OPTIMAL CONTROLLER DESIGN FOR LEADER-FOLLOWER SYSTEM BETWEEN UGV AND UAV

Aidana Tassanbi, B.Eng. in Mechanical Engineering

Submitted in fulfillment of the requirements
for the degree of Master of Science
in Mechanical & Aerospace Engineering

NAZARBAYEV UNIVERSITY

School of Engineering and Digital Sciences
Department of Mechanical & Aerospace Engineering
Nazarbayev University

53 Kabanbay Batyr Avenue,
Astana, Kazakhstan, 010000

Supervisor: Assistant Professor Md. Hazrat Ali
Co-supervisor: Assistant Professor Ardak Kashkynbayev

April 2023

DECLARATION

I hereby, declare that this manuscript, entitled "*Optimal Controller Design for Leader-Follower System Between UGV and UAV*", is the result of my own work except for quotations and citations, which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or intentional institution.

Name: Aidana Tassanbi

Date: 11.04.2022

Abstract

Unmanned vehicles are already in wide use. For instance, they can be applied for military purposes, space exploration, in routine life, delivery services, etc. Unmanned vehicles are in great demand since they can be exploited at places that are dangerous or unachievable for humans. This research aims to create a leader-follower system for unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) that could have many applications. The drone will fly ahead and scan the surface from above for information. While the robot will create its own route in order to get to the point determined by the drone. This can be used to explore hard-to-reach places where the drone is conducting reconnaissance, and the robot is collecting materials with a robotic arm. For the project, the Quanser system was used, which includes a tracking system, communications, QDrone as a UAV, and QBot 2e as a UGV. A fuzzy controller was created for QDrone, which ensures flight stability. This thesis details the dynamic models of vehicles and the subtleties of tuning the controller. In addition, methods for creating a leader-follower algorithm are described in detail. In addition to everything, an obstacle avoidance algorithm for QBot 2e was created, which was added to the leader-follower algorithm.

The experiments conducted in this thesis with Fuzzy, PID and Fuzzy with prefilter controllers have shown that the Fuzzy controller performs more accurately and better in controlling the position of the quadcopter. An algorithm for the Leader-Follower system with obstacle avoidance was developed, and experiments were conducted to compare the Fuzzy and PID controllers under different conditions. Finally, the optimal Fuzzy controller was integrated into the Leader-Follower system with a Fuzzy controller. The results show that the proposed controller is effective in controlling the system and avoiding obstacles.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported and contributed to the completion of this thesis.

Firstly, I would like to thank Aziz Iskakov, a fourth-year undergraduate student who played a vital role in the establishment of the obstacle avoidance algorithm for the leader-follower algorithm, and who also helped me with the leader-follower algorithm itself. Aziz's contribution was invaluable, and I am grateful for his hard work and dedication.

I would also like to express my deep appreciation to my supervisor, Professor Md. Hazrat Ali, and co-supervisor, Professor Ardak Kashkynbayev, for their invaluable guidance, advice, and support throughout the entire research process. Their expertise, insights, and encouragement were instrumental in shaping my ideas and helping me stay focused and motivated.

Furthermore, I would like to thank Professor Ton Duc Do from the Robotics Department, who provided me with invaluable assistance in developing the fuzzy controller. His expertise in this area was invaluable, and I am grateful for his willingness to share his knowledge with me.

I am also grateful to the Head of Mechanical and Aerospace Department, Professor Essam Shehab, for providing me with feedback throughout my work. His support and encouragement were greatly appreciated.

Finally, I would like to express my gratitude to all my colleagues, friends, and family members, who also contributed to this research and helped me in various ways.

Once again, thank you to everyone who supported me during this journey.

Table of Contents

Abstract 2

Acknowledgements 3

Table of Contents 4

List of Abbreviations & Symbols 5

List of Tables 5

List of Figures 6

Chapter 1 – Introduction 8

1.1. Background 8

1.2. Problem statement 8

1.3. Motivation 8

1.4. Aim and objectives 9

1.5. Scope and constraints 9

Chapter 2 – Literature Review 11

2.1. Overview 11

2.2. Leader-follower algorithm and obstacle avoidance 11

2.3. Optimal controller 12

2.4. Research gap analysis 13

Chapter 3 – Methodology 15

3.1. Overview 15

3.2. Dynamic modeling 16

3.2.1. Dynamic Modelling of UAV 16

3.2.2. Dynamic Modelling of UGV 18

3.3. Communication and localization 19

3.4. Leader-follower algorithm 20

3.5. Obstacle avoidance algorithm 23

3.5.1. Attaching Sensors to Raspberry PI 3 Model b+ 23

3.5.2. Obstacle Avoidance Algorithm Creation 24

3.6. Fuzzy logic controller design 27

Chapter 4 – Results and Discussions: Leader-Follower System with Fuzzy controller 32

4.1. Experimental Setup 32

4.2. Experimental Results: Obstacle Avoidance Algorithm 34

4.3. Experimental Results: Implementation of Fuzzy Logic Controller 40
4.3. Experimental Results: Final tests of the Leader-Follower system with optimal controller 50
4.4. Experimental Results: Final tests of the obstacle avoidance algorithm in leader-follower system with the optimal controller 52
Chapter 5 – Conclusions and Future Works 55
5.1. Conclusion 55
5.2. Contribution to Knowledge 56
5.3. Future Work 56
References 58
Appendices 0

List of Abbreviations & Symbols

UGV	Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
FLC	Fuzzy Logic Controller

List of Tables

TABLE 3.1: QUADROTOR PARAMETER DESCRIPTION....17
TABLE 3.2: FUZZY RULES TABLE....30
TABLE 4.1: THE OBSTACLE AVOIDANCE EXPERIMENT RESULTS....35
TABLE 4.2. TECHNICAL SPECIFICATION OF PEDESTAL FAN....40
TABLE 4.3. HOVER TEST RESULTS FOR 3 TYPES OF CONTROLLERS WITH AND WITHOUT DISTURBANCE....48

List of Figures

Figure 1.1: Defining the scope of research 10
Figure 3.1: System communication scheme 15
Figure 3.2: Schematic diagram for UAV 17
Figure 3.3: Schematic diagram for UGV 19
Figure 3.4: Algorithm for the follower 21
Figure 3.5: Mission server algorithm 22
Figure 3.6: Algorithm for the leader 23
Figure 3.7: Raspberry PI 3 model B+ GPIO [29] 24
Figure 3.8: Connection of an ultrasonic sensor to raspberry Pi 24
Figure 3.9: Obstacle avoidance algorithm concept 25
Figure 3.10: Path generation by the obstacle avoidance algorithm 26
Figure 3.11: a) Block diagram of control scheme; b) Integration of FLC into Simulink model 28
Figure 3.12: Flowchart for operation procedure for controller tests 29
Figure 3.13: Membership functions design 31
Figure 4.1: Workspace covered by safety net 32
Figure 4.2: OptiTrack Camera 33
Figure 4.3: QUANSER's QDrone 34
Figure 4.4: QUANSER's QBot 2e and the attached HC-SR04 sensors 34
Figure 4.5:The path generated by obstacle avoidance algorithm which must be followed by QBot 2e in ideal case 35
Figure 4.6: Obstacle avoidance algorithm turns on from the beginning 37
Figure 4.7: QBot 2e avoids the first obstacle 37
Figure 4.8: Leader-follower algorithm with obstacle avoidance test (WP – waypoint, OB – obstacle) 38
Figure 4.9: QBot 2e collides with the second obstacle 39
Figure 4.10: QBot 2e avoids the second obstacle 40
Figure 4.11: Position error and velocity error (X) vs time for hover test with no disturbance applied 42
Figure 4.12: Position error and velocity error (Yaw) vs time for hover test with no disturbance applied 43
Figure 4.13: Position error and velocity error (Y) vs time for hover test with disturbance applied at 45◦ angle 44
Figure 4.14: Yaw vs time graphs for position control test with no disturbance applied 45
Figure 4.15: Z vs time graphs for position control test with disturbance applied at 0◦ angle 46
Figure 4.16: Y vs time graphs for position control test with disturbance applied at 45◦ angle 48
Figure 4.17: Location of waypoints for Leader-Follower test (WP - waypoint) 50
Figure 4.18: Leader-follower results for PID 51
Figure 4.19: Leader-follower results for Fuzzy 52
Figure 4.20: Waypoints and obstacle locations for obstacle avoidance tests with the optimal controller 53
Figure 4.21: Performance of the obstacle avoidance algorithm in leader-follower system with the optimal controller 54
Figure A.1: Unsaturated controller commands vs time (Hover test, PID) 1
Figure A.2: Position error and Velocity error vs time (Hover test, PID) 2
Figure A.3: Position vs time (Hover test, PID) 4
Figure A.4: Unsaturated controller commands vs time (Position control test, PID) 5
Figure A.5: Position error and Velocity error vs time (Position control test, PID) 7

FIGURE A.6: POSITION VS TIME (POSITION CONTROL TEST, PID).....8
FIGURE A.7: UNSATURATED CONTROLLER COMMANDS VS TIME (HOVER TEST, FUZZY).....10
FIGURE A.8: POSITION ERROR AND VELOCITY ERROR VS TIME (HOVER TEST, FUZZY).....11
FIGURE A.9: POSITION VS TIME (HOVER TEST, FUZZY).....13
FIGURE A.10: UNSATURATED CONTROLLER COMMANDS VS TIME (POSITION CONTROL TEST, FUZZY).....14
FIGURE A.11: POSITION ERROR AND VELOCITY ERROR VS TIME (POSITION CONTROL TEST, FUZZY).....16
FIGURE A.12: POSITION VS TIME (POSITION CONTROL TEST, FUZZY).....17
FIGURE A.13: UNSATURATED CONTROLLER COMMANDS VS TIME (HOVER TEST, FUZZY CONTROLLER WITH PREFILTER).....19
FIGURE A.14: POSITION ERROR AND VELOCITY ERROR VS TIME (HOVER TEST, FUZZY CONTROLLER WITH PREFILTER).....20
FIGURE A.15: POSITION VS TIME (HOVER TEST, FUZZY CONTROLLER WITH PREFILTER).....22
FIGURE A.16: UNSATURATED CONTROLLER COMMANDS VS TIME (POSITION CONTROL TEST, FUZZY CONTROLLER WITH PREFILTER).....23
FIGURE A.17: POSITION ERROR AND VELOCITY ERROR VS TIME (POSITION CONTROL TEST, FUZZY CONTROLLER WITH PREFILTER).....25
FIGURE A.18: POSITION VS TIME (POSITION CONTROL TEST, FUZZY CONTROLLER WITH PREFILTER).....26

Chapter 1 – Introduction

1.1. Background

Today, process automation is an essential part of any advanced system. The demand for unmanned vehicles and other modes of transport is growing every year. The Unmanned Ground Vehicle (UAV) and Unmanned Aerial Vehicle (UGV) are robotic systems that operate on land or in the air without an onboard human operator. The vehicles can work under remote control by a human operator, or with various degrees of autonomy, up to fully autonomous vehicles which have no provision for human intervention. These robots can be used in space exploration, firefighting, exploration of the bowels and surface of the earth, military purposes, and everyday life. Also, they can be used to study places that are difficult to access or dangerous for humans or even to do routine work. Automated auxiliary robots, for example, are popular not only in manufacturing but also in everyday life, such as robotic vacuum cleaners, restaurant waiter robots, and others. Furthermore, there has recently been an increase in research aimed at developing leader-follower algorithms for multi-vehicle systems. These systems can be utilized in a variety of applications, ranging from military purposes to delivery services.

1.2. Problem statement

The objective of this research is to design an optimal controller for a Leader-Follower system composed of a quadrotor as a leader and a two-wheeled mobile robot as a follower, in which the quadrotor follows predetermined waypoints while the follower robot avoids obstacles and maintains a safe distance from the leader. The system employs Quanser equipment, including the QDrone, QBot 2e, OptiTrack localization system, communication tools, and ultrasonic sensors for obstacle detection.

1.3. Motivation

The Leader-Follower system is a common configuration used in various applications such as autonomous search and rescue, precision agriculture, and surveillance. In this system, it is essential to maintain a safe distance between the leader and the follower while ensuring that the follower avoids obstacles and follows the leader accurately. Optimal controller design plays a

crucial role in achieving these objectives. The integration of Quanser equipment and ultrasonic sensors offers a practical solution to implement the Leader-Follower system in real-world scenarios. Therefore, the motivation of this research is to develop an optimal controller for the Leader-Follower system that can accurately follow the leader and avoid obstacles in a safe and efficient manner.

1.4. Aim and objectives

The goal of this research is to develop a leader-follower system for unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) with obstacle avoidance. The UAV serves as a leader in this system, and it is given the coordinates of a predetermined path that it must follow according to its algorithm. While the UGV is a follower, it must avoid obstacles in order to follow the leader. Aside from that, it is necessary to create an appropriate controller that will assure continual flying. The main objectives are as follows:

- Obstacle avoidance algorithm development for Unmanned Ground Vehicle (UGV)
- Development of control algorithm for leader, follower and mission server
- Real-time monitoring and surveillance of drone and robot
- Development of a controller for sustainable flight

The expected outcome of this project is to develop a sustainable leader-follower system between quadrotor and 2WMR (two-wheeled mobile robot) with obstacle avoidance.

1.5. Scope and constraints

This research aims to create a sustainable leader-follower system between unmanned vehicles, which is able to avoid obstacles. The algorithm should be written in a way that the system will be almost fully autonomous.

In this system a quadrotor acts as a leader and follows waypoints, which are preliminary assigned in algorithm. While a 2WMR (two-wheeled mobile robot) follows the leader and avoids the obstacles. In addition, it is required to create an optimal controller, which will control the flight

and ensure its safety. For this, Quanser equipment was used, which includes QDrone, QBot 2e, OptiTrack localization system and communication tools. Additionally ultrasonic sensors were integrated into the system for obstacle detection.

The scope of the study is the development of algorithms, the development of controllers, the implementation of sensors, the establishment of communication and the localization of UGV.

The scope of research does not include building vehicles, improvement of aerodynamics, studying of stability of flight with payload, and conducting experiments outside of the workspace (Figure 1.1).

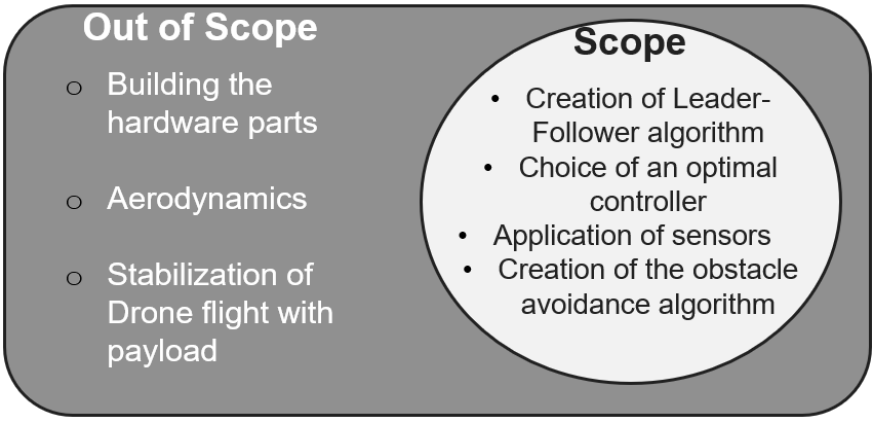


Figure 1.1: Defining the scope of research

Chapter 2 – Literature Review

2.1. Overview

The Leader-Follower system has been widely used in various applications, such as autonomous search and rescue, precision agriculture, and surveillance. In this system, a leader device leads a group of follower devices, which may include aerial or ground vehicles. To ensure that the follower devices maintain a safe distance from the leader and avoid obstacles, optimal controller design is essential. Over the years, researchers have proposed several techniques for optimal controller design, such as model-based control, adaptive control, and fuzzy logic control, among others. This literature review discusses the leader-follower system for unmanned vehicles, including studies on systems consisting of several UGVs and UAVs forming a formation when following the leader. Various linear and non-linear controllers are also reviewed, including PID controllers, backstepping control strategies, and the use of fuzzy controllers for UAVs. The potential applications of this system include delivery services, the study of hard-to-reach places and objects, military purposes, and rescue work.

2.2. Leader-follower algorithm and obstacle avoidance

The demand for the leader-follower system in unmanned vehicles is high, and there are numerous studies on this topic. Some of these studies involve formations consisting of both Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs), with one leadyк and several followers [1-3]. Gopakumar and Shihabudheen considered the system with 1 leader and 6 followers, three of which are quadrotors, while the other 4 vehicles are two-wheeled mobile robots (2WMRs) [4]. The aim of this study was to develop a cooperative control algorithm for reaching a consensus between the agents through communication. Other studies considered the loss of one of the agents (followers) and the ability of them to re-form into another formation during the mission [5-7]. Research has also been carried out to create a leader-follower system consisting of several 2WMRs that are able to bypass obstacles. In this case, the obstacle is detected by the leader, which re-computes the route, while the followers simply follow the leader [8]. In this study, the authors use the potential field method, the main idea of which is to determine the range of obstacles for different orientations of the sensors and to determine the polar density of the

obstacle. After that, the path with the lowest density is determined, which the vehicle must follow in order to avoid the obstacle and reach the target point [8]. Apart from that, a system was created where the UAV follows the landing station in the form of a UGV in order to be able to land on it at any time [9]. The objective of this article is to develop a controller for a leader-follower system involving a quadcopter and a two-wheeled mobile robot that can avoid obstacles. This system is unique in that the leader is a drone that faces no obstacles in the air, while the robot must follow and navigate through ground-level obstacles. Such a system has a broad range of potential applications, including delivery services, exploration of inaccessible areas and objects, military operations, and rescue missions. Since there are typically fewer obstacles in the air than on the ground, this type of system is necessary.

2.3. Optimal controller

Various linear and non-linear controllers can be used to control UAVs. Proportional-integral-derivative (PID) controllers and their variations are among the most researched. A PID controller designed by Pounds, Mahoni, and Corke [10] allowed a large, 4 kg quadcopter carrying a 1 kg payload to stabilize within ±1 degrees in less than 5 seconds. By combining the PID controller with the linear-quadratic regulator (LQR), Argentim *et al.* [11] were able to make it more effective than using PID and LQR separately. In a recent study, PID and LQR controllers were tested on the Quanser Qball-X4 by Liu, Pan, and Chang [12]. Compared to the PID controller in this study, LQR produced results that were more accurate and stable. Milhim *et al.* [13] developed gain scheduling-based PID fault-tolerant control to effectively handle the failure of one or two quadcopter motors.

