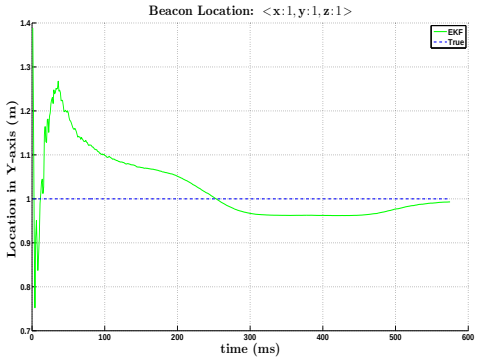
Table 4.6: Backstepping controller parameters for robust SLAM-based haptic feedback algorithm.



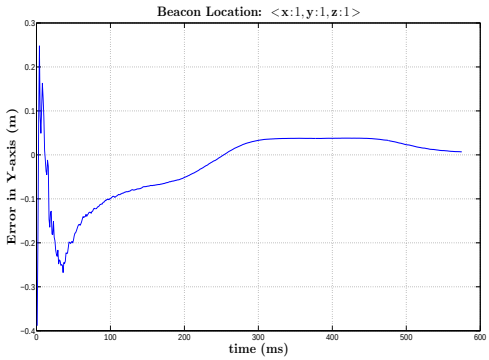
(a) The beacon's estimated vs. true location along the x-axis



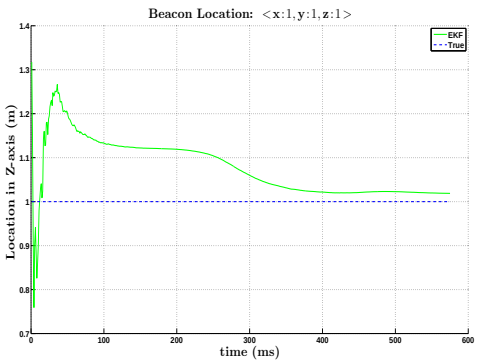
(b) Estimation error (m) along the x-axis



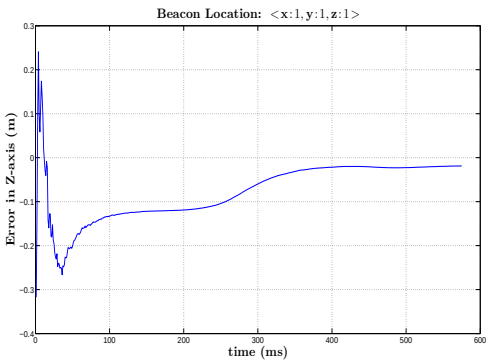
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

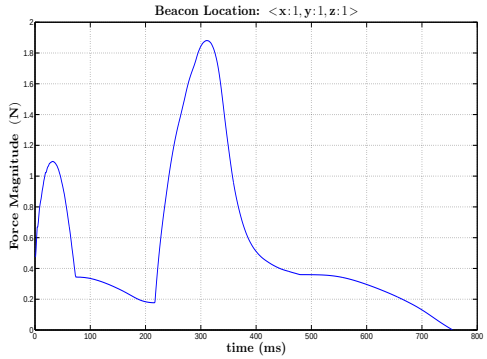


(e) The beacon's estimated vs. true location along the z-axis

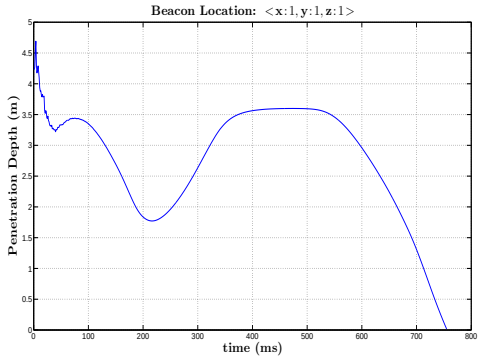


(f) Estimation error (m) along the z-axis

Figure 4.34: Robust SLAM-based haptic feedback experiment, beacon 1. Estimated and true location vs. time (left); estimation errors vs. time (right).

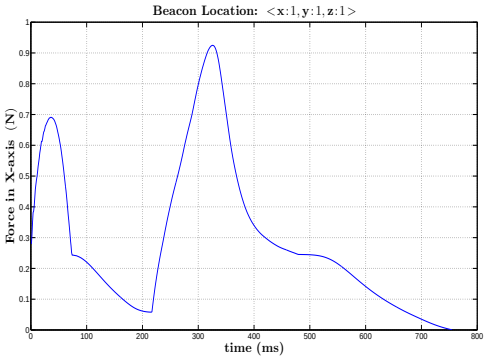


(a) Magnitude of the reflected force

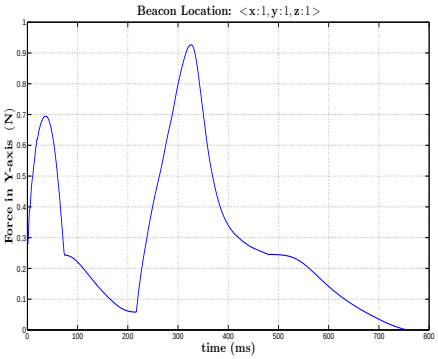


(b) The penetration distance $\mathbf{p}_d$

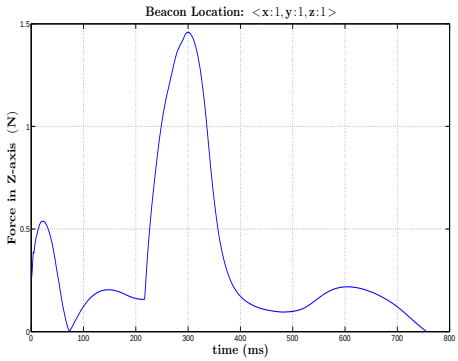
Figure 4.35: Robust SLAM-based haptic feedback experiment, beacon 1. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis

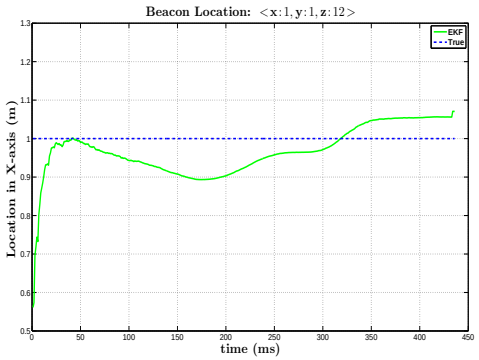


(b) Reflected force component along y-axis

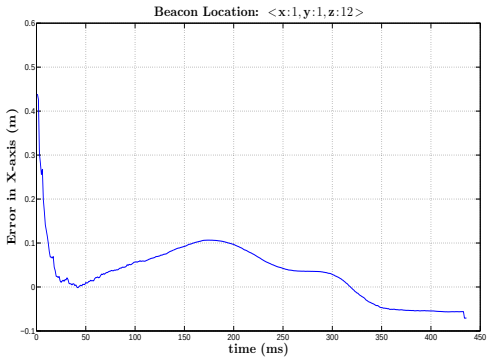


(c) Reflected force component along z-axis

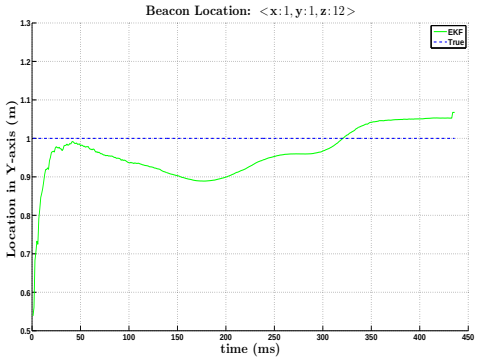
Figure 4.36: Robust SLAM-based haptic feedback experiment, beacon 1. The reflected force components along x, y, and z axes vs. time.



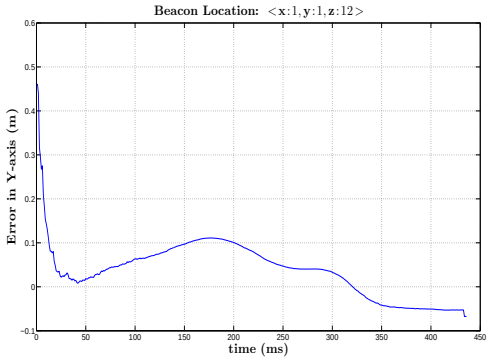
(a) The beacon's estimated vs. true location along the x-axis



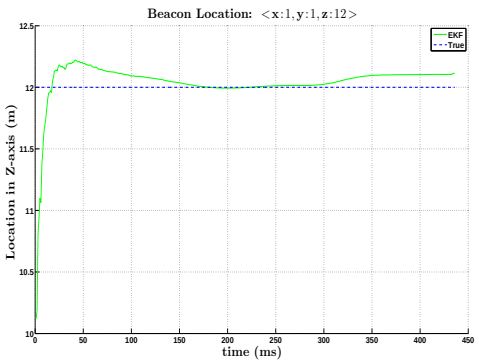
(b) Estimation error (m) along the x-axis



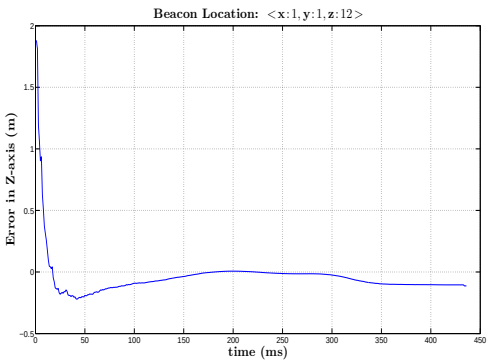
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

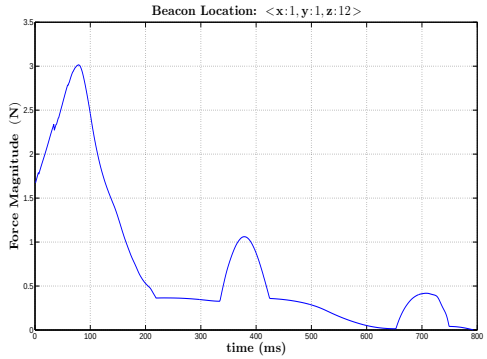


(e) The beacon's estimated vs. true location along the z-axis

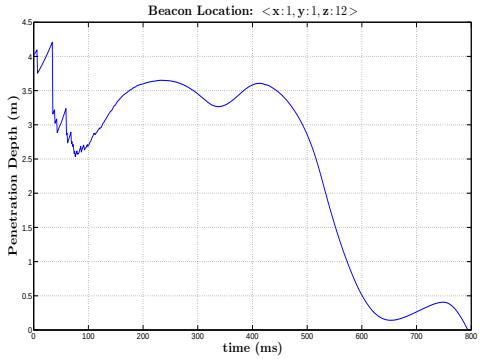


(f) Estimation error (m) along the z-axis

Figure 4.37: Robust SLAM-based haptic feedback experiment, beacon 2. Estimated and true location vs. time (left); estimation errors vs. time (right).

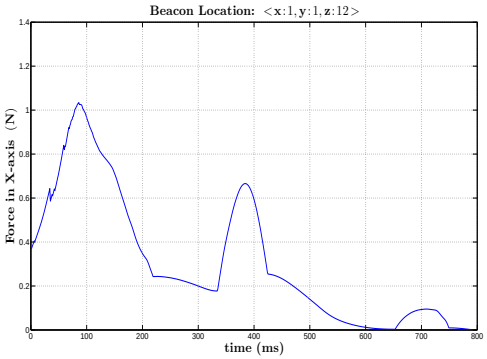


(a) Magnitude of the reflected force

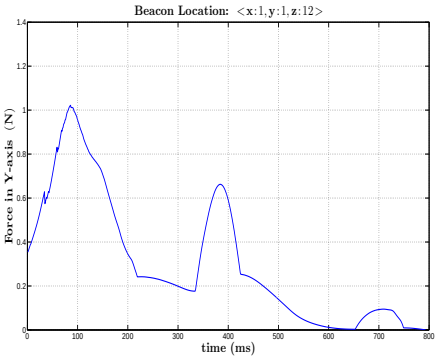


(b) The penetration distance $\mathbf{p}_d$

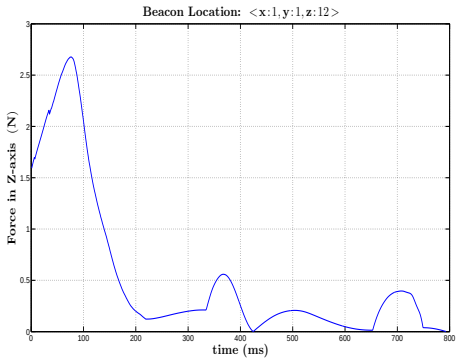
Figure 4.38: Robust SLAM-based haptic feedback experiment, beacon 2. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along y-axis

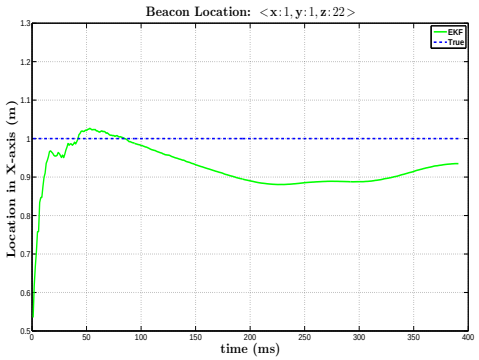


(b) Reflected force component along y-axis

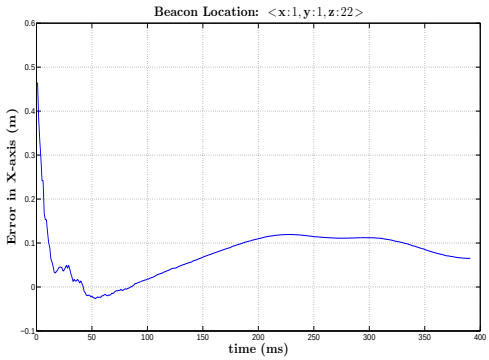


(c) Reflected force component along z-axis

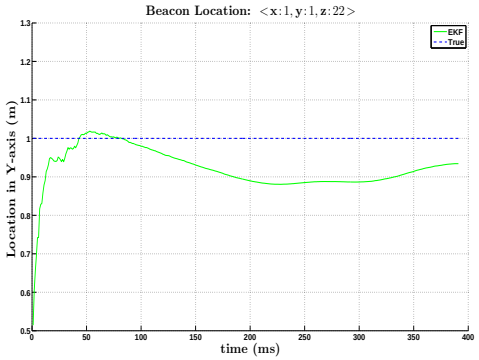
Figure 4.39: Robust SLAM-based haptic feedback experiment, beacon 2. The reflected force components along x, y, and z axes vs. time.