Lee, Leok, and McClamroch [14-15], studied quadcopter geometric tracking control and were able to achieve a sufficient level of stability and control to perform high-angle maneuvers. This control strategy was used to operate a tiltrotor quadcopter UAV by Invernizzi and Loverza [16].

Ha *et al.* [17] studied UAV backstepping control strategies and found that an adaptive backstepping controller based on passivity performed significantly better than a non-adaptive controller in lifting an unknown load to a predetermined height. In another study, the three positional parameters (X, Y, and Z) of the drone were successfully controlled by a backstepping

controller integrated with sliding mode control (SMC) [18], with deviations from the reference staying within ±0.11 m and peak-to-peak values remaining within 0.2 m. In the study by Xiong and Zheng [19], the SMC controller itself demonstrated its effectiveness when compared to the PID. In a different study, a disturbance observer was added to the adaptive backstepping control scheme, which was based on fuzzy logic [20]. Compound Adaptive Fuzzy Control Scheme was introduced by Zhang *et al.* [21] and demonstrated success in comparison to LQR and backstepping-SMC controllers.

The use of fuzzy controllers for unmanned aerial vehicles (UAVs) is a current trend in control engineering. In two separate computer simulations conducted in 2012 [22] and 2018 [23], it was found that fuzzy controllers were more effective at stabilizing UAVs than simple proportional-derivative (PD) controllers. A similar simulation by Bodrumlu, Soylemez, and Mutlu [24] reached the same conclusion when comparing fuzzy controller to PID controller. Additionally, Ye, Yu, and Wang [25] conducted a simulation using an adaptive fuzzy event-triggered attitude tracking control method, which was found to be more efficient in terms of data transfer and controller update time. An aero-pendulum experiment [26] also demonstrated the improved performance of a fuzzy controller enhanced by particle swarm optimization. More recently, Rao *et al.* [27] used a fuzzy neural network controller to control the position of a UAV, although this method only showed minor improvements compared to PID. This research indicates the potential for using neural networks to fine-tune fuzzy controllers.

The purpose of the current study is to develop a tracking fuzzy controller and compare it to a proportional-integral-derivative (PID) controller in experimental conditions. The goal of this research is also to examine the experimental results of a system that uses three types of controllers under conditions where it is affected by disturbance from a fan.

2.4. Research gap analysis

According to the literature review presented above, the following research gaps can be defined:

- Study of the effectiveness of different control strategies for unmanned aerial vehicles (UAVs) in various scenarios, such as fuzzy controllers and PID controllers.

- Study of the efficiency of combining different types of vehicles in leader-follower systems for improved control and efficiency, particularly in scenarios that require obstacle avoidance and formation control.
- Study of the advanced control methods for unmanned aerial vehicles (UGVs) that can cope with the loss of one or more followers and re-form into a new formation.
- Study of the efficiency of application of neural networks to tune fuzzy controllers for better performance in controlling UAVs.

Chapter 3 – Methodology

3.1. Overview

The system consists of several hardware parts which communicate with each other by wired and wireless connections. The general scheme of communication is shown in Figure 3.1. The mission server reads the location of both vehicles from the cameras and constantly exchanges information with the UAV stabilizer and the UGV commander. This information contains the location of the vehicles and the coordinates of the point they need to reach. After starting the system, the robot's desired point is the X- and Y- coordinates of the drone, while the drone's desired points are the waypoints sent by the mission server. At the same time, the drone starts moving to the next waypoint only if the robot has approached it at a distance of less than 0.25 m. At this time, the ultrasonic sensors of the robot send information about the presence of obstacles to the commander, which turns on the obstacle avoidance algorithm if the sensors detect an object 0.3 m ahead of the robot. When the vehicles reach the last waypoint, the mission server sends a mission end message and the drone flies toward the center of the workspace.

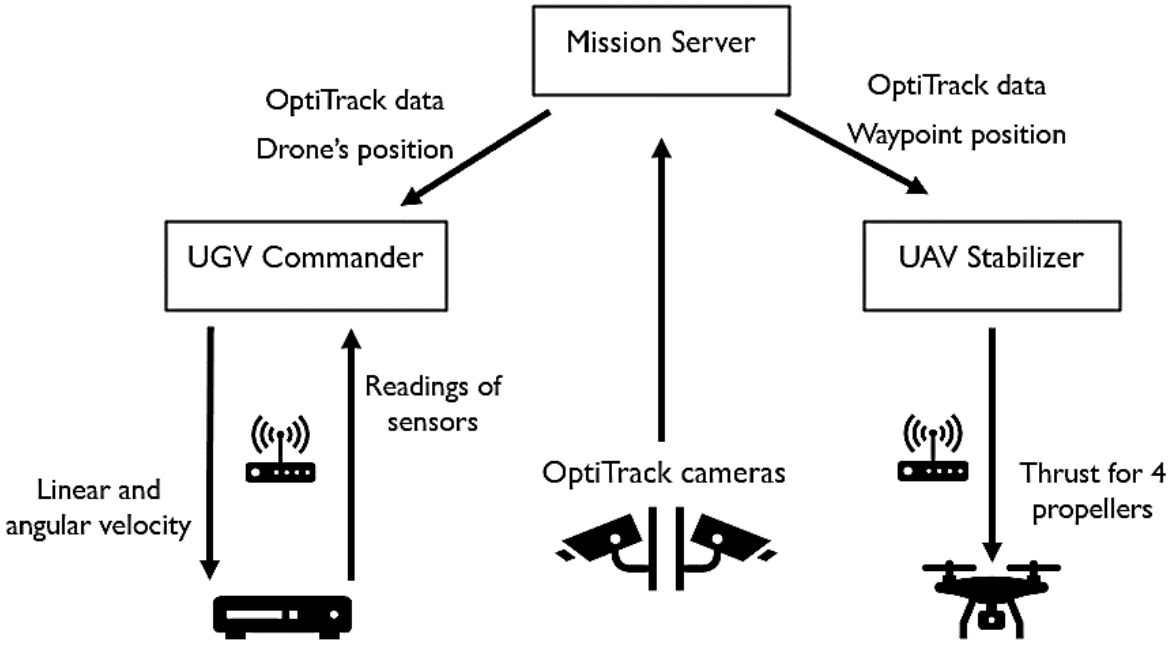


Figure 3.1: System communication scheme

3.2. Dynamic modeling

3.2.1. Dynamic Modelling of UAV

Using Newton's laws and the Euler-Lagrange method, the dynamic model of a quadcopter can be created [28]. The motion of the UAV can be described using a system consisting of six non-linear differential equations:

$$\begin{cases} \ddot{x} = (\sin\psi + cos\psi\ sin\theta\ cos\varphi)\frac{F_z}{m}, & \ddot{\phi} = \frac{F_\varphi}{J_{xx}} \\ \ddot{y} = (\sin\psi\ sin\theta\ cos\varphi - cos\psi\ sin\theta)\frac{F_z}{m}, & \ddot{\theta} = \frac{F_\theta}{J_{yy}} \\ \ddot{z} = -g + (cos\varphi\ sin\theta)\frac{F_z}{m}, & \ddot{\psi} = \frac{F_\psi}{J_{zz}} \end{cases} \quad (3.1)$$

where x, y, and z are the Cartesian coordinates of the quadcopter (Figure 3.2); φ, θ, ψ are the Euler angles (φ is the yaw, θ is the pitch, and ψ is the roll angle); Jxx, Jyy, and Jzz are diagonal elements of UAV's inertia tensor; m is the mass of the quadcopter; g is the acceleration due to gravity; F (F_z, F_φ, F_θ, F_ψ) – virtual control forces, which can be described by the motor control forces as:

$$\begin{aligned} F_z &= T_1 + T_2 + T_3 + T_4 \\ F_\varphi &= \frac{L_{Roll}}{2}(-T_1 - T_2 + T_3 + T_4) \\ F_\theta &= \frac{L_{Pitch}}{2}(T_1 - T_2 + T_3 - T_4) \\ F_\psi &= d(T_1 - T_2 - T_3 + T_4) \end{aligned} \quad (3.2)$$

where L_{Roll} and L_{Pitch} are the half-distances between 1st and 4th, and 2nd and 3rd motors (Figure 3.2), d - constant obtained experimentally, that depends on the relationship between thrust and yawing moment; T- thrust force generated by the rotation of propellers.

$$T_i = bu_i \quad (3.3)$$

where b is a thrust factor derived experimentally, and u – motor percentage commands (%PWM pulse from 0 to 1).

More detailed derivations are provided in Appendix I. Finally, u can be expressed as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & -\frac{1}{2bL_{roll}} & \frac{1}{2bL_{pitch}} & \frac{1}{4d} \\ \frac{1}{4b} & -\frac{1}{2bL_{roll}} & -\frac{1}{2bL_{pitch}} & \frac{1}{-4d} \\ \frac{1}{4b} & \frac{1}{2bL_{roll}} & \frac{1}{2bL_{pitch}} & -\frac{1}{4d} \\ \frac{1}{4b} & \frac{1}{2bL_{roll}} & -\frac{1}{2bL_{pitch}} & \frac{1}{4d} \end{bmatrix} \begin{bmatrix} F_z \\ F_\varphi \\ F_\theta \\ F_\psi \end{bmatrix} \quad (3.4)$$

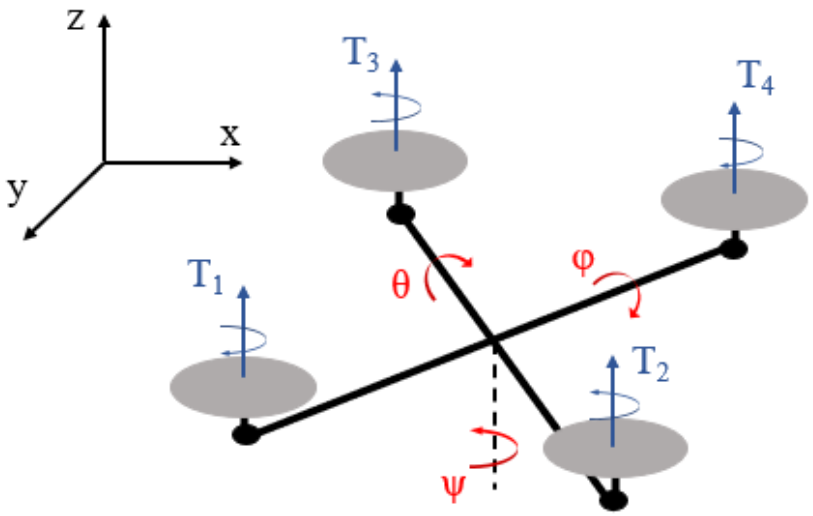


Figure 3.2: Schematic diagram for UAV

Table 3.1: Quadrotor parameter description

Symbol	Description	Value	Unit
J_{xx}	Roll Moment of Inertia	0.01	kg*m^2
J_{yy}	Pitch Moment of Inertia	0.082	kg*m^2
J_{zz}	Yaw Moment of Inertia	0.0148	kg*m^2
b	Maximum commanded thrust force	5.11	N
d	Normalized yaw torque constant	0.0487	Nm
M	Total mass of the quadcopter	1.121	kg
g	Gravitational constant	9.81	m/s^2
L_{Roll}	Roll motor-to-motor distance	0.2136	m
L_{Pitch}	Pitch motor-to-motor distance	0.1758	m

3.2.2. Dynamic Modelling of UGV

In this system UGV (QBot 2e) has only two wheels that are connected to the motor, that can rotate and act independently from each other. This allows QBot to drive and make a U-turn right on the spot around its axis. From Figure 3.3 it can be seen that point C is set in the center of the wheel axis and this point can be counted as the current position of the robot with (x, y) coordinates and orientation angle φ that was taken with respect to the x-axis.

Since the floor surface is a rough, dense rubber with an abrasive pattern, a lot of friction is established between the QBot 's wheels and the floor, allowing the robot to move and turn with negligible slip. It is also worth considering that the weight of the robot is not sufficient to deform the wheels. Considering all this, the non-linear kinematic equation for the Robot will be the following [29]:

$$\dot{C} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.5}$$

where v and ω are linear and angular velocities of QBot, respectively. Since Qbot's wheels are driven separately, we will have two equations for them, that are:

$$\begin{cases} v_R = v + l\omega \\ v_L = v - l\omega \end{cases} \tag{3.6}$$

where l is the distance from the center of the wheels' axis to each wheel. With simple manipulation, the equation for the angular velocity of each wheel can be calculated as shown below, where R corresponds to the radius of the wheels.

$$\begin{cases} \omega_R = \frac{v_R}{R} \\ \omega_L = \frac{v_L}{R} \end{cases} \tag{3.7}$$

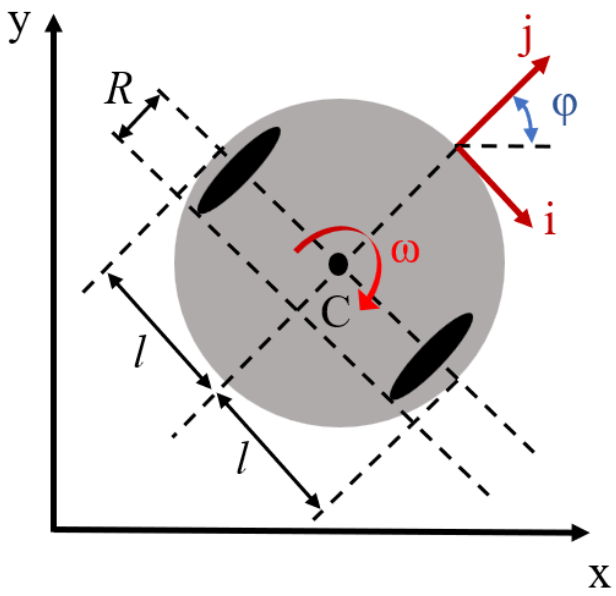


Figure 3.3: Schematic diagram for UGV

So that $\dot{C}$ can be defined by rotational speeds of wheels:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\varphi & \frac{R}{2}\cos\varphi \\ \frac{R}{2}\sin\varphi & \frac{R}{2}\sin\varphi \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{3.8}$$

3.3. Communication and localization

Communication between the vehicles and the computer is accomplished via a router and a wireless Wi-Fi connection, which assigns each computer, drone, and robot a unique IP address. Control and command of vehicles were supplied by configuring hardware settings in MATLAB Simulink and employing QUARC blocks that can transfer information to the carrier of the chosen IP address.

OptiTrack technology was used to determine the location of vehicles in space. Six cameras were positioned at the corners and sides of the workspace and wired to the ground station (PC). The cameras detect the location and orientation of an object by recording the placement of unique

markers attached to vehicles. 5 such marks were mounted to each vehicle for identification purposes according to a specific scheme provided by the manufacturer. This system is quite accurate and, according to the manufacturer, allows determining the location of an object with an accuracy of 2 mm [30].

3.4. Leader-follower algorithm

When the system is first started, communication between the drone and robot is established. The "Mission server" receives, analyses, and delivers data from OptiTrack to the leader and follower as soon as the connection is confirmed, and all three Simulink models are successfully launched. This data comprises information on the vehicles' location and orientation. If OptiTrack receives inaccurate data or fails to recognize the cars, the QDrone Commander model is instantly canceled. To perform the algorithm, QDrone must take off to a height of at least 0.5 m, which is accomplished with the joystick by giving the drone the "Arm" and "Takeoff" commands. As soon as both switches have been activated, QBot 2e starts its algorithm which is provided in Figure 3.4.

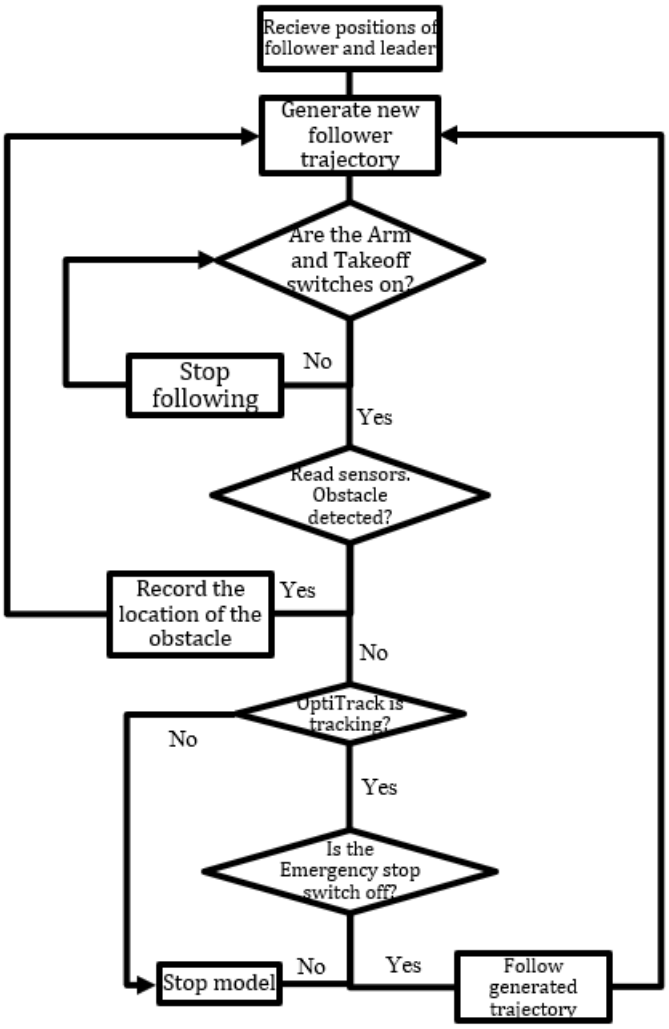


Figure 3.4: Algorithm for the follower

The follower trajectory is generated by the QBot 2e Commander as soon as it receives the leader and follower's coordinates. The QBot 2e receives these data in the form of angular and linear speeds. The 2-wheeled mobile robot will recalculate its trajectory if sensors detect an obstacle, using the obstacle avoidance algorithm outlined below. As seen in Figure 3.4, the algorithm keeps running until there is a tracking problem or the "Stop" button on the joystick is pressed. The robot will stop following the leader but will still calculate the trajectory if the "Arm" or "Take off/Landing" switches are off.

When the system is activated, the QDrone hovers over the takeoff point, and the QBot approaches it until they are within 0.25 m of each other. When the "Start/Stop" button in "Mission Server" is pressed, the algorithm begins, and the QDrone flies to predefined waypoints. As can be

seen from Figure 3.5, the “Mission Server” constantly compares the coordinates of the leader and follower. The algorithm communicates the next waypoint to the leader as soon as the follower reaches the leader. When the system reaches the final waypoint, the "Mission Server" communicates the coordinates of the work area's center, allowing the drone to land in the center.

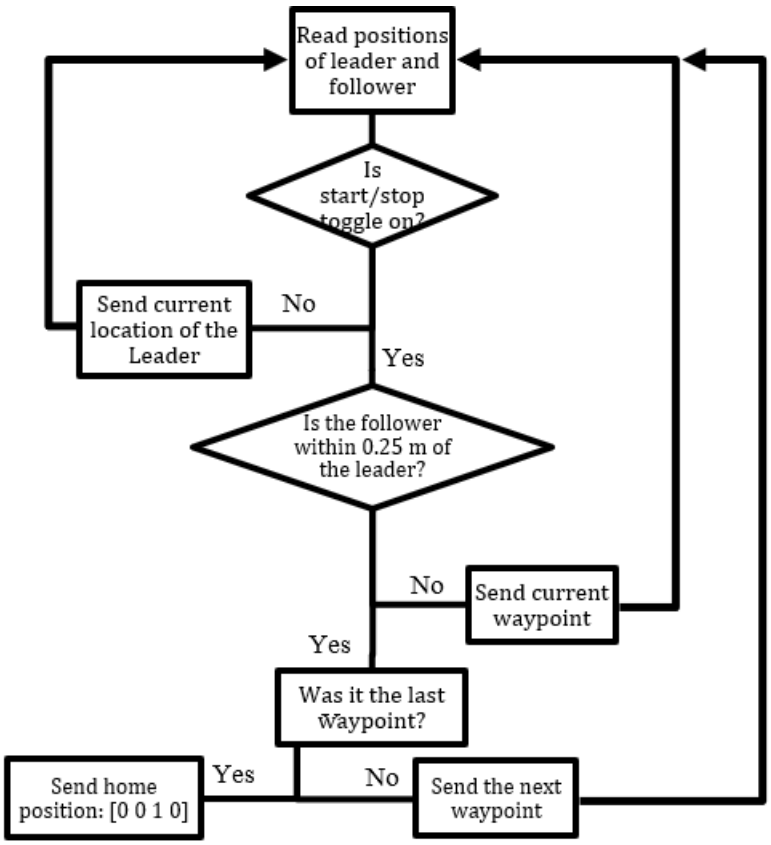


Figure 3.5: Mission server algorithm

In the case of a UAV, the algorithm checks the system for battery level, flying altitude, communication or localization issues, and receives directions to land or follow to a new waypoint. This algorithm is shown in Figure 3.6. In addition, the QDrone, as the leader, constantly sends its location to the follower.

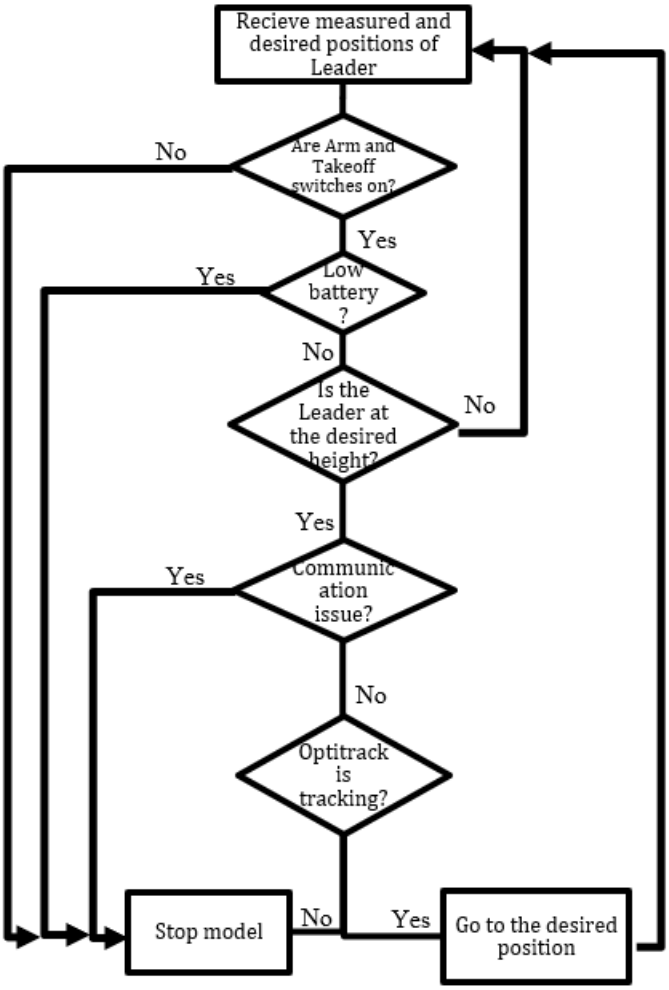


Figure 3.6: Algorithm for the leader

3.5. Obstacle avoidance algorithm

3.5.1. Attaching Sensors to Raspberry PI 3 Model b+

For obstacle detection, two ultrasonic sensors (model name: HC-SR04) were attached to the front of the robot. The robot itself has an onboard CPU, which is Raspberry pi 3 model b+ (Figure 3.7). In Figure 3.8 the scheme shows how an ultrasonic sensor was attached to Raspberry pi module. The basic idea is that if the sensor detects an obstacle within 30 cm, then the Raspberry pi sends a trigger to a free pin (in the diagram it is GPIO 16, which is connected by a yellow line to the computer in Figure 3.8). Subsequently, the program written in Simulink runs the obstacle avoidance algorithm.

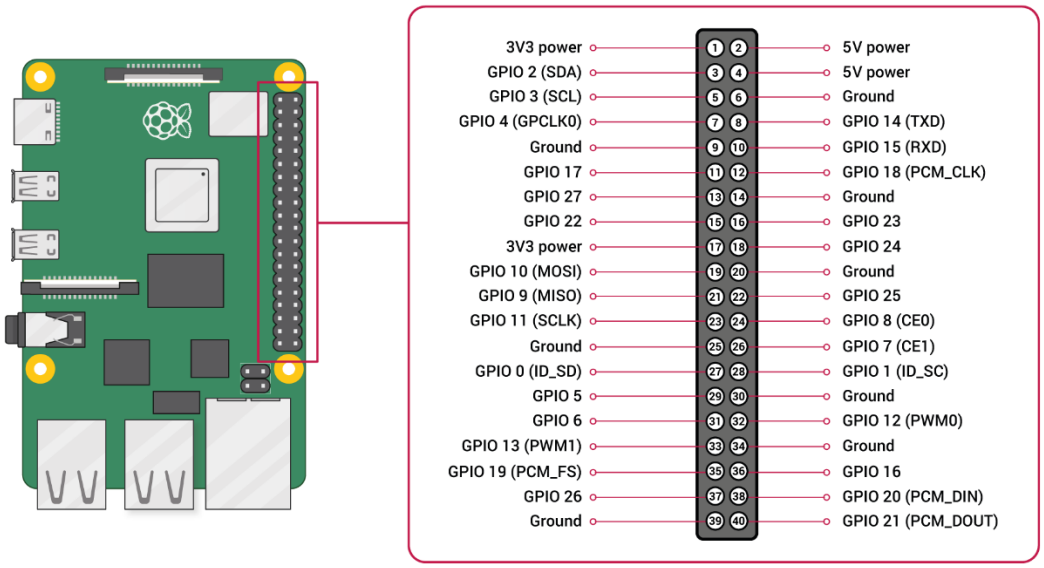


Figure 3.7: Raspberry PI 3 model b+ GPIO [31]

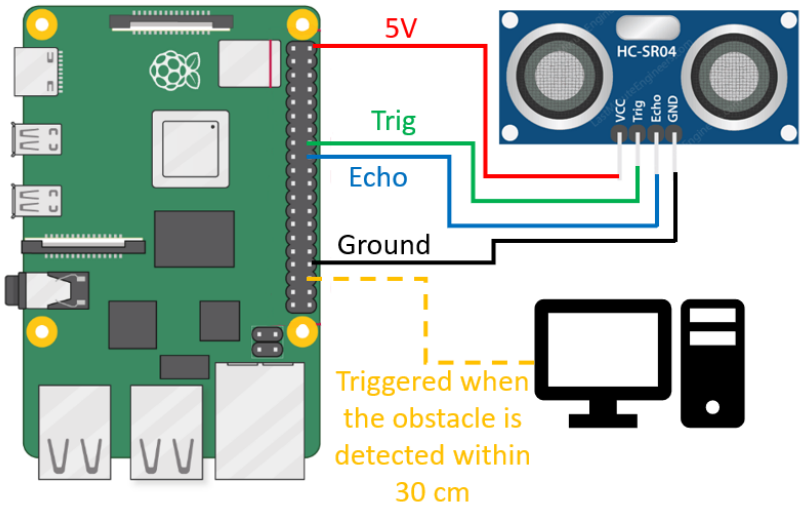


Figure 3.8: Connection of an ultrasonic sensor to raspberry Pi

3.5.2. Obstacle Avoidance Algorithm Creation

The obstacle avoidance algorithm is based on a simple method for solving mazes. The entire working space is evenly covered with virtual dots at a distance of 0.1 m from each other. The point where the robot is located is indicated by the number 1 and counting starts from it to the

point where the drone is located. The points at which the obstacle was spotted get a value of 0. That is, 4 neighboring points become 2, the points around these points become 3, and so on. As soon as the drone's location point acquires its value, say 7, a path is built from 1 to 7, where all numbers go in sequence, and accordingly, points with a value of 0 where obstacles are located cannot be part of this way (Figure 3.9). This algorithm is triggered and updated when the sensors find an obstacle at a distance closer than 0.3 m.

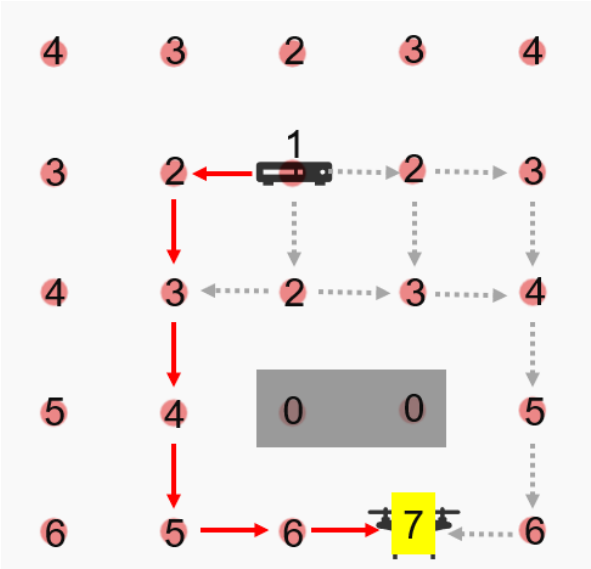


Figure 3.9: Obstacle avoidance algorithm concept

The algorithm generated a 60 by 40 matrix in which it recorded all the obstacles it encountered. Each index in this matrix denotes a field with a 10 cm square size. Despite the fact that the experimental field was 5 by 3 meters in size, it was decided to use a matrix that was 6 by 4 meters in size and place constraints along the edges of the virtual workspace. The matrix was extended to prevent the mistake of going beyond the matrix [32].

In the absence of detected obstacles, the QBot 2e moved directly to the coordinates of the QDrone. When a signal was detected, the QBot 2e placed a 30 cm forward obstacle in the corresponding matrix. Obstacles were set to the right of the center of the QBot 2e if a signal was registered from the right sensor, and to the left of the center if a signal was registered from the left sensor. If signals from both sensors were recorded, a continuous 60 cm obstacle appeared in the matrix 30 cm from the front of the QBot 2e. Due to the algorithm's characteristics, the detour path would pass near the barrier, prompting the inclusion of an area to the right and left of the obstacle

that was half the QBot 2e's diameter, or 17.5 cm. In practice, a barrier of only 6 indices or 60 cm was found to be effective, despite the algorithm theoretically requiring a 70 cm or 7 index wall. To navigate around obstacles, the robot added them to the matrix as soon as they were detected, continuing until it reached the QDrone.

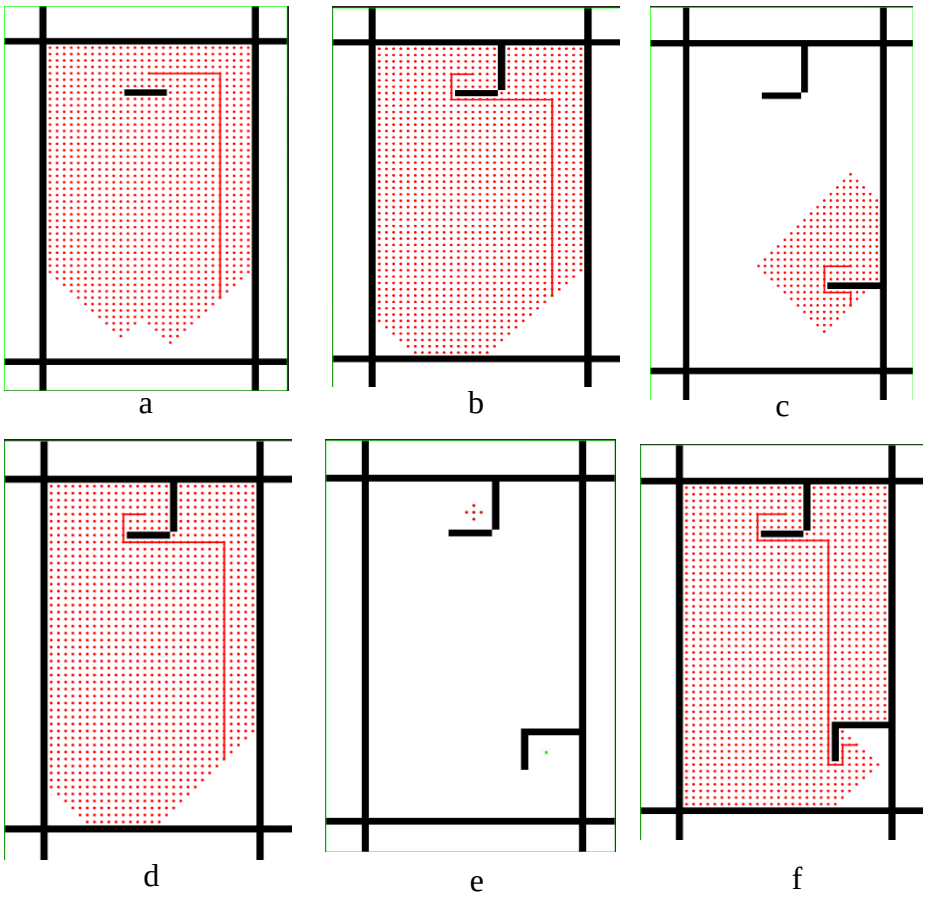


Figure 3.10: Path generation by the obstacle avoidance algorithm

In Figure 3.10 the diagram of the path created by the obstacle avoidance algorithm is provided. Figure 3.10e demonstrates all the obstacles that are detected by QBot 2e, and in Figure 3.10f the trajectory of QBot 2e that is followed in this experiment can be seen.

3.6. Fuzzy logic controller design

The PID controller was designed by using standard Simulink blocks, including 'Gain', 'Integrator', and 'Sum'. The input parameters are position error and velocity error, while the output parameters are controller commands for Thrust, Roll, Pitch, and Yaw:

$$Controller\ commands = K_p e + K_i \int e\, dt + K_d \dot{e} \qquad (3.9)$$

Figure 3.11a illustrates the control scheme applied in this project. It involves obtaining the locations and orientations, converting them to a horizon frame (HF), then sending them as input to a controller, and obtaining the thrust and torque values as output, which are then converted to pulse data and sent to the plant.

The Fuzzy Logic Controller blocks are attached to the Simulink model as it is shown in Figure 3.11. The controller is aimed to minimize the position error by controlling thrust forces, which affect the position error rate. In figure 4 FLC block can be viewed. Each has the inputs of the error and error rate of one of the components of position (x, y, z, and yaw). While the outputs of the controllers are the thrust and torques for roll, pitch, and yaw.

Figure 3.12 shows the operation procedure. When the "Mission Server" and "QDrone Stabilizer" models start up, OptiTrack begins reading the position of the quadcopter and sends it to the "Mission Server". Depending on whether the position control algorithm is activated, the "Mission Server" calculates the desired position and sends it along with the current position to the "QDrone Stabilizer." If the "Take off" button has been triggered, the drone takes off and goes to the desired position. If the position control algorithm is not activated, the drone hovers at a height of 0.8 m from the takeoff point. When the position control algorithm is activated, the "Mission Server" continuously changes the desired position using the time-dependent signal generator block, creating an oscillatory motion. QDrone continues to receive commands from the “Mission Server” until it receives the "Land" command, at which point it lands and completes the mission.