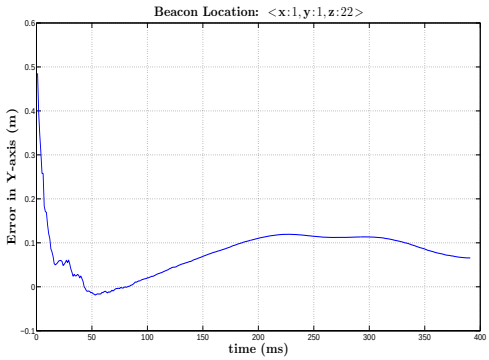
(a) The beacon's estimated vs. true location along the x-axis



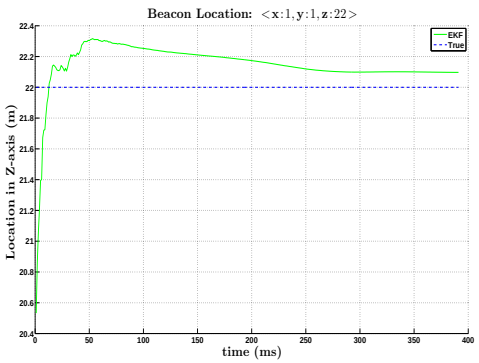
(b) Estimation error (m) along the x-axis



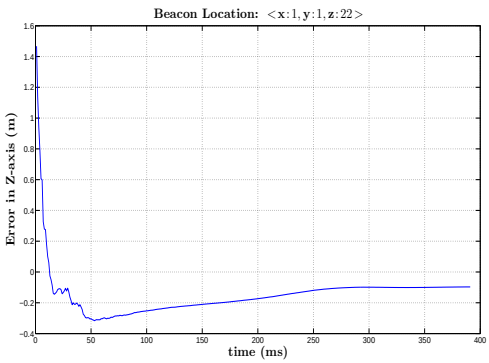
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

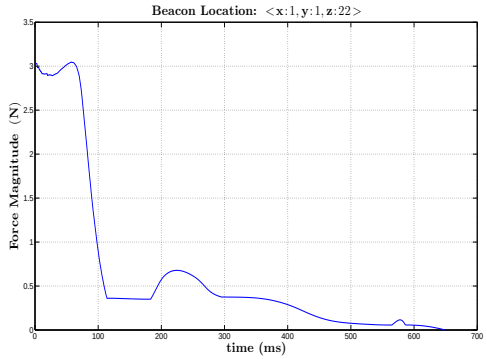


(e) The beacon's estimated vs. true location along the z-axis

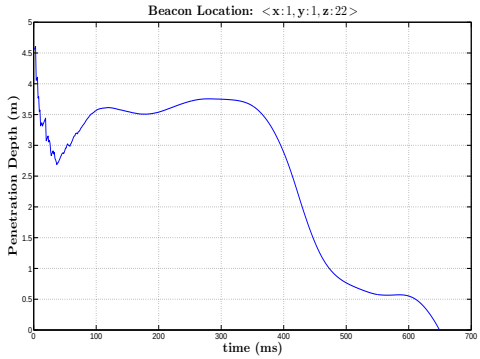


(f) Estimation error (m) along the z-axis

Figure 4.40: Robust SLAM-based haptic feedback experiment, beacon 3. Estimated and true location vs. time (left); estimation errors vs. time (right).

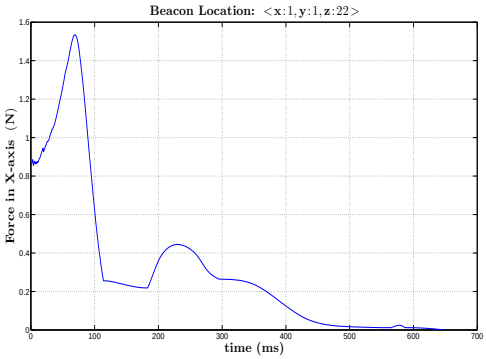


(a) Magnitude of the reflected force

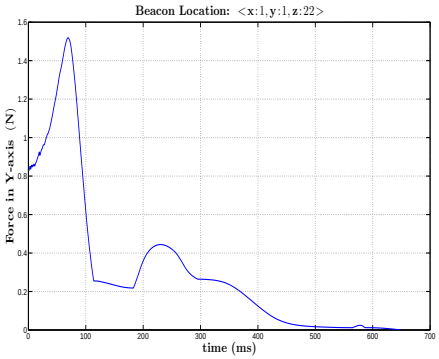


(b) The penetration distance $\mathbf{p}_d$

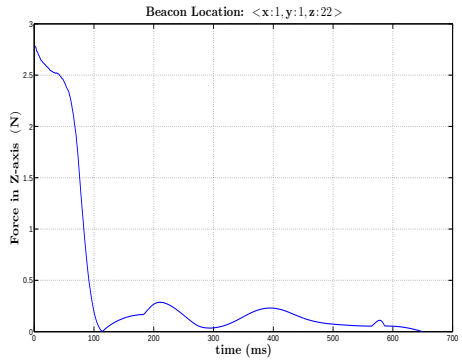
Figure 4.41: Robust SLAM-based haptic feedback experiment, beacon 3. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis

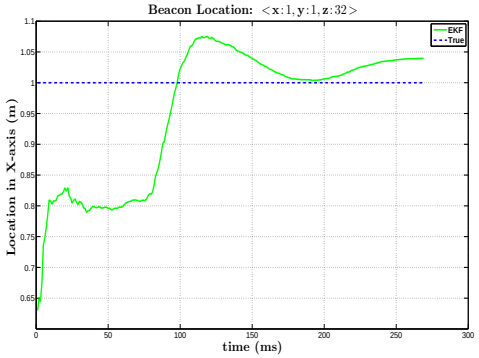


(b) Reflected force component along y-axis

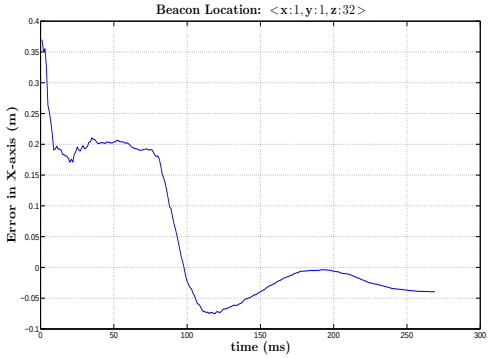


(c) Reflected force component along z-axis

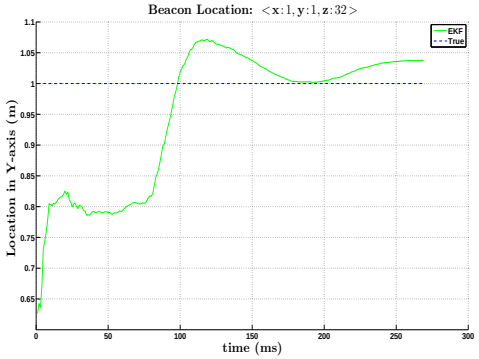
Figure 4.42: Robust SLAM-based haptic feedback experiment, beacon 3. The reflected force components along x, y, and z axes vs. time.