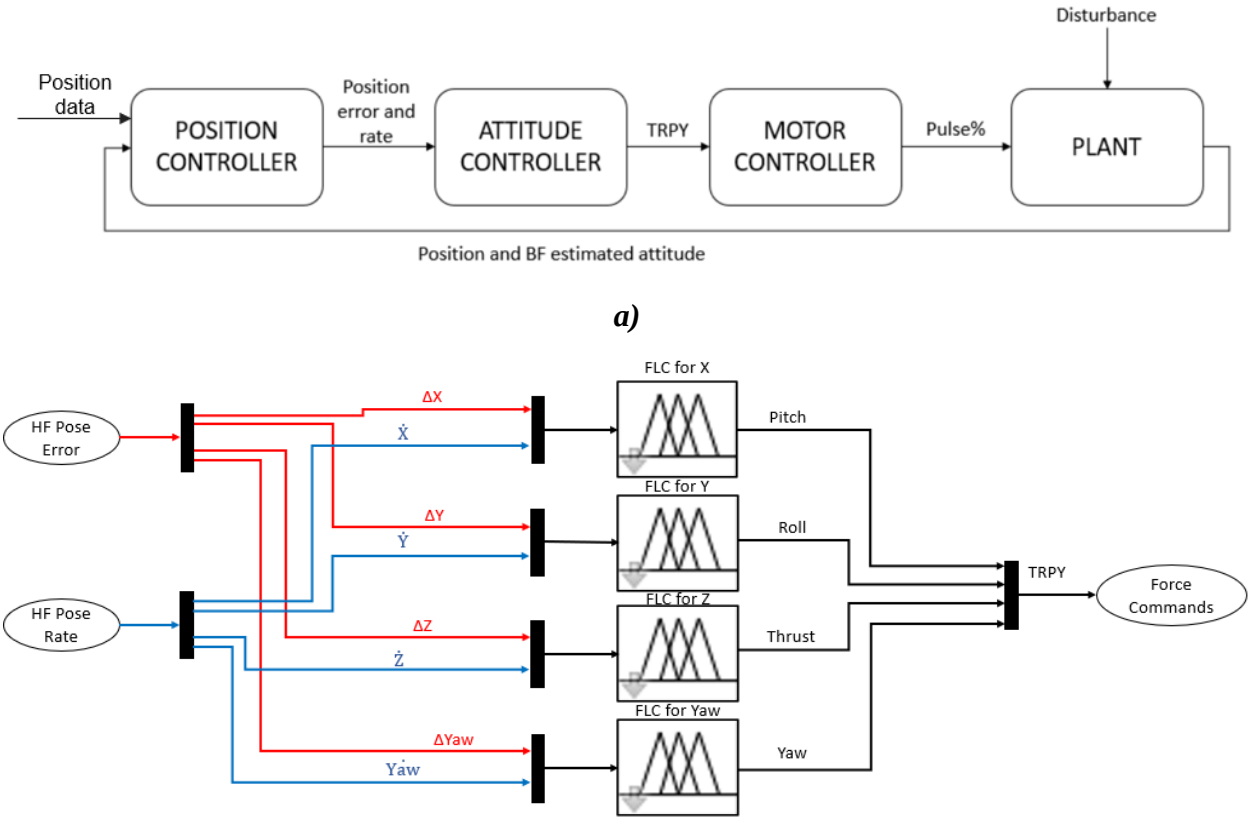


Figure 3.11: a) Block diagram of control scheme; b) Integration of FLC into Simulink model

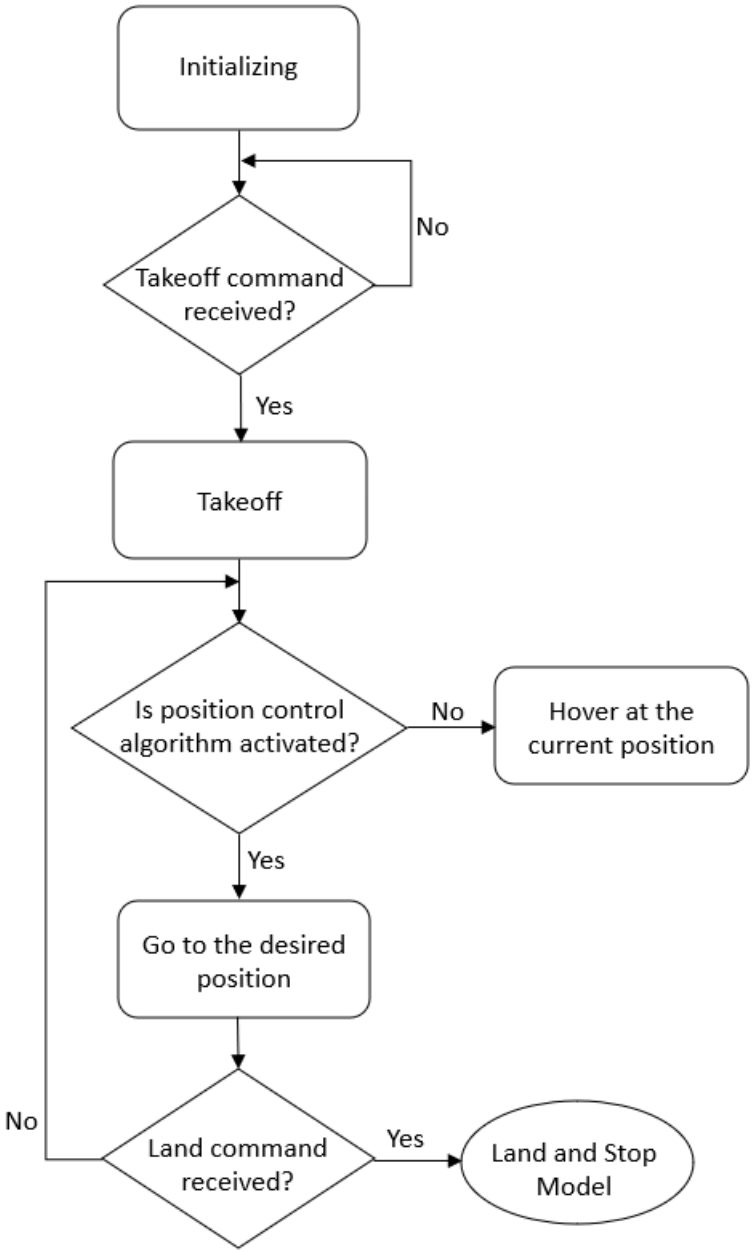


Figure 3.12: Flowchart for operation procedure for controller tests

The fuzzy logic controller is created by using 7 membership functions for each input, and in total 49 rules were created. The utilization of 81 fuzzy rules, each with 9 membership functions per input, was initially attempted, however, it was discovered that the execution of more than 2 controllers concurrently caused system deceleration. To address this issue, a fuzzy controller with 25 rules and 5 membership functions for each input was developed. Despite this modification, it was still not possible to operate all four controllers simultaneously. As a result, a decision was

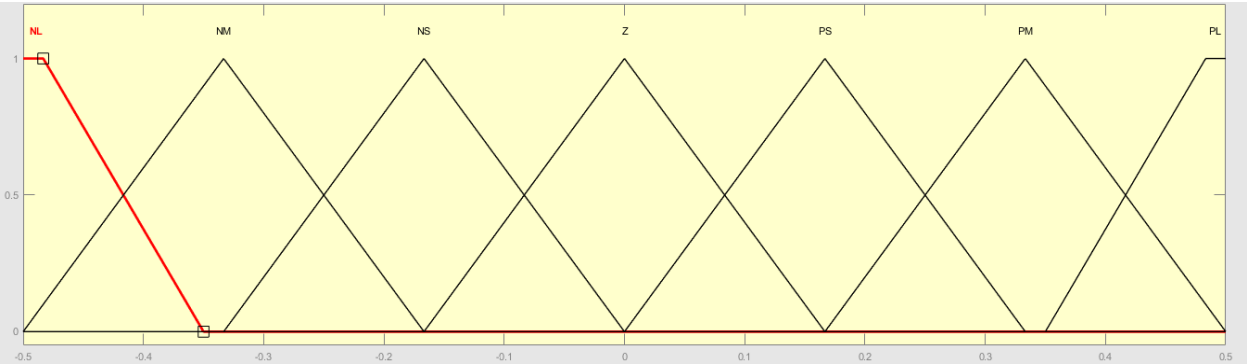
made to evaluate one controller at a time by employing 49 rules for the fuzzy controller. The outcome of this experiment indicated superior control as compared to the 25-rule-controller.

The fuzzy rules are represented in Table 3.2. Here E – is the position error, while $\dot{E}$ – is the position error rate. The membership functions are defined as NL – negative large, NM - negative medium, NS – negative small, Z – zero, PS – positive small, PM – positive medium, and PL – positive large. Similar rules table was applied by Raziyev *et al.* for controlling a lifting device [33].

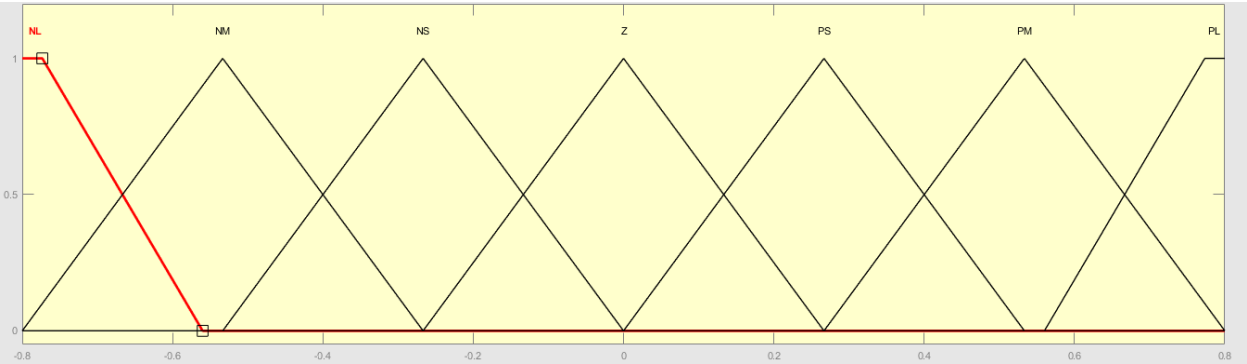
Table 3.2: Fuzzy rules table

$\dot{E}$ \ ΔE	NL	NM	NS	Z	PS	PM	PL
NL	Z	PS	PS	PM	PM	PL	PL
NM	NS	Z	PS	PS	PM	PL	PL
NS	NS	NS	Z	PS	PS	PM	PM
Z	NM	NS	NS	Z	PS	PS	PM
PS	NM	NM	NS	NS	Z	PS	PS
PM	NL	NL	NM	NS	NS	Z	PS
PL	NL	NL	NM	NM	NS	NS	Z

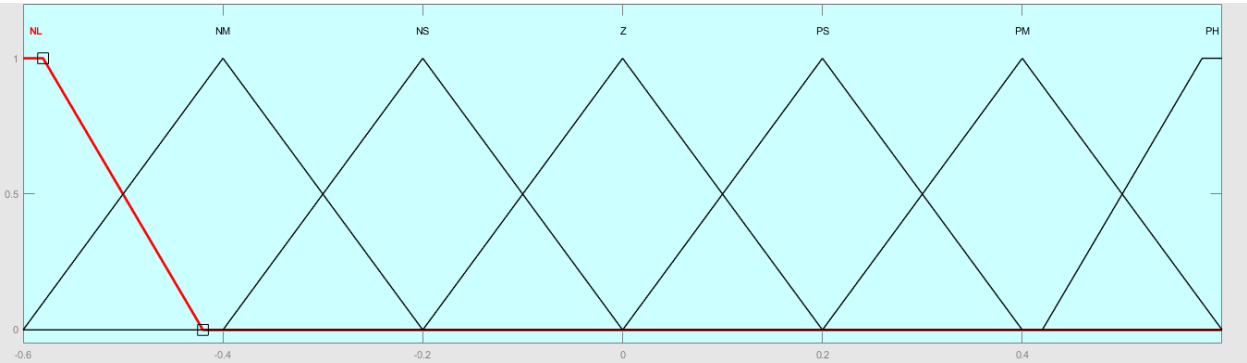
The fuzzy logic controller designer is shown in Figure 3.13. The tuning is made by changing the ranges of member functions and their shapes. For the final version of the X- and Y-position controllers the 1st input has a range from -0.5 to 0.5, the 2nd input ranges between -0.8 to 0.8, and the output from -0.6 to 0.6. Also, the rules were changed by experience. For the defuzzification, the centroid method was applied.



a) Input variable "Position error"



b) Input variable "Position rate"



c) Output variable "TRPY" (input "X" is converted to output "Pitch")

Figure 3.13: Membership functions design

For the experiments with disturbance, the ordinary fan was applied. The fan creates airflow which acts as a disturbance for the drone. This is done in order to test flight stability and compare the behaviors of two controllers (FLC and PID).

Chapter 4 – Results and Discussions: Leader-Follower System with Fuzzy controller

4.1. Experimental Setup

In order to establish a communication system between the leader and follower and to implement advanced algorithms, the QUANSER platform in conjunction with MATLAB & Simulink products were utilized. The experiment was conducted in a 3x5x3 meter enclosed workspace which provided sufficient height for safe UAV flight and enough space to position obstacles. The experimental setup included six OptiTrack cameras, installed at four corners and one in the middle of the long side of the workspace, to track the coordinates of the vehicles. Additionally, a PC with a router served as the ground station. The workspace was covered with a safety net and a 2 cm thick rubber flooring (as shown in Figure 4.1) was used to prevent fatal crashes of the UAV(s) and ensure operator safety.

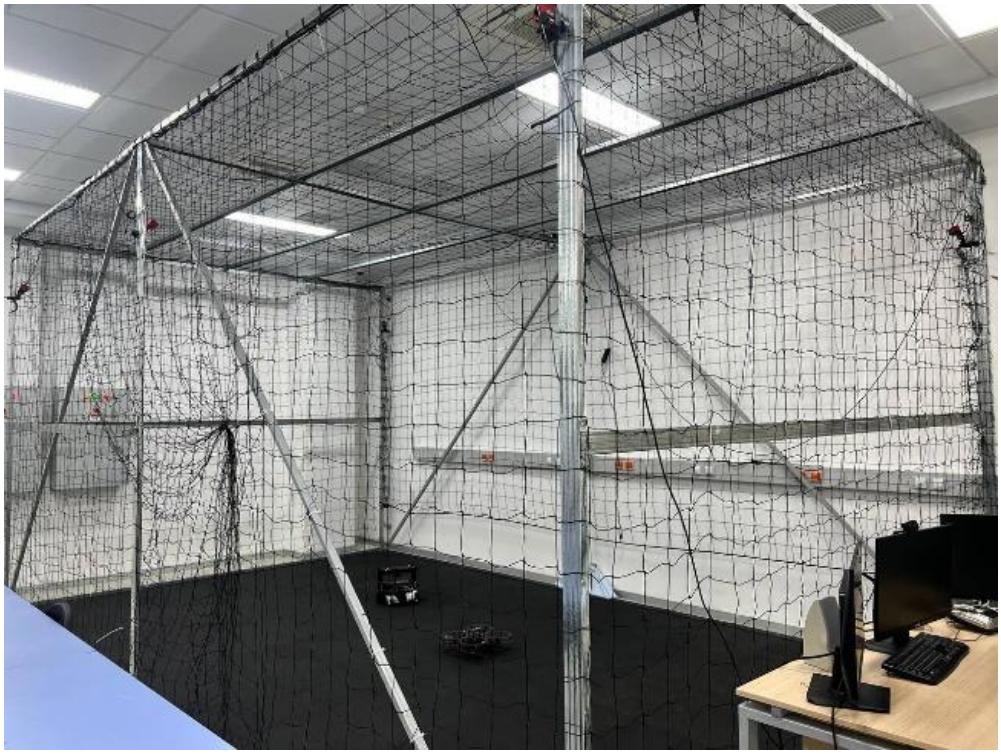


Figure 4.1: Workspace covered by safety net

The PC and router are part of the ground station. The UAV(s) and UGV(s)' onboard modules send data to the router, which then sends it to the PC. The PC has been configured to run

QUARC software, which only supports the MATLAB & Simulink integrated development environment. As a result, QUARC software receives all of the data from the router and the OptiTrack cameras. The Simulink-based Quanser toolbox provides access to this data. Simulink contains all of the control system algorithms, allowing programmers to quickly prototype controls and conduct real-time HIL testing.

OptiTrack cameras were employed in the experiment and were mounted on metallic bars at a height of 2.5 meters from the ground to ensure optimal coverage of the entire workspace, as depicted in Figure 4.2. The markers placed on the bodies of the UAV(s) and UGV(s) accurately determine the position and orientation data. The collected data is then sent directly to the computer where it can be utilized by Simulink.

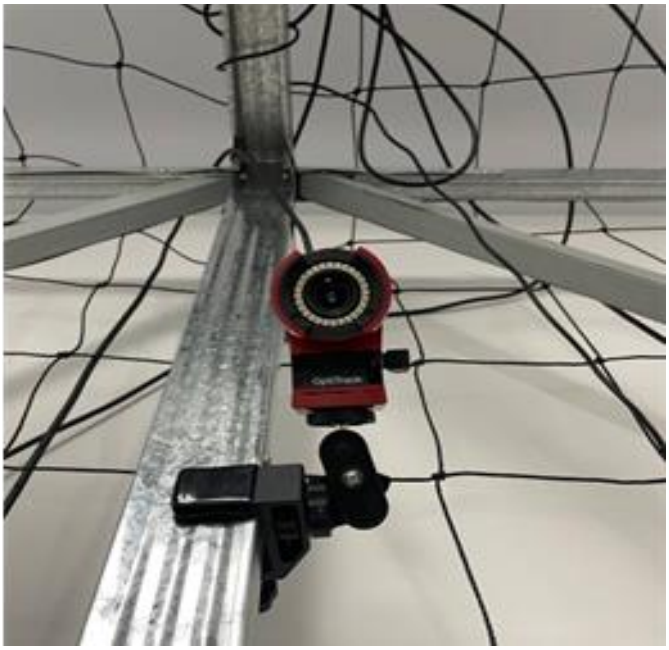


Figure 4.2: OptiTrack Camera

The QDrone from QUANSER (Figure 4.3) is an unmanned aerial vehicle with an Intel Aero Computing Board already installed for connecting to a ground station. In order for the OptiTrack to read the precise XYZ coordinates, pitch, yaw, and roll data, QDrone has five markers on its body. With an onboard raspberry pi 3 model b+ module for Wi-Fi communication and sensor data processing, QBot 2e serves as an unmanned ground vehicle. Additionally, QBot 2e has five markers on its body that are used to measure the object's coordinates, pitch, roll, and yaw. Two

ultrasonic sensors, model number HC-SR04, were mounted on the front of the QBot 2e to allow it to navigate obstacles (Figure 4.4).

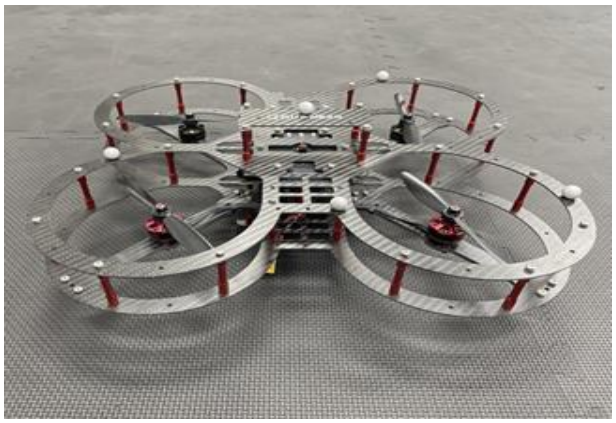


Figure 4.3: QUANSER's QDrone

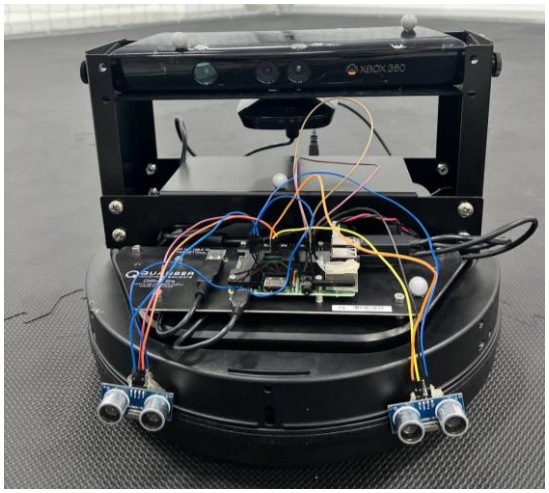


Figure 4.4: QUANSER's QBot 2e and the attached HC-SR04 sensors

4.2. Experimental Results: Obstacle Avoidance Algorithm

In order to test the algorithms for effectiveness in the different situations, 5 formations of obstacles were used. Figure 4.5 shows the ideal trajectory of the algorithm and the obstacles that were placed.