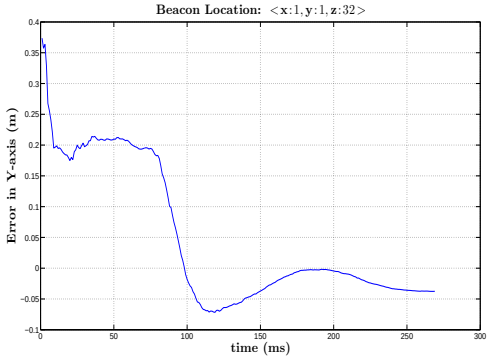
(a) The beacon's estimated vs. true location along the x-axis



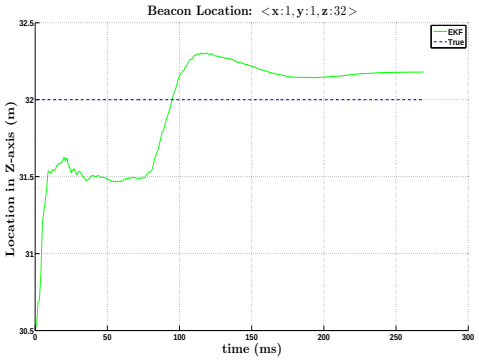
(b) Estimation error (m) along the x-axis



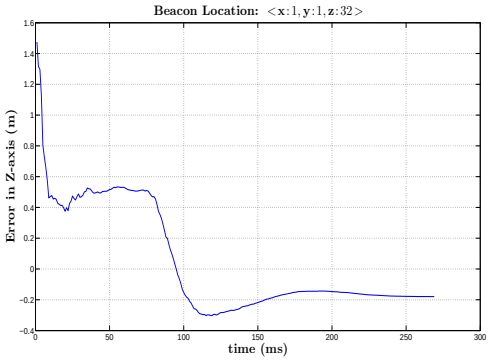
(c) The beacon's estimated vs. true location along the y-axis



(d) Estimation error (m) along the y-axis

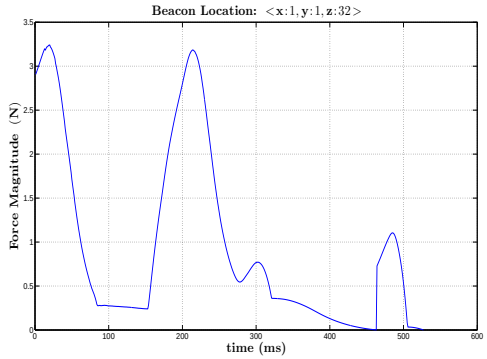


(e) The beacon's estimated vs. true location along the z-axis

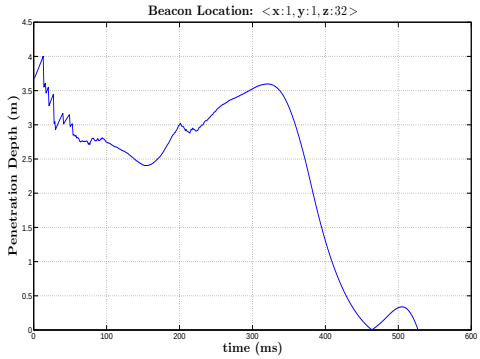


(f) Estimation error (m) along the z-axis

Figure 4.43: Robust SLAM-based haptic feedback experiment, beacon 4. Estimated and true location vs. time (left); estimation errors vs. time (right).

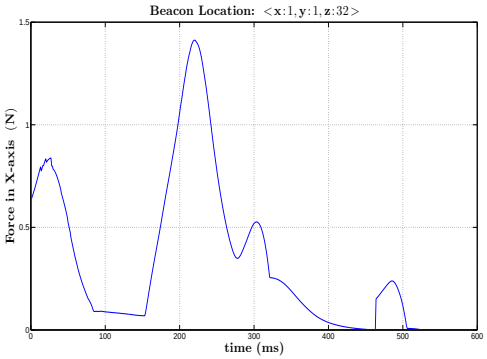


(a) Magnitude of the reflected force

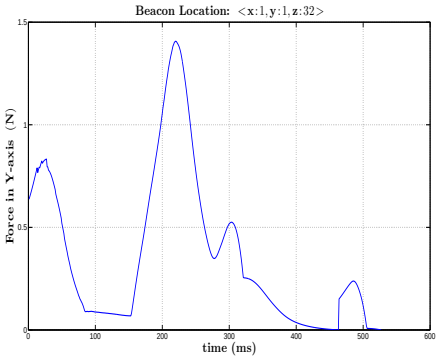


(b) The penetration distance $\mathbf{p}_d$

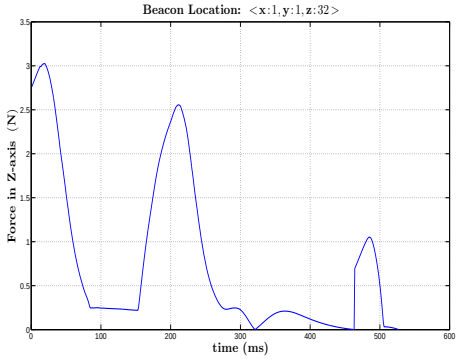
Figure 4.44: Robust SLAM-based haptic feedback experiment, beacon 4. Magnitude of the reflected force vs. time (left); APF penetration distance vs. time (right).



(a) Reflected force component along x-axis



(b) Reflected force component along y-axis



(c) Reflected force component along z-axis

Figure 4.45: Robust SLAM-based haptic feedback experiment, beacon 4. The reflected force components along x, y, and z axes vs. time.

4.7 Conclusion

In this chapter, we have investigated the possibility of using the EKF-SLAM algorithm to generate a virtual model of the remote environment, and subsequently use this model to provide the human operator with haptic and visual feedback. The haptic feedback is rendered by building artificial potential field around estimated locations of the obstacles in the virtual environment. Two algorithms for building SLAM-based haptic feedback are proposed. In the first algorithm, artificial potential field around the obstacles is assumed to be of fixed size, regardless of the level of uncertainty in the estimation of obstacle location. A possible drawback of this approach is that the uncertainty of the obstacle location is not taken into account. To address this issue, a second algorithm is proposed where the size of the APF changes depending on the level of uncertainty in the available estimates of the obstacles' locations. The semi-experimental results obtained demonstrate satisfactory performance of both basic and robust SLAM-based haptic feedback algorithms.

Chapter 5

Conclusion

5.1 Summary

In this thesis, a new type of haptic teleoperator system for remote control of UAVs have been proposed and developed, where the simultaneous localization and mapping (SLAM) algorithms have been used to generate the haptic feedback. More specifically, the haptic feedback is provided to the human operator through interaction with artificial potential field which is built around the obstacles in the virtual environment located at the master site of the teleoperator system. The obstacles in the virtual environment replicate essential features of the actual remote environment where the UAV executes its tasks. The state of the virtual environment is generated and updated in real time using EKF SLAM algorithms based on measurements performed by the UAV in the actual remote environment. Two methods for building haptic feedback from SLAM algorithms have been developed. The basic algorithm uses fixed size potential field around obstacles, while the robust algorithm changes the size of potential field around the obstacle depending on the amount of uncertainty in obstacle location, which is represented by the covariance estimate provided by EKF. Simulations and experimental results are presented that evaluate the performance of the proposed teleoperator system.

5.2 Future Research

The following recommendations and ideas can be considered as possible directions for future research:

- Implementation of a more realistic model for the remote environment which contains walls, corners, *etc*. In particular, both static and dynamic environments can be considered.

- Theoretical and experimental investigation of stability and performance of teleoperation systems with SLAM-based haptic feedback in the presence of time delays, both constant and time-varying.

- Validation of the proposed SLAM-based haptic feedback approach by performing real-time experiments.

- Development of new algorithms for SLAM-based haptic feedback, which could be based on implementation of different types of filters and/or new methods for building the artificial potential fields.

Bibliography

[1] Abdulmotaleb El Saddik. The potential of haptic technologies. *IEEE Instrumentation Measurement Magazine*, 10(1):10–17, Feb 2007.

[2] Sensable. *OpenHaptics Toolkit: Programmer's Guide*. 181 Ballardvale Street, Wilmington, MA 01887, 1999-2012.

[3] A. J. Silva, O. A. D. Ramirez, V. P. Vega, and J. P. O. Oliver. PHANToM OMNI haptic device: Kinematics and manipulability. In *Electronics, Robotics and Automotive Mechanics Conference CERMA '09.*, pages 193–198, 2009.

[4] P. Castillo, R. Lozano, and A. Dzul. *Modelling and Control of Mini-Flying Machines*. Springer London, 2005.

[5] George J. Vachtsevanos and Kimon P. Valavanis. Military and civilian unmanned aircraft. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 93–103. Springer Netherlands, 2014.

[6] Rahul Goel, Sapan M. Shah, Nitin K. Gupta, and N. Ananthkrishnan. Modeling, simulation and flight testing of an autonomous quadrotor. *Proceedings of ICEAE*, pages 1–7, 2009.

[7] Amr Nagaty, Sajad Saeedi, Carl Thibault, Mae Seto, and Howard Li. Control and navigation framework for quadrotor helicopters. *Journal of Intelligent and Robotic Systems*, 70(1-4):1–12, 2013.

[8] Anil Güçlü. *Attitude and altitude control of an outdoor quadrotor*. PhD thesis, Atilim University, 2012.

[9] George J. Vachtsevanos and Kimon P. Valavanis. Military and civilian unmanned aircraft. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 93–103. Springer Netherlands, 2014.

[10] A.M. Samad, N. Kamarulzaman, M.A. Hamdani, T.A. Mastor, and K.A. Hashim. The potential of Unmanned Aerial Vehicle (UAV) for civilian and mapping application. In *3rd IEEE International Conference on System Engineering and Technology (ICSET)*, pages 313–318, Aug 2013.

[11] A. Camargo, R. R. Schultz, Yi Wang, R. A. Fevig, and Qiang He. GPU-CPU implementation for super-resolution mosaicking of unmanned aircraft system (UAS) surveillance video. In *2010 IEEE Southwest Symposium on Image Analysis Interpretation (SSIAI)*, pages 25–28, May 2010.

[12] Xiangfeng Liu, Peng Chen, Xiaohua Tong, Shuang Liu, Shijie Liu, Zhonghua Hong, Lingyun Li, and Kuifeng Luan. UAV-based low-altitude aerial photogrammetric application in mine areas measurement. In *2012 Second International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, pages 240–242, June 2012.

[13] A. Idries, N. Mohamed, I. Jawhar, F. Mohamed, and J. Al-Jaroodi. Challenges of developing UAV applications: A project management view. In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)*, pages 1–10, March 2015.

[14] Xin Li and Lian Yang. Design and implementation of UAV intelligent aerial photography system. In *4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, pages 200–203, Aug 2012.

[15] Amazon. Amazon prime air, 2015. [Online; accessed 29-June-2015].

[16] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi. Linear vs. nonlinear control techniques for a quadrotor vehicle. In *7th International Symposium on Mechatronics and its Applications (ISMA)*, pages 1–10, April 2010.

[17] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In Georges Giralt and Gerhard Hirzinger, editors, *Robotics Research*, pages 613–625. Springer London, 1996.

[18] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *International Journal of Robotics Research*, 5(4):56–68, December 1986.

[19] S.B. Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte. Autonomous underwater simultaneous localization and map building. In *IEEE International Conference on Robotics and Automation ICRA '00*, volume 2, pages 1793–1798 vol.2, 2000.

[20] D. Ribas, P. Ridao, J. D. Tardos, and J. Neira. Underwater SLAM in a marina environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2007*, pages 1455–1460, Oct 2007.

[21] Jong-Hyuk Kim and S. Sukkarieh. Airborne simultaneous localization and map building. In *IEEE International Conference on Robotics and Automation ICRA 2003*, volume 1, pages 406–411, Sept 2003.

[22] Jorge Artieda, Jose M. Sebastian, Pascual Campoy, Juan F. Correa, Ivan F. Mondragon, Carol Martainez, and Miguel Olivares. Visual 3-D SLAM from UAVs. *Journal of Intelligent and Robotic Systems*, 55(4-5):299–321, 2009.

[23] Stefan Winkvist, Emma Rushforth, and Ken Young. Towards an autonomous indoor aerial inspection vehicle. *Industrial Robot: An International Journal*, 40(3):196–207, 2013.

[24] F Guth, L Silveira, S Botelho, P Drews, and P Ballester. Underwater SLAM: Challenges, state of the art, algorithms and a new biologically-inspired approach. In *5th IEEE RAS*

& EMBS International Conference on Biomedical Robotics and Biomechatronics, pages 981–986. IEEE, 2014.

[25] Seongsoo Lee and Sukhan Lee. Embedded Visual SLAM: Applications for Low-Cost Consumer Robots. *IEEE Robotics & Automation Magazine*, 20(4):83–95, 2013.

[26] Matja Å Mihelj and Janez Podobnik. Introduction to haptics. In *Haptics for Virtual Reality and Teleoperation*, volume 64 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 35–39. Springer Netherlands, 2012.

[27] Kang Wen, D. Necsulescu, and G. Basic. Development system for a haptic interface based on impedance/admittance control. In *The 3rd IEEE International Workshop on Haptic, Audio and Visual Environments and Their Applications HAVE 2004*, pages 147–151, Oct 2004.

[28] Kay M. Stanney. *Handbook Of Virtual Environments: Design, Implementation, And Applications*. Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, New Jersey, 2002.

[29] Ming C. Lin and Minguel A. Otaduy. *Haptic Rendering: Foundations, Algorithms, and Applications*. A K Peters, Ltd, Wellesley, MA 02482, 2008.

[30] S.D. Laycock and A.M. Day. A hybrid collision detection approach for the haptic rendering of deformable tools. In *Proceedings of Computer Graphics International 2004*, pages 148–155, 2004.

[31] S. Hirche and M. Buss. Human-oriented control for haptic teleoperation. *Proceedings of the IEEE*, 100(3):623–647, March 2012.

[32] D.A. Lawrence. Stability and transparency in bilateral teleoperation. *IEEE Transactions on Robotics and Automation*, 9(5):624–637, Oct 1993.

[33] Peter F. Hokayem and Mark W. Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035 – 2057, 2006.

[34] R. Anderson and M.W. Spong. Bilateral control of teleoperators with time delay. *IEEE Transactions on Automatic Control*, 34(5):494–501, May 1989.

[35] G. Niemeyer and J.-J.E. Slotine. Stable adaptive teleoperation. In *American Control Conference, 1990*, pages 1186–1191, May 1990.

[36] Thanh Mung Lam. *Haptic interface for UAV teleoperation*. TU Delft, Delft University of Technology, 2009.

[37] Abdulmotaleb El Saddik, Mauricio Orozco, Mohamad Eid, and Jongeun Cha. Human haptic perception. In *Haptics Technologies*, Springer Series on Touch and Haptic Systems, pages 45–66. Springer Berlin Heidelberg, 2011.

[38] Robert Jutte. Haptic perception: an historical approach. In Martin Grunwald, editor, *Human Haptic Perception: Basics and Applications*, pages 3–13. Birkhuser Basel, 2008.

[39] MatjaÅ Mihelj, Domen Novak, and Samo Begus. Haptic modality in virtual reality. In *Virtual Reality Technology and Applications*, volume 68 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 161–194. Springer Netherlands, 2014.

[40] G. Niemeyer and J.-J.E. Slotine. Stable adaptive teleoperation. In *American Control Conference, 1990*, pages 1186–1191, May 1990.