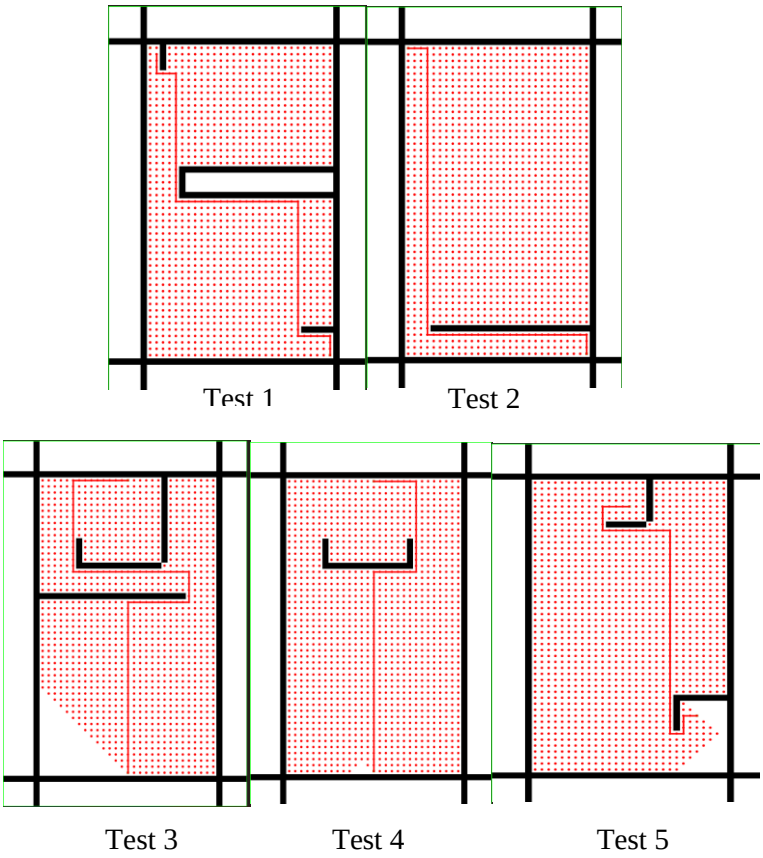


Figure 4.5:The path generated by obstacle avoidance algorithm which must be followed by QBot 2e in ideal case

Table 4.1: The obstacle avoidance experiment results

Test No.	Number of collisions with obstacles	Number of attempts to successfully achieve desired position	Successfully arrived to desired position
1	2	1	True
2	2	3	True
3	1	-	False
4	1	1	True
5	1	1	True

The first and fifth tests, as shown in Table 4.1, were successful because the obstacles were positioned so that the QBot 2e was moving slowly when the sensors detected the obstacle. This allowed for effective obstacle avoidance, and after colliding with obstacles, the sensors in front of the robot were not damaged.

There weren't many barriers in the second and fourth tests. The fourth case, however, was simpler to win than the second. As a result of the obstacle being closer to QBot 2e's starting point in the fourth case than it is in case 2, its speed when determining the presence of an obstacle is lower than in case 2. In the second instance, QBot 2e hit its top speed as it approached QDrone and ran into the wall without having a chance to slow down. The sensors' wires were severed as a result of the collision, rendering them inoperable. The third attempt saw the QBot 2e successfully return to the QDrone after the sensor's wires were reattached.

In the third test, despite six attempts, the robot was unable to reach QDrone. The algorithms believed there was no way to return to QDrone because, during deceleration, QBot 2e was able to fix a 5-decimeter thick wall directly in front of it, blocking the existing 4-decimeter wide corridor.

From the tested cases, several limitations of the current algorithm and system can be identified [32]:

1. In situations where the QBot 2e gains high momentum, it may create a wall in front of itself that is several decimeters thick, resulting in failure to complete the task as seen in case 3.

2. The path generated by the algorithm may not always be the most optimal due to the obstacles being recognized only right in front of the sensors. As a result, the QBot 2e may only avoid obstacles and follow QDrone instead of taking the shortest path to the desired point.

3. The algorithm generates a path that is too close to obstacles, which can lead to unnecessary collisions and time management issues.

To carry out a comprehensive experiment with waypoints and drone movement, a simpler set of obstacles was chosen, consisting of two obstacles placed within the workspace (see Figure 4.6). The first obstacle (OB1) was positioned directly in front of the QBot 2e at the start, while the second obstacle was placed between the first and second waypoints to ensure that the follower avoids it while passing through the predetermined trajectory.

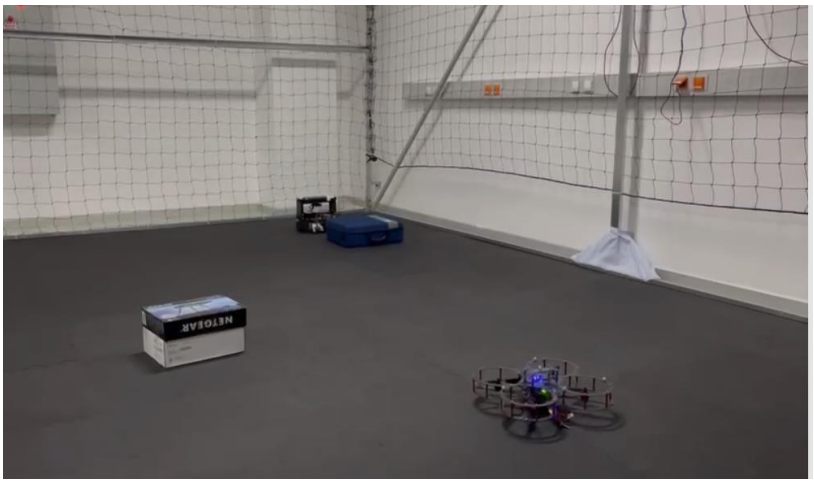


Figure 4.6: Obstacle avoidance algorithm turns on from the beginning

Figure 4.7: QBot 2e avoids the first obstacle

The trajectories of the leader and follower are presented in Figure 4.8, which shows that both vehicles were able to reach their assigned waypoints. This indicates that the leader did not proceed to the next waypoint until the follower was in close proximity to it.

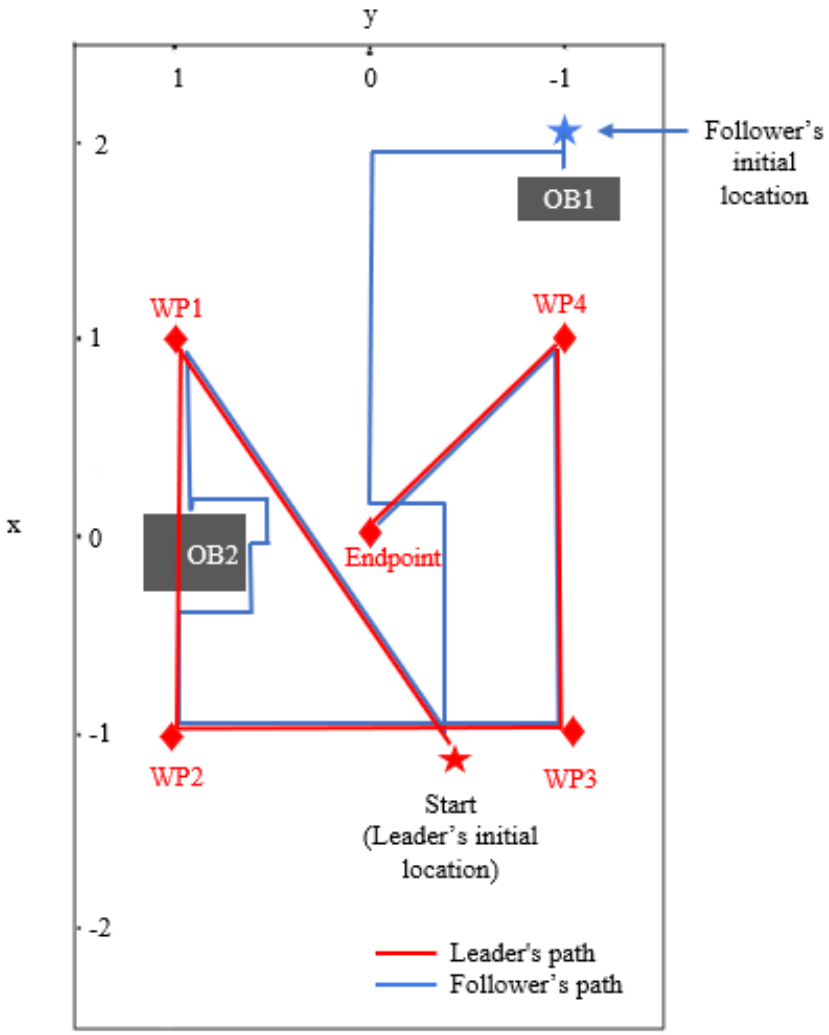


Figure 4.8: Leader-follower algorithm with obstacle avoidance test (WP – waypoint, OB – obstacle)

Additionally, the figure illustrates how the follower successfully avoided obstacles, which is further supported by Figure 4.7 and Figure **4.10**. However, during the avoidance of obstacle No. 2 (OB2), the QBot 2e collided with it (Figure 4.9). This was due to the fact that the QBot 2e had accelerated significantly from the previous stop and did not have enough time to decelerate before reaching the obstacle. To address this issue, it was proposed to program the sensors to detect obstacles at both long and short distances. However, this was not feasible as the Raspberry Pi was not capable of processing such a large amount of information.

After the UGV collided with an obstacle, it bypassed it, but when it did the check by turning back towards the obstacle, the sensors did not detect the obstacle. Although the obstacle slightly blocked the path of the UGV (it can be seen in Figure 4.8), this led to a second collision. This was

due to the limited computing power of the Raspberry Pi, which prevented the use of three sensors. As a result, it was decided to install two sensors on either side of the QBot 2e, but with both sensors facing forward, creating blind spots on either side of the vehicle.

Asad *et al.* designed a hurdle avoidance controller for QBot 2e, using an Arduino Uno microcontroller [34]. This study also used 2 ultrasonic sensors for obstacle detection. According to the results of the study, it can be concluded that the system built by Asad *et al.* more smoothly bypasses obstacles. This is because, in this study, the sensors read the distance of the obstacle and subsequently classify them as far, medium, and near, which allows for a smoother response to triggers. Moreover, it is important to note that in this study, the onboard microcontroller did not communicate with the earth station, which did not create delays. The same cannot be implemented for this system at this stage, because in addition to the task of avoiding obstacles, the goal of the study is also following the leader, which complicates the system and requires more memory.

Figure 4.9: QBot 2e collides with the second obstacle

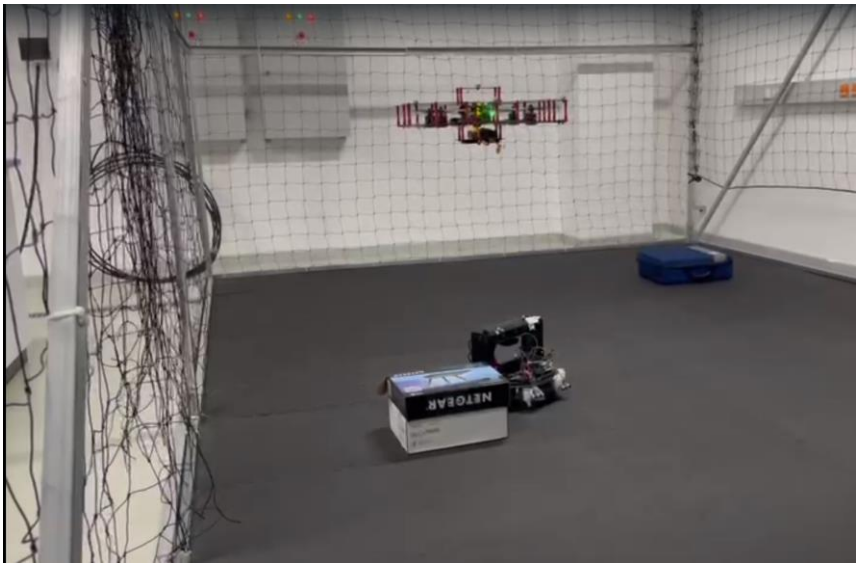


Figure 4.10: QBot 2e avoids the second obstacle

4.3. Experimental Results: Implementation of Fuzzy Logic Controller

Table 4.3 shows the experimental results and demonstrates that tests were conducted for three different controller types under three different disturbance scenarios:

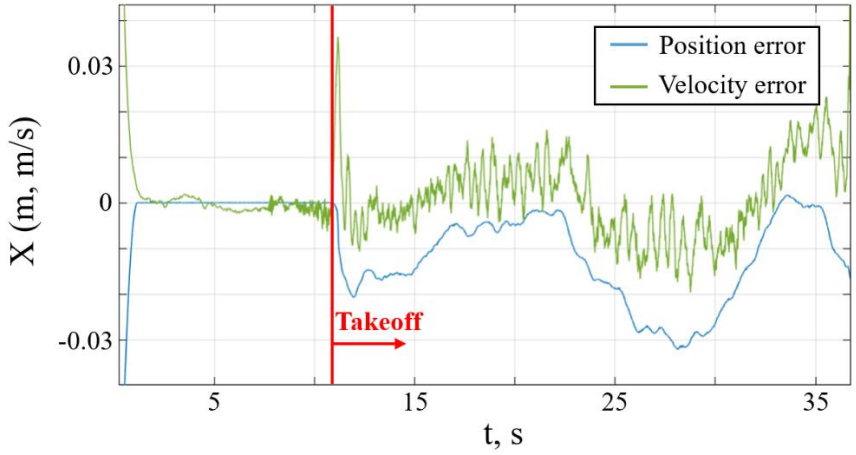
- No disturbance
- Disturbance is produced at a 0-degree angle to the positive X or Y direction
- Disturbance is produced at a 45-degree angle to the positive X and Y direction

The disturbance was created by a simple pedestal fan manufactured by Polaris Inc. The used fan has propeller diameter of 400 mm, and rotation speed of 1350 rpm when the maximum speed mode is turned on. The specifications of the fan are provided in Table 4.2. In all experiments with disturbance the fan was placed 1 m away from the QDrone takeoff point, which is the center of the XY-coordinate system (0, 0). For the case when the disturbance is applied at 0◦ angle, the fan was located at (-1, 0) and (0, -1) on XY axes towards positive X and positive Y directions. While when the disturbance is applied at 45◦ angle, the fan is located at the coordinates (- 0.71, -0.71).

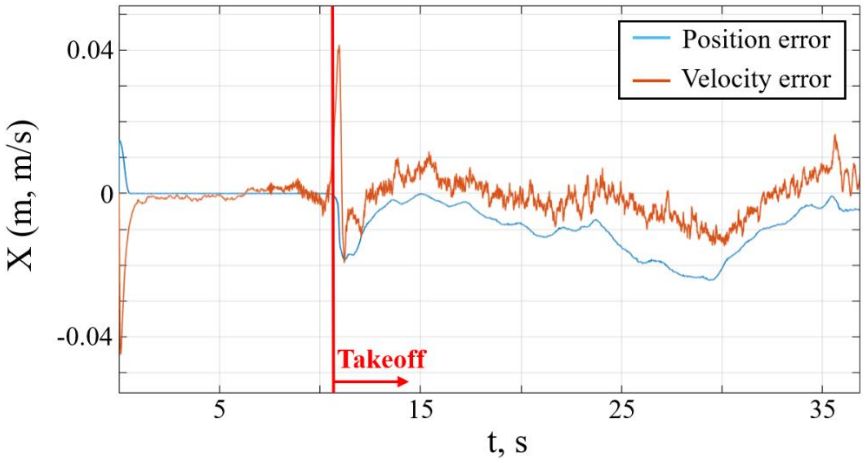
Table 4.2. Technical Specification of Pedestal Fan