[41] I.G. Polushin, P. X. Liu, and Chung-Horng Lung. A control scheme for stable force-reflecting teleoperation over IP networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2005*, pages 2731–2736, Aug 2005.

[42] K. Kosuge, H. Murayama, and K. Takeo. Bilateral feedback control of telemanipulators via computer network. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 3, pages 1380–1385 vol.3, Nov 1996.

[43] G. Niemeyer and J.-J.E. Slotine. Using wave variables for system analysis and robot control. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1619–1625 vol.2, Apr 1997.

[44] E. J. Rodriguez-Seda, Dongjun Lee, and M. W. Spong. An experimental comparison study for bilateral internet-based teleoperation. In *2006 IEEE International Conference on Control Applications*, pages 1701–1706, Oct 2006.

[45] Allison M Okamura. Haptic feedback in robot-assisted minimally invasive surgery. *Current opinion in urology*, 19(1):102, 2009.

[46] C. Pacchierotti, F. Chinello, and D. Prattichizzo. Cutaneous device for teleoperated needle insertion. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS EMBS International Conference on*, pages 32–37, June 2012.

[47] D. Prattichizzo, C. Pacchierotti, and G. Rosati. Cutaneous force feedback as a sensory subtraction technique in haptics. *Haptics, IEEE Transactions on*, 5(4):289–300, Fourth 2012.

[48] S.B. Schorr, Z.F. Quek, R.Y. Romano, I. Nisky, W.R. Provancher, and A.M. Okamura. Sensory substitution via cutaneous skin stretch feedback. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2341–2346, May 2013.

[49] Nathan Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, Sept 2010.

[50] Caitlin Powers, Daniel Mellinger, and Vijay Kumar. Quadrotor kinematics and dynamics. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 307–328. Springer Netherlands, 2014.

[51] Yogianandh Naidoo, Riaan Stopforth, and Glen Bright. Quad-rotor unmanned aerial vehicle helicopter modelling & control. *International Journal of Advanced Robotic Systems*, 8(4):139–149, 2011.

[52] Peng Lu, Erik-Jan van Kampen, and Qiping P. Chu. Nonlinear quadrotor control with online model identification. In Jol Bordeneuve-Guib, Antoine Drouin, and Clment Roos, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 81–98. Springer International Publishing, 2015.

[53] Abdelkader Abdessameud and Abdelhamid Tayebi. Background and preliminaries. In *Motion Coordination for VTOL Unmanned Aerial Vehicles*, Advances in Industrial Control, pages 11–26. Springer London, 2013.

[54] Guowei Cai, Ben M. Chen, and Tong Heng Lee. Coordinate systems and transformations. In *Unmanned Rotorcraft Systems*, Advances in Industrial Control, pages 23–34. Springer London, 2011.

[55] Qiong Hu, Qing Fei, Qinghe Wu, and Qingbo Geng. Research and application of nonlinear control techniques for quadrotor UAV. In *Proceedings of the 31st Chinese Control Conference (CCC)*, pages 706–710, July 2012.

[56] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[57] Vladimir Dobrokhodov. Kinematics and dynamics of fixed-wing UAVs. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 243–277. Springer Netherlands, 2014.

[58] Swee King Phang, Kun Li, Ben M. Chen, and Tong H. Lee. Systematic design methodology and construction of micro aerial quadrotor vehicles. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 181–206. Springer Netherlands, 2014.

[59] J. Seddon. *Basic helicopter aerodynamics : an account of first principles in the fluid mechanics and flight dynamics of the single rotor helicopter*. Oxford: BPS Professional, 1990.

[60] O. Araar and N. Aouf. Full linear control of a quadrotor UAV, LQ vs. H_∞. In *2014 UKACC International Conference on Control*, pages 133–138, July 2014.

[61] S. Bouabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation ICRA 2005*, pages 2247–2252, April 2005.

[62] T. Madani and A. Benallegue. Backstepping control for a quadrotor helicopter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260, Oct 2006.

[63] E. de Vries and K. Subbarao. Backstepping based nested multi-loop control laws for a quadrotor. In *11th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1911–1916, Dec 2010.

[64] V. Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*, 22(2):212–222, Apr 1977.

[65] J. Liu and X. Wang. *Advanced Sliding Mode Control for Mechanical Systems: Design, Analysis and MATLAB Simulation*. Springer Berlin Heidelberg, 2011.

[66] K. Runcharoon and V. Srichatrapimuk. Sliding mode control of quadrotor. In *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on*, pages 552–557, May 2013.

[67] Rong Xu and U. Ozguner. Sliding mode control of a quadrotor helicopter. In *Decision and Control, 2006 45th IEEE Conference on*, pages 4957–4962, Dec 2006.

[68] H. Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, pages 1–6, April 2009.

[69] V. Mistler, A. Benallegue, and N.K. M'Sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 586–593, 2001.

[70] Miroslav Krstic, Petar V. Kokotovic, and Ioannis Kanellakopoulos. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1995.

[71] J. Zhou and C. Wen. *Adaptive Backstepping Control of Uncertain Systems*, volume 372. Springer-Verlag Berlin Heidelberg, 1st edition, 2008.

[72] Thor I. Fossen and Jan P. Strand. Tutorial on nonlinear backstepping: applications to ship control. *Modeling, identification and control*, 20(2):83, 1999.

[73] Hebatalla Mohamed Nabil El Kholy. Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches. Master's thesis, The American University in Cairo, 2014.

[74] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.

[75] Tim Bailey. *Mobile Robot Localization and Mapping in Extensive Outdoor Environments*. PhD thesis, The University of Sydney, 2002.

[76] C. M. Smith, J. J. Leonard, A. A. Bennett, and C. Shaw. Feature-based concurrent mapping and localization for auvs. In *OCEANS '97. MTS/IEEE Conference Proceedings*, volume 2, pages 896–901 vol.2, Oct 1997.

[77] F. Andert and L. Goormann. Combined grid and feature-based occupancy map building in large outdoor environments. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2065–2070, Oct 2007.

[78] Martin Adams, John Mullane, and Ebi Jose. *Robotic navigation and mapping with radar*. Artech House, 2012.

[79] M. Montemerlo and S. Thrun. The SLAM problem. In *FastSLAM*, volume 27 of *Springer Tracts in Advanced Robotics*, pages 13–26. Springer Berlin Heidelberg, 2007.

[80] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *Automatic Control, IEEE Transactions on*, 45(3):477–482, Mar 2000.

[81] J.J. LaViola. A comparison of unscented and extended kalman filtering for estimating quaternion motion. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2435–2440 vol.3, June 2003.

[82] E. A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *The IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

[83] M. St-Pierre and D. Gingras. Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 831–835, June 2004.

[84] Arthur G. O. Mutambara. *Decentralized Estimation and Control for Multisensor Systems*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1998.

[85] M. Adams, J. Mullane, E. Jose, and B. Vo. *Robotic Navigation and Mapping with RADAR*. Artech House, 2010.

[86] J. Castellanos and J. Tardos. *Mobile Robot Localization and Map Building*. Kluwer Academic Publishers, 1999.

[87] Y. Tsumaki and M. Uchiyama. Predictive display of virtual beam for space teleoperation. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 96*, volume 3, pages 1544–1549, Nov 1996.

[88] Chi-Cheng Cheng. *Predictor displays–theory development and application to towed submersibles*. PhD thesis, Massachusetts Institute of Technology, 1991.

[89] A.K. Bejczy and Won S. Kim. Predictive displays and shared compliance control for time-delayed telemanipulation. In *IEEE International Workshop on Intelligent Robots and Systems IROS '90 "Towards a New Frontier of Applications"*, pages 407–412, Jul 1990.