Parameter	Details
Model name	Polaris PSF 40 Das
Type	Floor standing fan

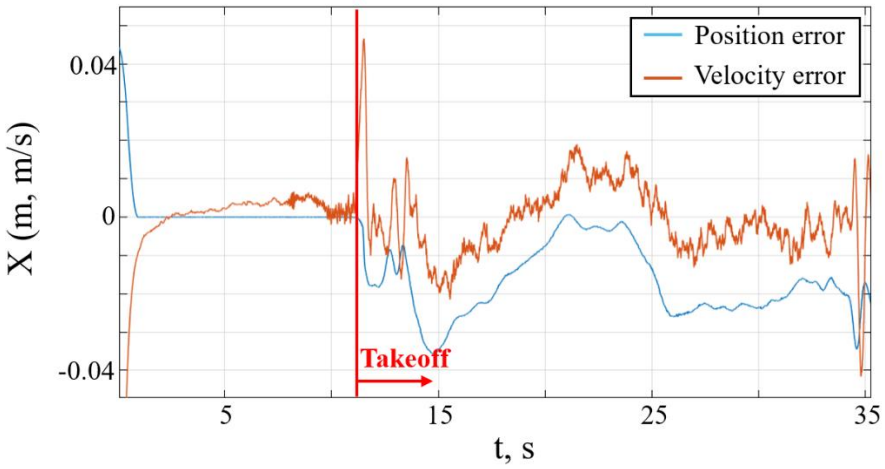
Manufacturer	Polaris Inc.
Power	55 W
Propellers diameter	400 mm
RPM	1350 rpm



a) PID

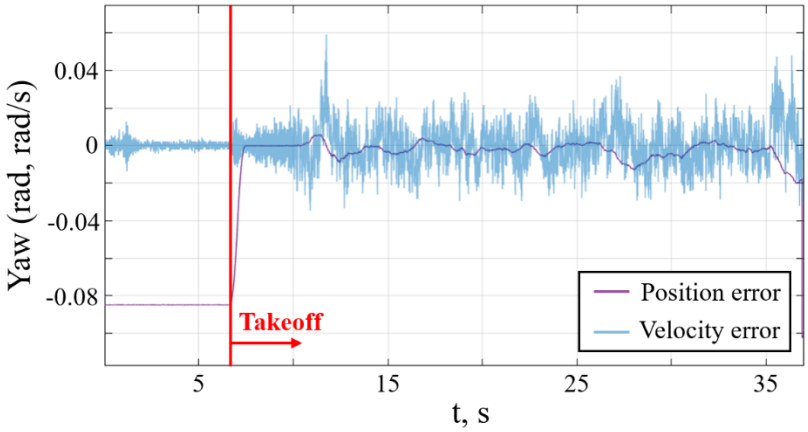


b) Fuzzy

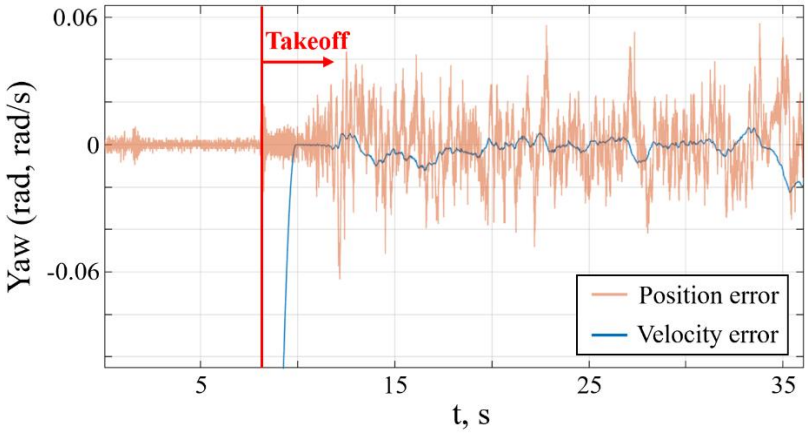


c) Fuzzy + prefilter

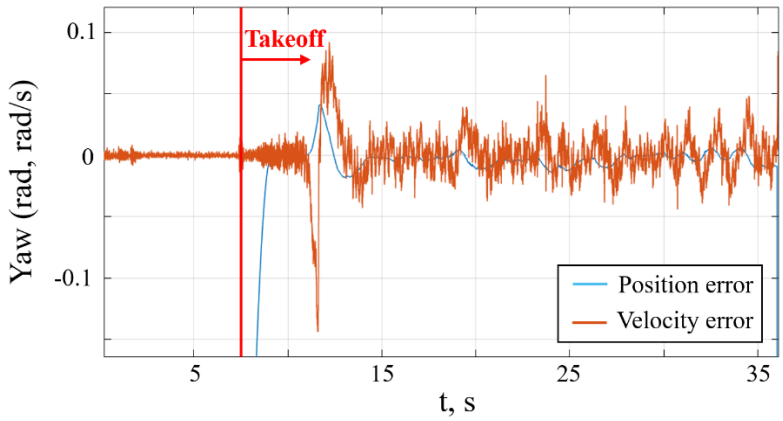
Figure 4.11: Position error and velocity error (X) vs time for hover test with no disturbance applied



a) PID



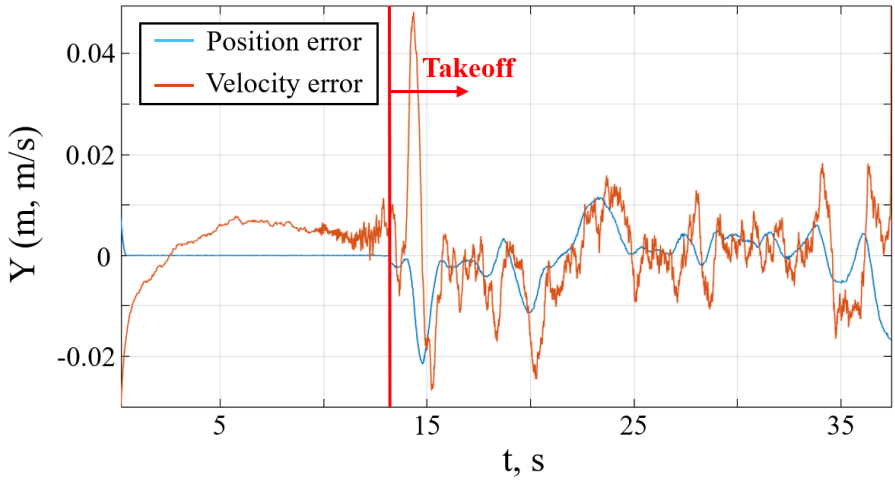
b) Fuzzy



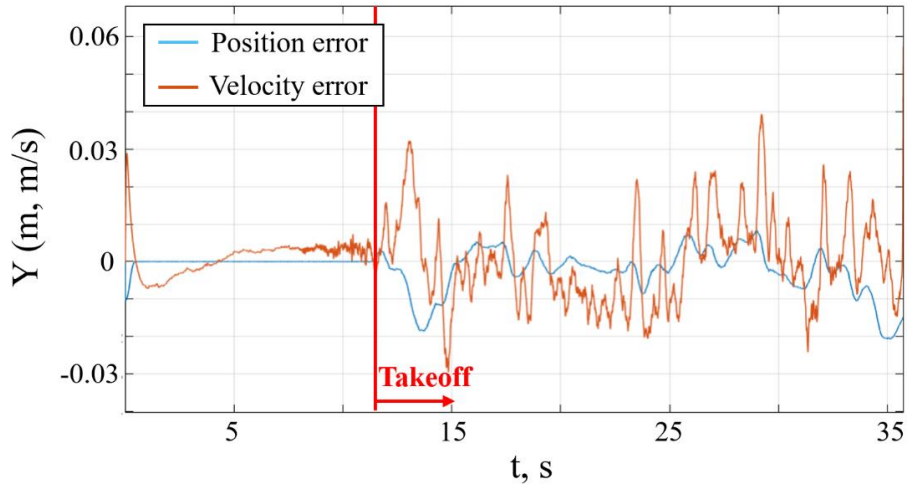
c) Fuzzy + prefilter

Figure 4.12: Position error and velocity error (Yaw) vs time for hover test with no disturbance applied

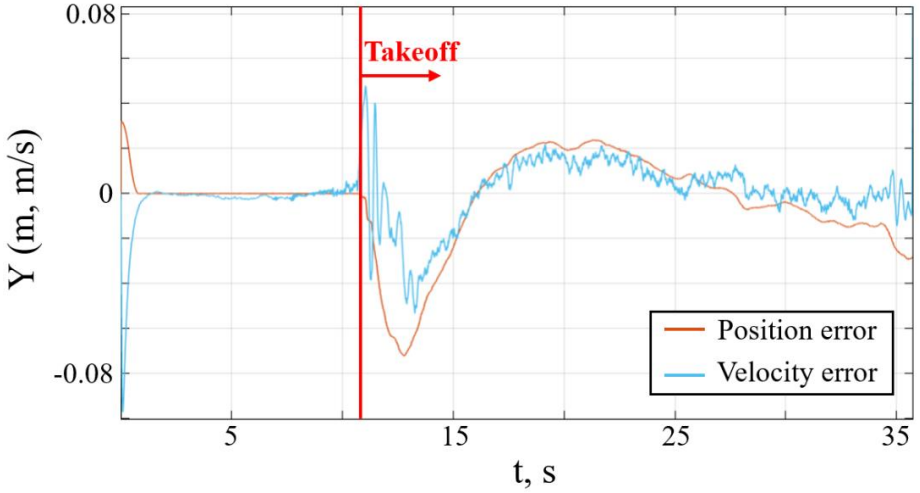
The results presented in Table 4.3 pertain to hovering tests, in which the drone was required to maintain its position over the takeoff point. The tests with disturbance applied at a 45- degree angle were not conducted for Z and Yaw position control, since the fan position can be changed in X and Y directions only. For this reason, Z and Yaw results with disturbance applied at a 45-degree angle are the same as for 0-degree results.



a) PID

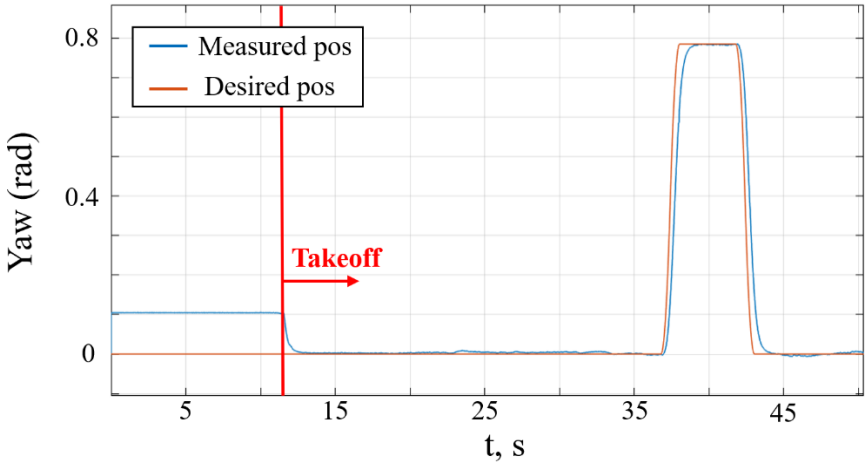


b) Fuzzy

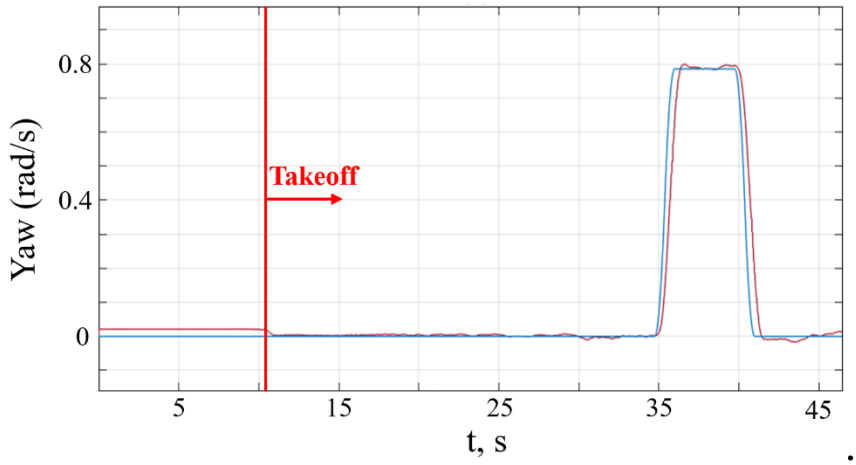


c) Fuzzy + prefilter

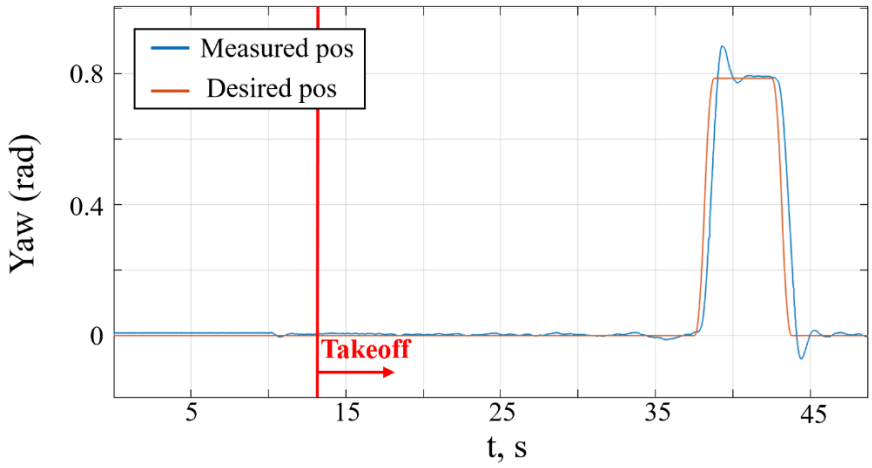
Figure 4.13: Position error and velocity error (Y) vs time for hover test with disturbance applied at 45◦ angle



a) PID



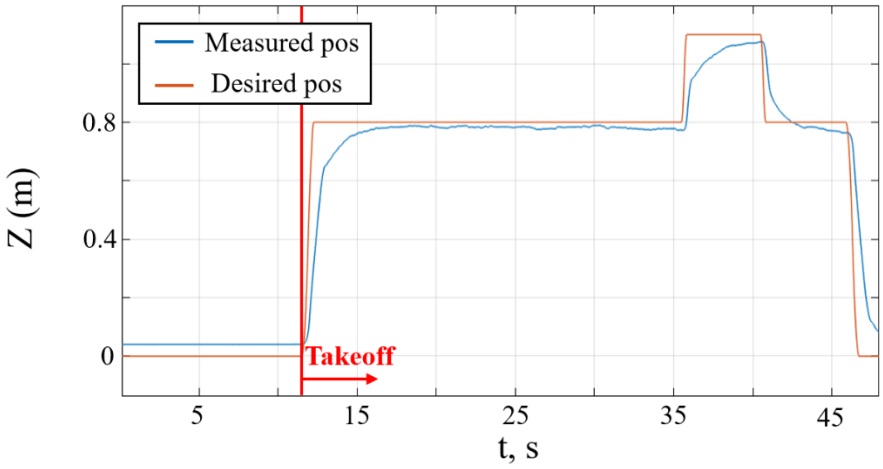
b) Fuzzy



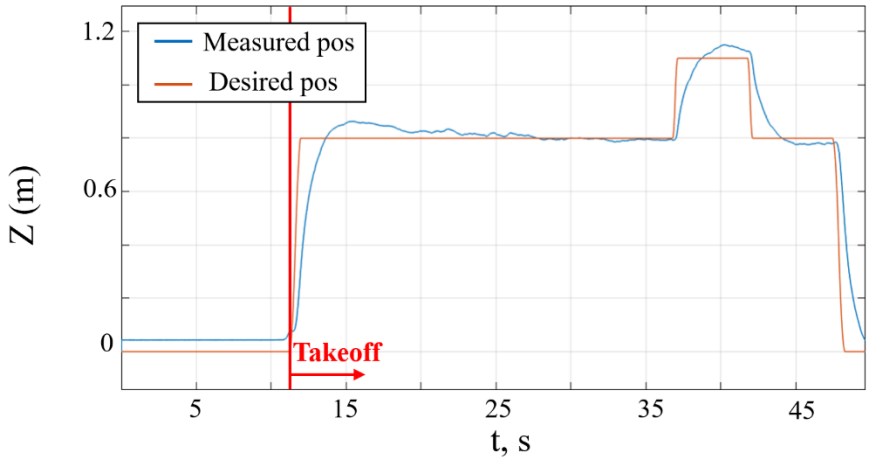
c) Fuzzy + prefilter

Figure 4.14: Yaw vs time graphs for position control test with no disturbance applied

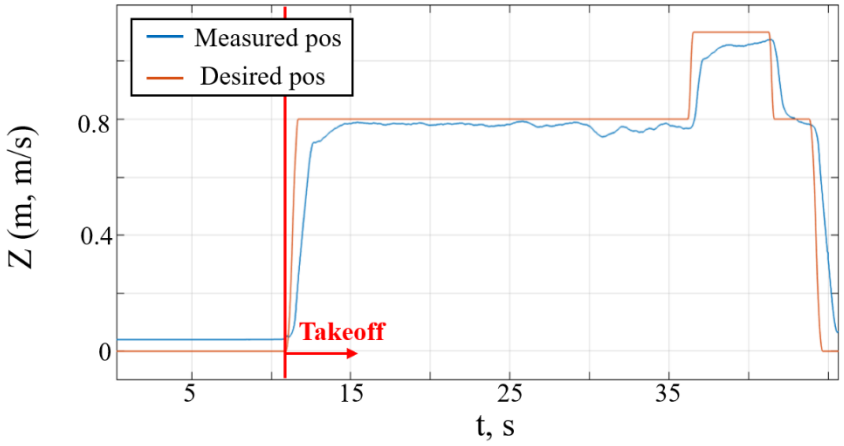
The most stable behavior for x, y, and yaw position control was achieved using an FLC in the tests without disturbance, with 0-degree disturbance, and with 45-degree disturbance, as determined by root mean square error (RMSE) and peak-to-peak (P2P) data analysis. Overall, it was noticed that for majority of cases the PID controller results are the best when the disturbance was not applied, and the worst when the disturbance is applied at a 0-degree angle. The FLC with a prefilter showed the most stable behavior in both experiments conducted for z-position control and exhibited the most accurate reference shape repeat. While for the other three components of position control (X, Y, and Yaw), the fuzzy controller without a prefilter shown the most satisfactory results.



a) PID



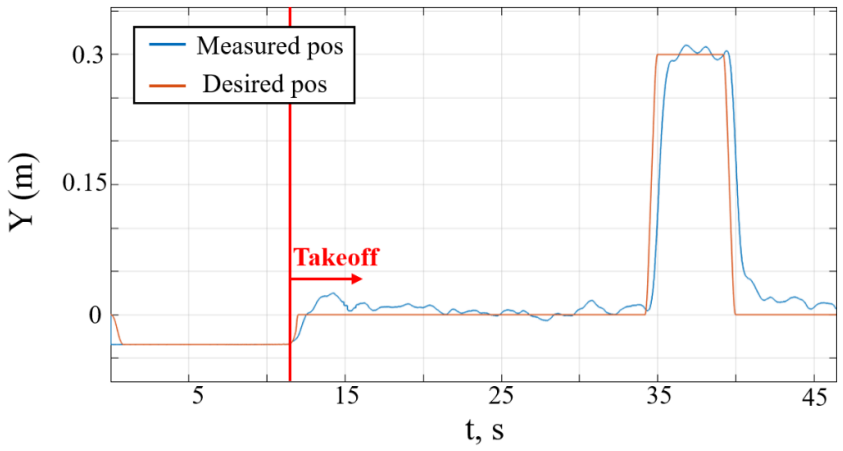
b) Fuzzy



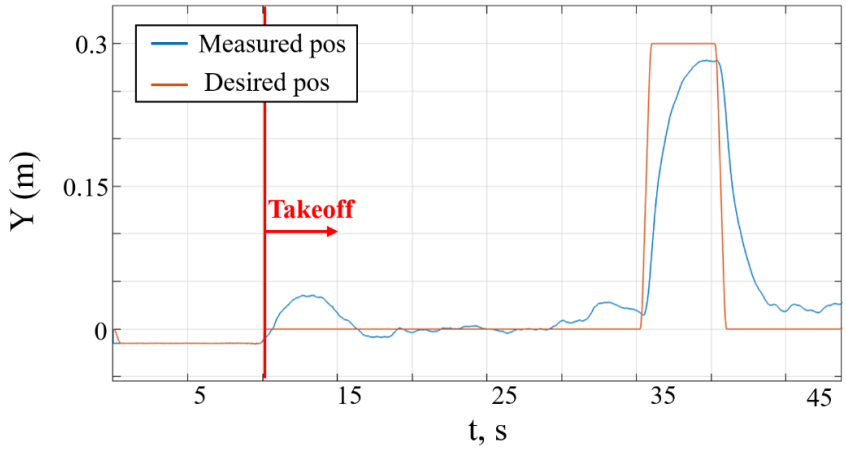
c) Fuzzy + prefilter

Figure 4.15: Z vs time graphs for position control test with disturbance applied at 0◦ angle

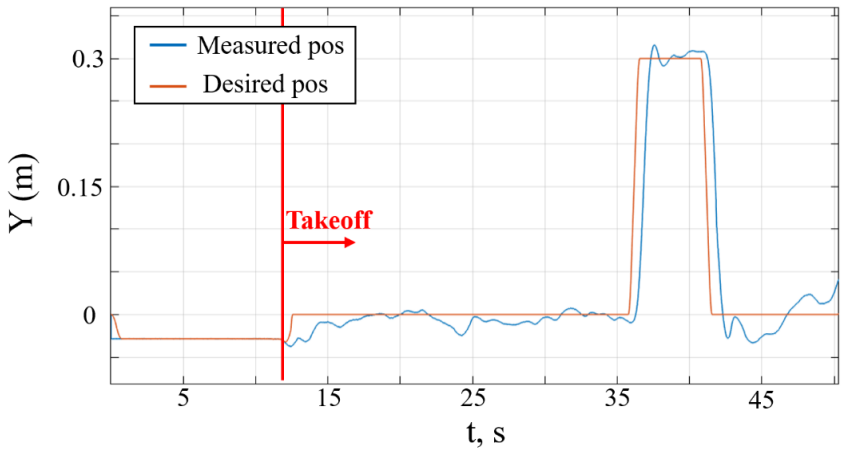
The PID controller performed similarly to the fuzzy controller in the hover tests conducted to control the X-position. This is evident from Figure 4.11. When comparing X and Y results for the hover test with no disturbance applied, it is observed that the PID controller outperformed the fuzzy+prefilter controller when the disturbance was not applied.



a) PID



b) Fuzzy



c) Fuzzy + prefilter

Figure 4.16: Y vs time graphs for position control test with disturbance applied at 45◦ angle

The results of the hover test for Yaw shown in Figure 4.12 demonstrated that the Fuzzy controller was slightly better than the PID controller in maintaining the hover position. However, the fuzzy controller with prefilter controller graph depicted an overshoot at the beginning. Similar behavior was also observed for the position control tests, as shown in Figure 4.14. In fact, the overshoot was observed for all fuzzy+prefilter controllers for Yaw (see Appendix A. Fig. 24).

In the hover test for Y position control, as depicted in Figure 4.13, the PID controller performed worse than the other two controllers. However, the table shows that the RMSE error for the PID controller is less than that for the fuzzy controller with prefilter. This was due to the fluctuations that appeared in the test with fuzzy with prefilter. Nevertheless, the peak value was significantly less, and the Fuzzy controller demonstrated the best result. When considering the position control test for Y in Figure 4.16, it is clear that the fuzzy and fuzzy+prefilter controller graphs are significantly more accurate compared to the PID controller.

Table 4.3. Hover test results for 3 types of controllers with and without disturbance

		No disturbance			Disturbance applied at 0deg angle			Disturbance applied at 45deg angle		
		PID	Fuzzy	Fuzzy +prefilter	PID	Fuzzy	Fuzzy +prefilter	PID	Fuzzy	Fuzzy +prefilter
X	max	1.692E-03	1.005E-05	6.985E-04	8.894E-03	6.980E-04	6.538E-03	7.695E-05	8.486E-05	3.853E-03
	min	-3.218E-02	-2.345E-02	-3.565E-02	-3.573E-02	-2.081E-02	-2.967E-02	-3.560E-02	-3.603E-02	-5.343E-02
	P2P	3.387E-02	2.346E-02	3.635E-02	4.463E-02	2.151E-02	3.621E-02	3.568E-02	3.611E-02	5.729E-02
	mean	-1.296E-02	-1.292E-02	-1.734E-02	-1.138E-02	-1.111E-02	-1.336E-02	-2.142E-02	-1.833E-02	-1.937E-02

	median	-1.257E-02	-1.224E-02	-1.860E-02	-1.078E-02	-1.109E-02	-1.448E-02	-2.191E-02	-1.706E-02	-1.982E-02
	RMSE	1.607E-02	1.387E-02	1.954E-02	1.674E-02	1.206E-02	1.660E-02	2.283E-02	2.011E-02	2.251E-02
Y	max	1.709E-02	1.715E-02	1.432E-02	5.963E-03	1.431E-02	1.648E-03	2.381E-02	8.311E-03	1.159E-02
	min	-1.451E-02	-5.950E-03	-1.686E-02	-4.570E-02	-2.458E-02	-2.040E-02	-7.232E-02	-2.069E-02	-2.148E-02
	P2P	3.160E-02	2.310E-02	3.118E-02	5.166E-02	3.889E-02	3.688E-02	9.613E-02	2.900E-02	3.308E-02
	mean	2.895E-03	4.401E-03	1.283E-04	-1.095E-02	-1.731E-03	1.694E-03	-5.285E-03	-1.847E-03	-4.506E-04
	median	3.579E-03	3.363E-03	-7.998E-04	-4.628E-03	-8.928E-07	2.926E-03	-3.963E-03	-2.863E-05	4.546E-04
	RMSE	7.872E-03	7.175E-03	7.076E-03	1.812E-02	8.600E-03	7.540E-03	2.420E-02	5.724E-03	5.943E-03
Z	max	1.291E-02	3.452E-02	2.847E-02	5.261E-03	3.934E-02	3.046E-02	x	x	x
	min	-5.709E-02	1.539E-03	3.073E-03	-5.830E-02	7.277E-03	6.456E-03	x	x	x
	P2P	7.000E-02	3.298E-02	2.539E-02	6.356E-02	3.207E-02	2.400E-02	x	x	x
	mean	-1.469E-02	1.930E-02	1.294E-02	-1.670E-02	2.406E-02	1.866E-02	x	x	x
	median	-1.124E-02	2.071E-02	1.201E-02	-7.928E-03	2.420E-02	1.839E-02	x	x	x
	RMSE	2.457E-02	2.106E-02	1.422E-02	2.636E-02	2.507E-02	1.923E-02	x	x	x
Yaw	max	7.503E-03	5.465E-03	2.026E-02	5.958E-03	7.960E-03	3.823E-03	x	x	x
	min	-3.488E-02	-1.370E-02	-1.008E-02	-1.025E-01	-2.002E-02	-3.473E-02	x	x	x
	P2P	4.238E-02	1.917E-02	3.034E-02	1.085E-01	2.798E-02	3.855E-02	x	x	x
	mean	-3.400E-03	-2.961E-03	-7.137E-04	-2.831E-02	-1.799E-03	-6.344E-03	x	x	x
	median	-3.834E-03	-2.598E-03	-1.937E-03	-1.772E-03	-1.265E-03	-4.317E-03	x	x	x
	RMSE	5.295E-03	4.376E-03	5.034E-03	7.713E-03	4.633E-03	1.028E-02	x	X	x

The results of the Z-position controller tests, as shown in Figure 4.15, are somewhat uncertain. The PID controller results exhibited some overshoots and undershooting when the position was changed. The fuzzy controller behaved more stably in comparison; however, the best reference line shape repeat was achieved by the fuzzy controller with prefilter controller, even though it showed the most significant fluctuations. The hover test results in the table show that the fuzzy+prefilter results had fewer deviations from the desired point in both cases.

A total of 60 experiments were conducted, with 20 for each of the three types of controllers tested. The results of the hover tests for the PID, fuzzy, and fuzzy with prefilter controllers under three different conditions are illustrated in Figure A.2, Figure A**.8**, and Figure A**.14**, respectively. The corresponding controller command graphs for these experiments are provided in Figure A.1, Figure A.**7**, and Figure A.**13**.

The position control test results are depicted in Figure A.6, Figure A.**12**, and Figure A.**18** for the PID, fuzzy, and fuzzy with prefilter controllers, respectively. The corresponding controller command graphs for these experiments can be found in Figure A.4, Figure A.**10**, and Figure A.**16**.

4.3. Experimental Results: Final tests of the Leader-Follower system with optimal controller

The leader-follower experiments were carried out under conditions similar to those described in Section 4.2. The only thing that has changed is that the square inscribed by the waypoints has been reduced to a size of 0.6*0.6 m (Figure 4.17).

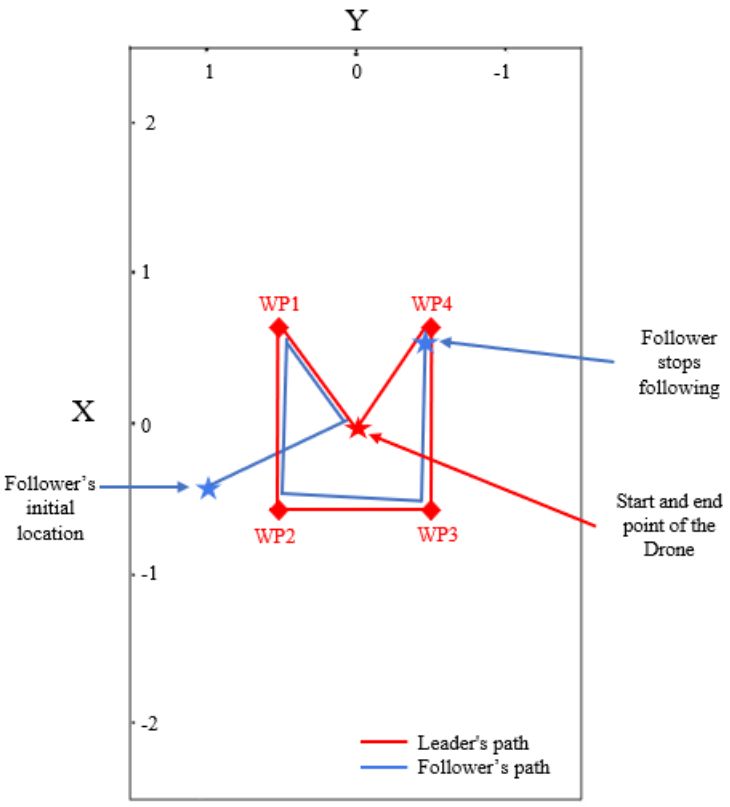
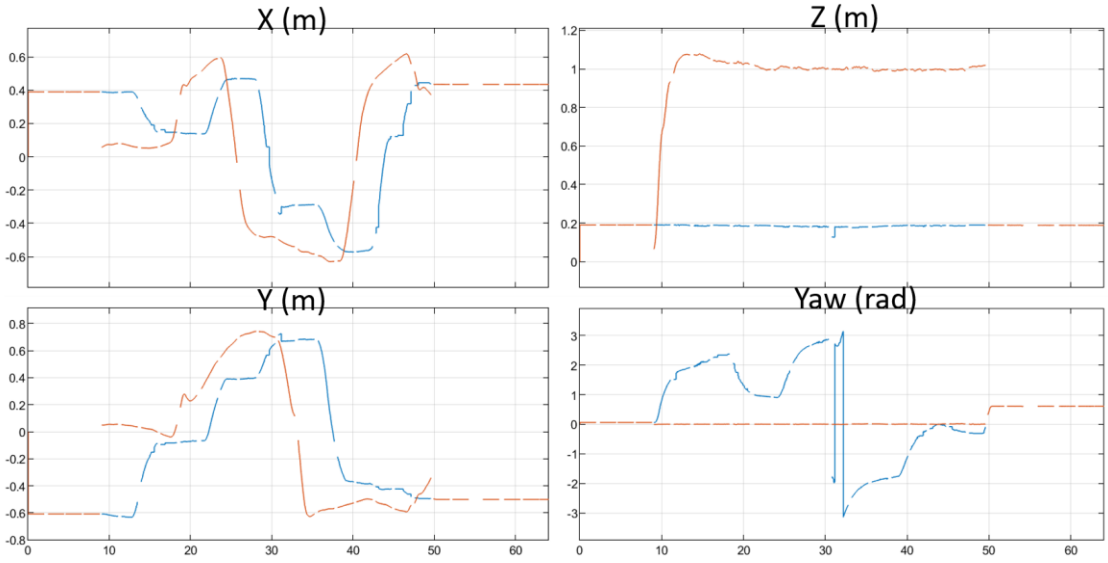


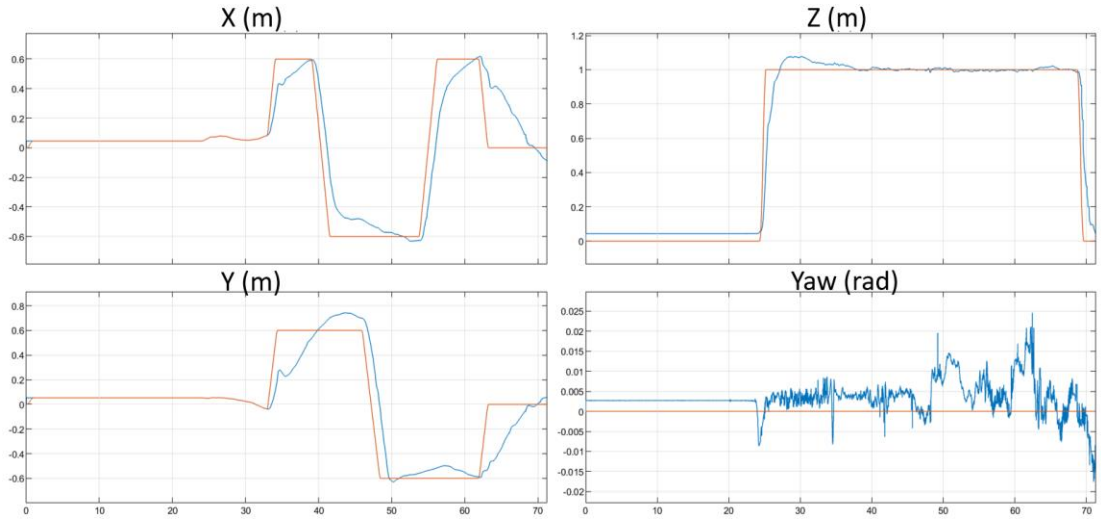
Figure 4.17: Location of waypoints for Leader-Follower test (WP - waypoint)

The results are presented in FiguresFigure **4.18** and Figure **4.19** for PID and fuzzy controllers As you can see, the QBot 2e trajectory data is not complete: graphs often break off due to lack of computing power. In fact, FLC is a block that creates a significant load on the PC when running only for QDrone. Now, when the system needs to communicate with two vehicles, the load on the PC increases, which leads to interruptions in Figure 4.18a and Figure 4.19a.

Comparing Figures Figure **4.18**Figure 4.19, it can be seen that the accuracy of the QDrone's repositioning is significantly higher in the test conducted for the fuzzy controller. Whereas QBot 2e worked the same in both experiments.

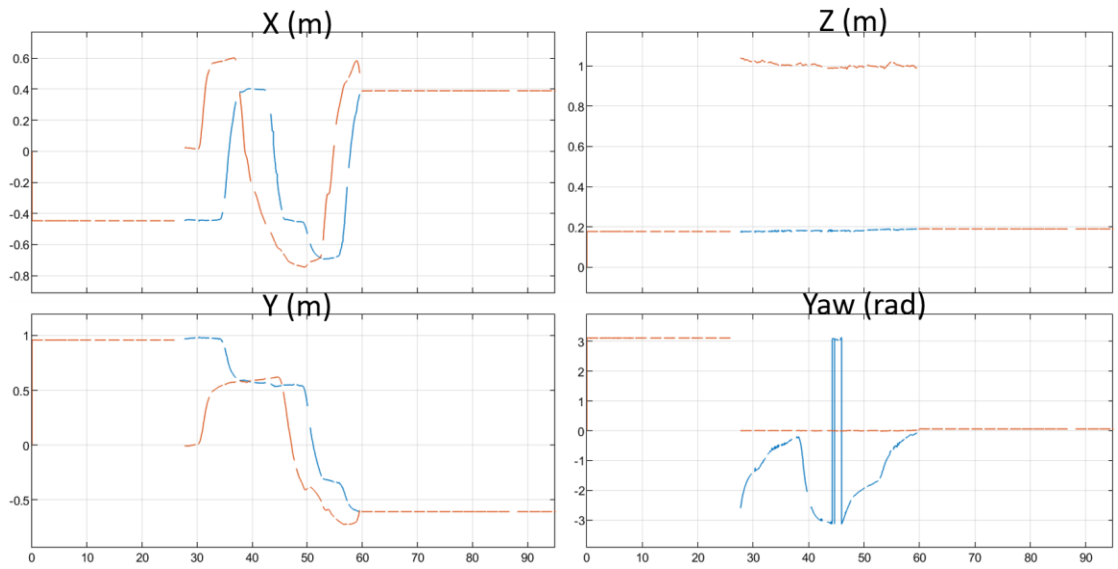


a) QBot 2e trajectory

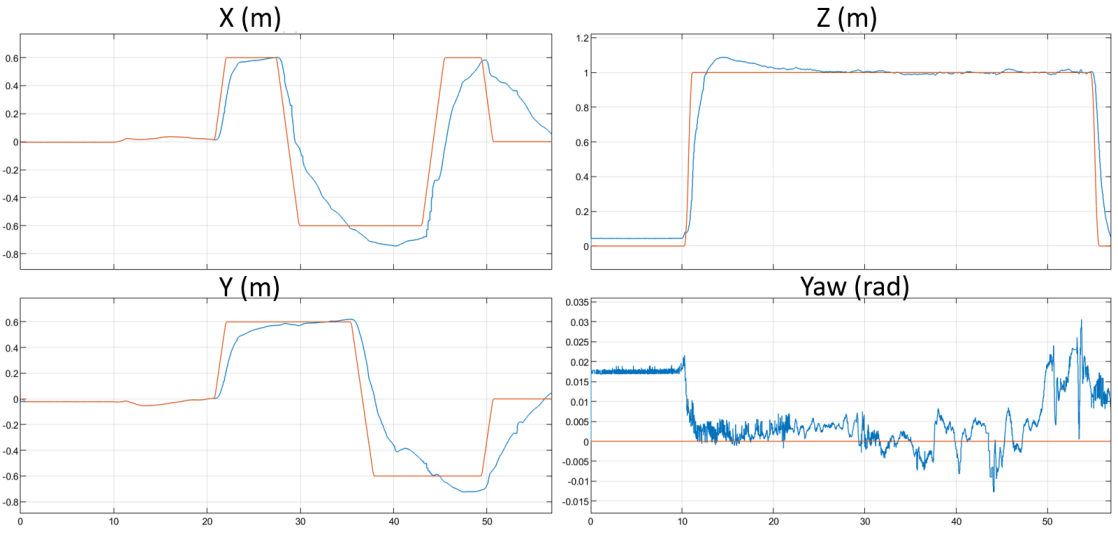


b) QDrone trajectory

Figure 4.18: Leader-follower results for PID



a) QBot 2e trajectory



b) QDrone trajectory

Figure 4.19: Leader-follower results for Fuzzy

4.4. Experimental Results: Final tests of the obstacle avoidance algorithm in leader-follower system with the optimal controller

The obstacle avoidance algorithm was subjected to testing using an optimal controller, under the same conditions as those outlined in Section 4.2, with the exception of the controller used and waypoint locations (Figure 4.20). The waypoint locations were changed in order to

shorten the testing time and prevent the tracking loss issue, as it was done for leader-follower algorithm testing in Section 4.3.