[90] A. K. Bejczy, W. S. Kim, and S. C. Venema. The phantom robot: predictive displays for teleoperation with time delay. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 546–551 vol.1, May 1990.

[91] T. Kotoku. A predictive display with force feedback and its application to remote manipulation system with transmission time delay. In *lEEE/RSJ International Conference on Intelligent Robots and Systems IROS 1992*, volume 1, pages 239–246, Jul 1992.

[92] T.M. Lam, H.W. Boschloo, M. Mulder, and M.M. van Paassen. Artificial force field for haptic feedback in UAV teleoperation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(6):1316–1330, Nov 2009.

[93] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(5):1179–1187, Sept 1989.

[94] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505, Mar 1985.

[95] Bruce H. Krogh. A generalized potential field approach to obstacle avoidance control. *Robotics Research: The Next Five Years and Beyond*, Aug 1984.

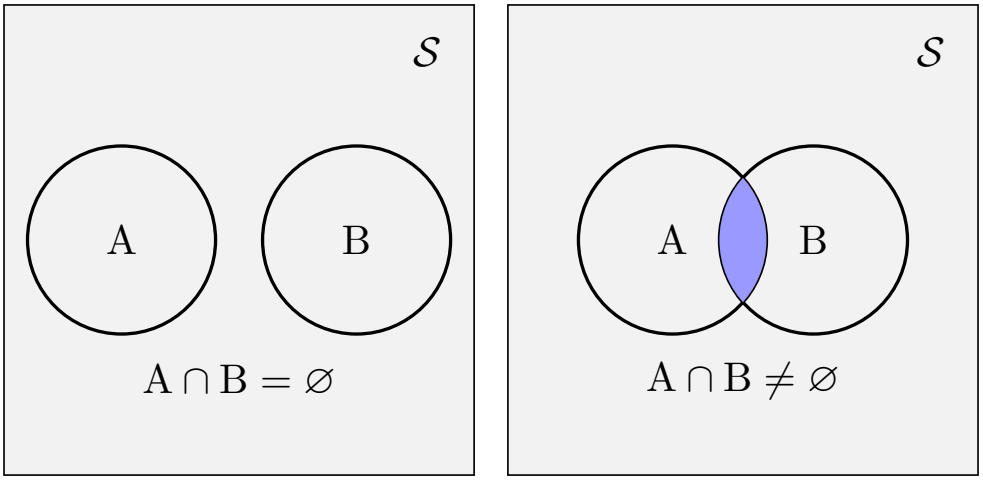
[96] `http://geomagic.com/en/`.

[97] T. Sansanayuth, I. Nilkhamhang, and K. Tungpimolrat. Teleoperation with inverse dynamics control for PHANToM Omni haptic device. In *Proceedings of SICE Annual Conference 2012*, pages 2121–2126, 2012.

Appendix A

Basic Probability Notions

This Appendix presents a brief description of some basic notions of probability theory in order to establish notation and provide appropriate background information for the material presented in Chapters 3 and 4. Probability theory is a mathematical discipline that deals with mathematical description of uncertainty. Nowadays, probability theory and its applications are ubiquitous due to the fact that majority of real world problems include some level of uncertainty; as a result, they can not be adequately solved using purely deterministic methods. In probability theory, an **experiment** is a process that can be performed repeatedly to yield **outcomes**. The set $\mathcal{S}$ of all possible outcomes of an experiment is called a **sample space**. For example, in rolling a die experiment, possible outcomes are $\{1, 2, 3, 4, 5, 6\}$, therefore, the sample space has six elements. For an experiment of tossing a coin, there are only two possible outcomes which are the head and the tail (*i.e.* $\mathcal{S}$ = {head,tail}). An **event** is a subset of the sample space that consists of the outcomes of interest. For example, in dice rolling, outcomes that consist of even numbers can be considered as an event. Probability theory assigns each outcome in $\mathcal{S}$ a numeric value between zero and one, called **probability**, which represents the likelihood of its occurrence in an experiment. For example, in tossing a coin experiment, $\mathcal{S}$ = {**H**,**T**}, where **H** and **T** are the outcomes of getting head and tail, respectively. Chances of each of these outcomes are 50%, which can be represented mathematically as $\mathbf{p}(\mathrm{H}) = \frac{1}{2}$, $\mathbf{p}(\mathrm{T}) = \frac{1}{2}$, where



(a) Independent Events (b) Dependent Events

Figure A.1: Venn diagram for the dependency and independency of two events.

p(H), **p**(T) are probabilities of the head and the tail outcomes, respectively. Experimenters are usually interested in the relationship between multiple events, and the effect of the occurrence of an event on other events. Let **A** and **B** be arbitrary events in a sample space $\mathcal{S}$. The event **A** is said to be independent of the event **B** (and *vice versa*), if there is no intersection between these events in the sample space (*i.e.*, $\mathbf{A} \cap \mathbf{B} = \varnothing$, where $\varnothing$ is an empty set). On the other hand, the event **A** is dependent on the event **B** (and *vice versa*), if the intersection between these events is non-empty (*i.e.*, $\mathbf{A} \cap \mathbf{B} \neq \varnothing$). This can be visualized by Venn diagram as shown in Figure A.1. For two events **A** and **B**, their joint probability is the probability of the simultaneous occurrence of event **A** and event **B**; such a probability is denoted by **p**(**A**,**B**). If **A** and **B** are independent events, their joint probability can be calculated as a product of their respective probabilities,

$$\mathbf{p}(\mathbf{A}, \mathbf{B}) = \mathbf{p}(\mathbf{A}) \cdot \mathbf{p}(\mathbf{B})$$

Experimenters are also frequently interested in determining the probability of the occurrence of particular events given the fact that other events occurred. For example, in a card game, if the outcome of drawing a card from a fairly shuffled deck belongs to the red suit, one might be interested in finding out the probability that the drawn card is the king of hearts. Such a

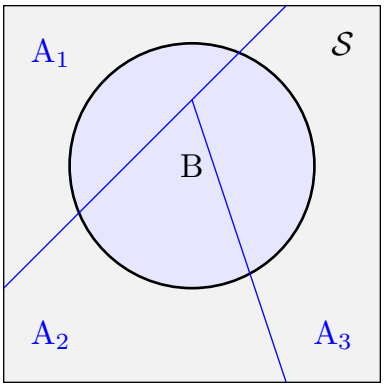


Figure A.2: Venn diagram that illustrates the total probability theorem.

probability is called **conditional probability**, and it is formulated as follows:

$$\mathbf{p(A|B)} = \frac{\mathbf{p(A \cap B)}}{\mathbf{p(B)}} \tag{A.1}$$

where $\mathbf{p(A|B)}$ is the probability of the occurrence of event **A** conditioned on the occurrence of event **B**; it is assumed in the above formula that $\mathbf{p(B)} \neq \mathbf{0}$. The notion of conditional probability can be extended to the case of multiple events, such that the occurrence of an event(s) can be conditioned on the occurrence of multiple other events. The **total probability theorem** states that, if there are multiple independent events that form a partition of a sample space $\mathcal{S}$, then the probability of an arbitrary event **B** can be computed as follows:

$$\begin{aligned} \mathbf{p(B)} &= \mathbf{p(A_1 \cap B)} + \cdots + \mathbf{p(A_n \cap B)} \\ &= \mathbf{p(A_1)} \cdot \mathbf{p(B|A_1)} + \cdots + \mathbf{p(A_n)} \cdot \mathbf{p(B|A_n)} \qquad \mathbf{n = 1, 2, ...} \end{aligned} \tag{A.2}$$

Figure A.2 shows the visualization of the total probability theorem. If the probability of an event **B** is known, and one is interested in finding out the probability of the event $\mathbf{A}_i$ conditioned on the event **B**, this can be done using the **Bayes' rule**, which states that

$$\begin{aligned} \mathbf{p(A_i|B)} &= \frac{\mathbf{p(A_i)} \cdot \mathbf{p(B|A_i)}}{\mathbf{p(B)}} \\ &= \frac{\mathbf{p(A_i)} \cdot \mathbf{p(B|A_i)}}{\mathbf{p(A_1)} \cdot \mathbf{p(B|A_1)} + \cdots + \mathbf{p(A_n)} \cdot \mathbf{p(B|A_n)}}. \end{aligned} \tag{A.3}$$

Appendix B

Phantom Omni Device

B.1 Introduction

Phantom Omni is currently one of the most popular haptic interfaces that provides the user with kinesthetic haptic feedback. The devices is manufactured by the Geomagic Touch company. It has six degrees of freedom (6-**DOF**) of position sensing and 3-**DOF** of force feedback. The maximum force the device can generate is 3.3 N, which is adequate to allow the user to feel the sense of touch. For more details about the technical specifications, the reader is addressed to [96].

Figure B.1: Phantom Omni

B.2 Forward and Inverse Kinematics of Phantom Omni

The mathematical equations of the forward and the inverse kinematics presented below are taken from [3]. The notation for variables and parameters are illustrated in Figure B.2. The

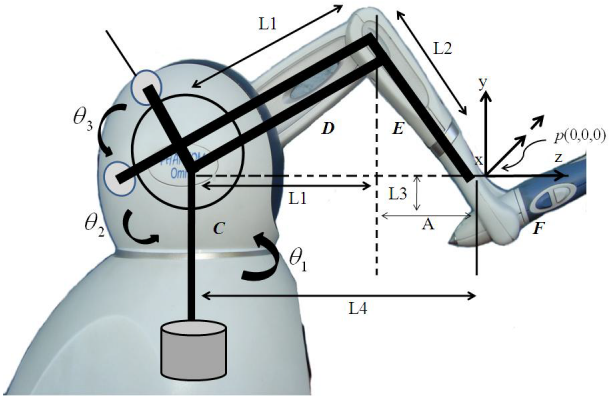


Figure B.2: Forward and Inverse Kinematics [3]

forward kinematics are described by the following equations,

$$x = -\sin\theta_1(L_2\sin\theta_3 + L_1\cos\theta_2),$$

$$y = -L_2\cos\theta_3 + L_1\sin\theta_2 + L_3,$$

$$z = L_2\cos\theta_1\sin\theta_3 + L_1\cos\theta_1\cos\theta_2 - L_4,$$

and the inverse kinematics equations are

$$\theta_1 = -a\tan 2(x, z + L_4),$$

$$\theta_2 = -\gamma + \beta,$$

$$\theta_3 = \theta_2 + \alpha - \frac{\pi}{2},$$

where

$$R = \sqrt{x^2(z + L_4)^2},$$

$$r = \sqrt{x^2(z + L_4)^2 + (y - L_3)^2},$$

$$\gamma = \cos^{-1}\left(\frac{L^2 + r^2 - L_2^2}{2L_1 r}\right),$$

$$\beta = a\tan 2(y - L_3, R),$$
$$\alpha = \cos^{-1}(\frac{L_1^2 + L_2^2 - r_2}{2L_1L_2}).$$

B.3 Jacobian Matrix

The Jacobian matrix $J(\theta)$ represents relationship between the joint velocities and the spatial velocity of the end-effector, as follows

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = J(\theta) \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

where

$$J(\theta) = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{bmatrix},$$

and the elements of the Jacobian are [97]

$$j_{11} = -\cos\theta_1(L_2\sin\theta_3 + L_1\cos\theta_2),$$
$$j_{12} = L_2\sin\theta_1\sin\theta_2,$$
$$j_{13} = -L_2\sin\theta_1\cos\theta_3,$$
$$j_{21} = 0,$$
$$j_{22} = L_1\cos\theta_2,$$
$$j_{23} = L_2\sin\theta_3,$$
$$j_{31} = -L_2\sin\theta_1\sin\theta_3 - L_1\sin\theta_1\cos\theta_2,$$
$$j_{32} = -L_1\sin\theta_2\cos\theta_1,$$
$$j_{33} = L_2\cos\theta_1\cos\theta_3.$$

B.4 OpenHaptics Toolkit

Phantom Omni device comes with a complete software toolkit called OpenHaptics which provides programmers with a variety of capabilities. For example, **Haptic Device API** (HDAPI) allows for access to the device hardware in order to render forces directly. On the other hand, **Haptic Library API** (HLAPI) allows programmers to design and build virtual environments that provide the user with kinaesthetic haptic feedback. The primary programming language of the toolkit is C/C++. For the Graphics library, OpenHaptic uses OpenGL API for rendering 2D and 3D virtual objects. Figure B.3 shows the integration of OpenHaptics toolkit.

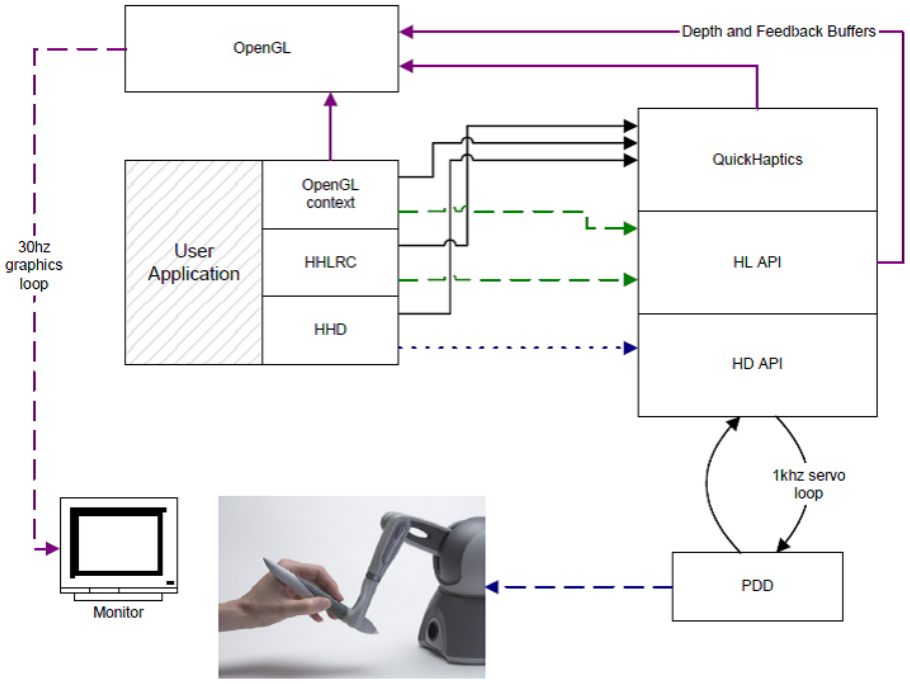


Figure B.3: OpenHaptics Toolkit [2].

Curriculum Vitae

Name:	Bandar Aldhafeeri
Post-Secondary Education and Degrees:	Electrical Engineering, Qassim University Qassim, Saudi Arabia 2004 - 2009 B.Sc. University of Western Ontario London, ON 2012 - 2015 M.Sc. (in progress)
Related Work Experience:	Teaching Assistant The University of Western Ontario 2013 - 2014