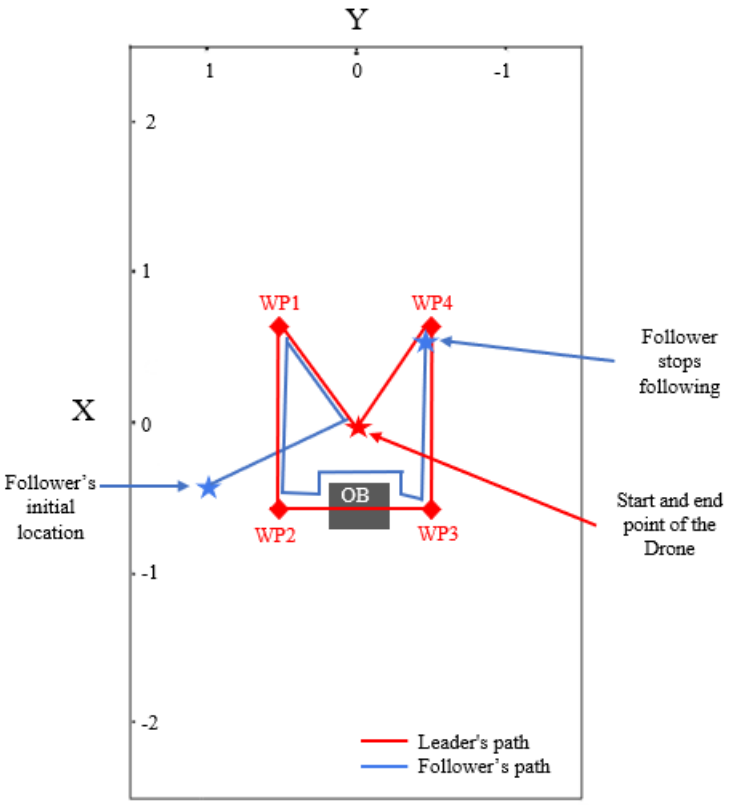
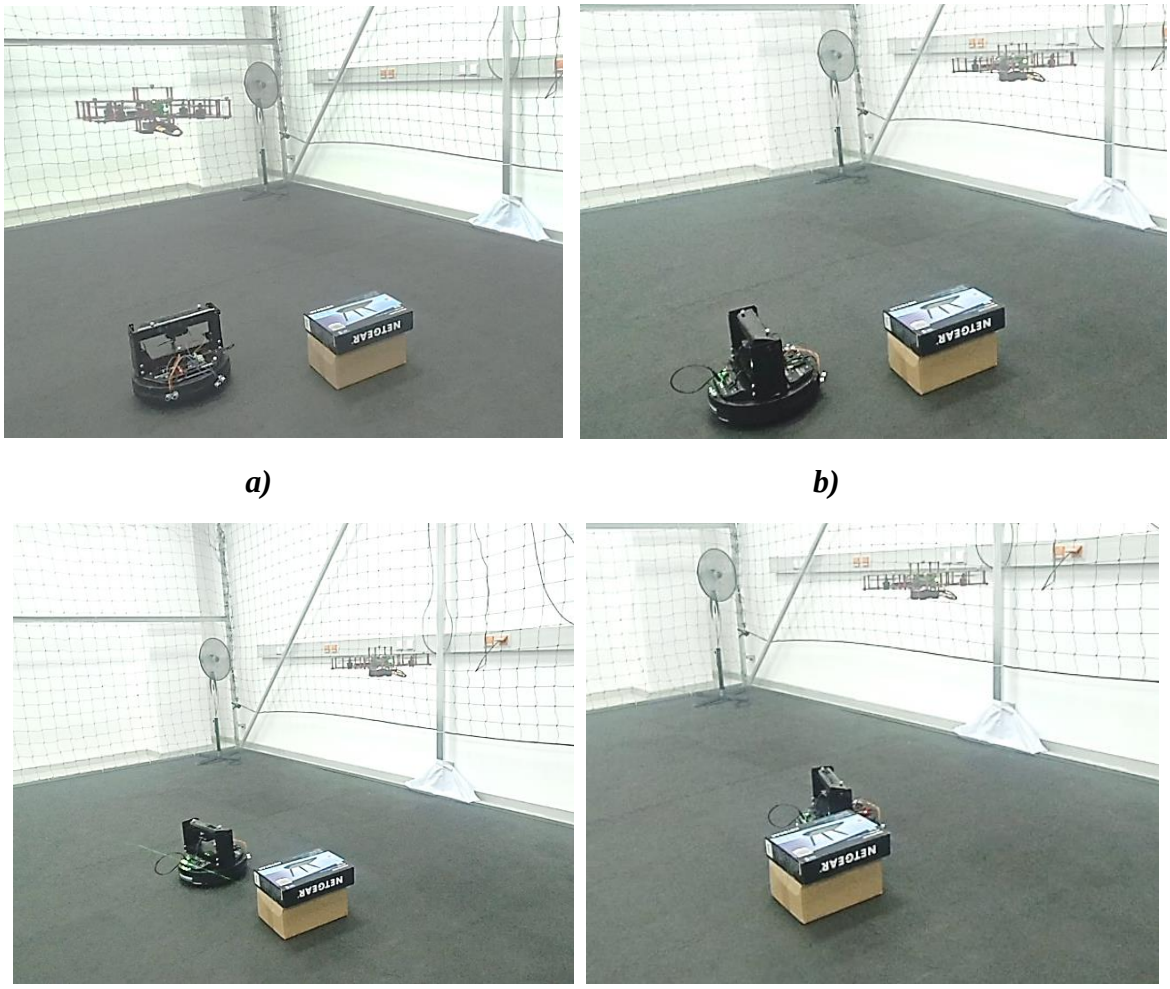


Figure 4.20: Waypoints and obstacle locations for obstacle avoidance tests with the optimal controller

A fuzzy tracking controller was employed for both the X and Y positions. The execution of the leader-follower algorithm was employed during the obstacle avoidance process, with several stages of the process captured in Figure 4.21. Initially, Figure 4.21a depicts the leader (QDrone) and follower (Qbot 2e) located at the same X-Y coordinates. Subsequently, the leader moves towards the next mission server-given waypoint, with the QBot 2e following suit until it encounters an obstacle (Figure 4.21b). Once the obstacle is detected, the obstacle avoidance algorithm generates a new path towards the desired point (Figure 4.21c). Finally, in Figure 4.21d the 2-wheeled mobile robot follows the newly generated path and reaches the desired point.



a) *b)*

c) *d)*

Figure 4.21: Performance of the obstacle avoidance algorithm in leader-follower system with the optimal controller

Chapter 5 – Conclusions and Future Works

5.1. Conclusion

In conclusion, this thesis presents the development of an optimal controller for a Leader-Follower system with obstacle avoidance. The experiments and simulations conducted in this thesis show that the proposed Fuzzy controller is more accurate and effective in controlling the position of the quadcopter than the PID controller. Furthermore, the integration of the optimal Fuzzy controller into the Leader-Follower system with a Fuzzy controller has shown to be effective in avoiding obstacles and maintaining the desired position of the quadcopter. The proposed algorithm and controller contribute to the advancement of the field of robotics and autonomous systems and can be applied in various industries such as transportation, surveillance, and agriculture.

The system behavior when using PID and fuzzy logic controllers is compared in this article. For this, a system was built using Quanser's QDrone and tested with and without fan disturbance. It was observed during the tests that the drone's behavior became noticeably unstable when the battery charge fell below 70 %. For this reason, all tests were carried out with high battery charge and at an equal distance from the take-off point to the pedestal fan. The hover test findings demonstrated that the prefiltered FLC caused the drone to behave most steadily during tests conducted for Z-position control. At that time, the FLC without a prefilter was the leader in tests of controlling the other 3 components of position (X, Y, Yaw) according to the RMSE analysis. However, according to the graphical results showed that during the Yaw-position tests the Drone is more stable when PID controller is applied. This is also seen in position control tests, where the results conducted for yaw show that the PID controller performs slightly better than the Fuzzy controller, while FLC with prefilter demonstrated the worst behavior, which was caused by overshoots. This means that the prefilter may be tuned better in further works. The X, Y, and Z position control test results show that PID controller performance is less accurate compared with the two other controllers.

The optimal controller choice process can be concluded as:

- Fuzzy with prefilter shown the most stable behavior for Z-position control
- Fuzzy shown the most stable behavior for X- and Y-positions and control

- ❖ PID shown the most stable behavior for Yaw-position control

The final test conducted on leader-follower algorithm also prove the competitiveness of fuzzy controller comparing with PID. So, the optimal fuzzy controller with 49 rules showed the most accurate results in all tests and can be implemented in various leader-follower systems.

5.2. Contribution to Knowledge

This thesis has made a significant contribution to the field of robotics by developing an optimal controller for a Leader-Follower system with obstacle avoidance. The system consisted of a quadcopter acting as the leader and a two-wheeled mobile robot acting as the follower.

Through experiments with Fuzzy, PID, and Fuzzy with prefilter controllers, it was determined that Fuzzy was the most accurate and effective in controlling the position of the quadcopter. An algorithm was then created for the Leader-Follower system with obstacle avoidance, and experiments were conducted to compare the performance of the Fuzzy and PID controllers under different conditions.

The findings of this thesis demonstrate that the optimal Fuzzy controller can successfully control the Leader-Follower system with obstacle avoidance and can perform more accurately and consistently than the PID controller. This contribution to knowledge will benefit the field of robotics by providing a new and effective controller design for Leader-Follower systems with obstacle avoidance.

5.3. Future Work

This thesis aimed to create an optimal controller for a Leader-Follower system with obstacle avoidance, using a quadcopter as the leader and a 2-wheeled mobile robot as the follower. The Fuzzy controller was found to perform more accurately and better in controlling the position of the quadcopter. An algorithm for the Leader-Follower system with obstacle avoidance was created, and experiments were conducted comparing the Fuzzy and PID controllers under different conditions. The optimal Fuzzy controller was then integrated into the system with the Leader-Follower algorithm with a Fuzzy controller. The contribution of this thesis lies in the development of an optimal Fuzzy controller for the Leader-Follower system with obstacle avoidance and the demonstration of its superior performance. Future work includes the usage of a higher order

prefilter, replacement of the onboard CPU of QBot 2e, and creation of adapting algorithms for the Fuzzy controller.

References

[1] Y. Wu, Z. Zuo, Q. Han, Y. Wang, and H. Yang, "Formation Control of Wheeled Mobile Robots with Multiple Virtual Leaders Under Communication Failures," *IEEE Transactions on Control Systems Technology*, 2022.

[2] S. Chang, Y., Wang, and Z. Zuo, "Formation Control for Wheeled Mobile Robots With Finite-Time Active Disturbance Rejection Control," in 2019 *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 818-822, August 2019.

[3] A. Drak, M. Hejase, M. ElShorbagy, A. Wahyudie, and H. Noura, "Autonomous formation flight algorithm and platform for quadrotor uavs," *International Journal of Robotics and Mechatronics*, vol. 1, no. 4, pp. 124-132, 2014.

[4] K. Gopakumar, and K. V. Shihabudheen, "Leader Follower Distributed Consensus Control of Heterogeneous Multi-agent System with Model Predictive Control," in *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pp. 25-30, September 2021.

[5] M. A. Kamel, X. Yu, and Y. Zhang, "Real-time optimal formation reconfiguration of multiple wheeled mobile robots based on particle swarm optimization," in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pp. 703-708, June 2016.

[6] M. A. Kamel, X. Yu, and Y. Zhang, "Fault-tolerant cooperative control design of multiple wheeled mobile robots," *IEEE Transactions on control systems technology*, vol. 26, no. 2, pp. 756-764, 2017.

[7] M. A. Kamel, K. A. Ghamry, and Y. Zhang, "Real-time fault-tolerant cooperative control of multiple UAVs-UGVs in the presence of actuator faults," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2, pp. 469-480, 2017.

[8] J. Luo, C. L. Liu, and F. Liu, "A leader-following formation control of multiple mobile robots with obstacle," in *2015 IEEE International Conference on Information and Automation*, pp. 2153-2158, August 2015.

[9] K. A. Ghamry, Y. Dong, M. A. Kamel, and Y. Zhang, "Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform," in *2016 24th Mediterranean conference on control and automation (MED)*, pp. 1236-1241, June 2016.

[10] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[11] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "Pid, lqr and lqr-pid on a quadcopter platform," in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2013, pp. 1–6.

[12] C. Liu, J. Pan, and Y. Chang, "Pid and lqr trajectory tracking control for an unmanned quadrotor helicopter: Experimental studies," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 10 845–10 850.

[13] A. Milhim, Y. Zhang, and C.-A. Rabbath, "Gain scheduling based pid controller for fault tolerant control of quad-rotor uav," in *AIAA infotech@ aerospace 2010*, 2010, p. 3530.

[14] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[15] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav for extreme maneuverability," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6337–6342, 2011.

[16] D. Invernizzi and M. Lovera, "Geometric tracking control of a quadcopter tiltrotor uav," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 565–11 570, 2017.

[17] C. Ha, Z. Zuo, F. B. Choi, and D. Lee, "Passivity-based adaptive backstepping control of quadrotor-type uavs," *Robotics and Autonomous Systems*, vol. 62, no. 9, pp. 1305–1315, 2014.

[18] M. E. Antonio-Toledo, E. N. Sanchez, A. Y. Alanis, J. Flórez, and M. A. Perez-Cisneros, "Real-time integral backstepping with sliding mode control for a quadrotor uav," *IFAC-PapersOnLine*, vol. 51, no. 13, pp. 549–554, 2018.

[19] J.-J. Xiong and E.-H. Zheng, "Position and attitude tracking control for a quadrotor uav," *ISA transactions*, vol. 53, no. 3, pp. 725–731, 2014.

[20] C. Li, Y. Wang, and X. Yang, "Adaptive fuzzy control of a quadrotor using disturbance observer," *Aerospace Science and Technology*, vol. 128, p. 107784, 2022.

[21] X. Zhang, Y. Wang, G. Zhu, X. Chen, Z. Li, C. Wang, and C.-Y. Su, "Compound adaptive fuzzy quantized control for quadrotor and its experimental verification," *IEEE transactions on cybernetics*, vol. 51, no. 3, pp. 1121–1133, 2020.

[22] B. Erginer and E. Altug, "Design and implementation ˘ of a hybrid fuzzy logic controller for a quadrotor vtol vehicle," *International Journal of Control, Automation and Systems*, vol. 10, no. 1, pp. 61–70, 2012.

[23] A. Al-Mahturi, F. Santoso, M. A. Garratt, and S. G. Anavatti, "Nonlinear altitude control of a quadcopter drone using interval type-2 fuzzy logic," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 236–241.

[24] T. Bodrumlu, M. T. Soylemez, and I. Mutlu, "Modelling and control of the qball x4 quadrotor system based on pid and fuzzy logic structure," in *Journal of Physics: Conference Series*, vol. 783, no. 1. IOP Publishing, 2017, p. 012039.

[25] P. Ye, Y. Yu, and W. Wang, "Event-triggered control for trajectory tracking of quadrotor unmanned aerial vehicle," *Systems Science & Control Engineering*, vol. 10, no. 1, pp. 241–254, 2022.

[26] O. Saleem, M. Rizwan, A. A. Zeb, A. H. Ali, and M. A. Saleem, "Online adaptive pid tracking control of an aero-pendulum using pso-scaled fuzzy gain adjustment mechanism," *Soft Computing*, vol. 24, no. 14, pp. 10 629–10 643, 2020.

[27] J. Rao, B. Li, Z. Zhang, D. Chen, and W. Giernacki, "Position control of quadrotor uav based on cascade fuzzy neural network," *Energies*, vol. 15, no. 5, pp. 1763, 2022.

[28] B. Kamel, B. Yasmina, B. Laredj, I. Benaoumeur, & A. F. Zoubir, "Dynamic modeling, simulation and PID controller of unmanned aerial vehicle UAV," In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, IEEE, pp. 64-69, 2017.

[29] M. Vazquez, M. Ardito-Proulx, & S. Wadoo, “Lyapunov Based Trajectory Tracking Dynamic Control for a QBOT-2”. In *2020 IEEE Integrated STEM Education Conference (ISEC)*, IEEE, pp. 1-6, 2020.

[30] NaturalPoint, Inc. DBA OptiTrack, “Motion Capture for Movement Sciences,” OptiTrack, 2022. Retrieved July 18, 2022, from https://OptiTrack.com/applications/movement-sciences/#:%7E:text=The%20most%20accurate%20measurement%20system,of%200.1%20mm%20or%20less.

[31] “Raspberry Pi Documentation - Raspberry Pi hardware,” Raspberry PI Ltd, 2021-2022. Retrieved November 14, 2022, from https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[32] A. Tassanbi, A. Iskakov, T. D. Do, & M. H. Ali, “Interactive Real-time Leader Follower Control System for UAV and UGV,” In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, IEEE, pp. 1-8, 2022.

[33] Y. Raziyev, R. Garifulin, A. Shintemirov, & T. D. Do, “Development of a power assist lifting device with a fuzzy PID speed regulator,” *IEEE Access*, vol. 7, pp. 30724-30731, 2019.

[34] M. U. Asad, J. Gu, U. Farooq, R. Dey, V. E. Balas, and G. Abbas, “Intelligent Obstacle Avoidance Controller for QBot2,” *IFAC-PapersOnLine*, vol. 55, no. 1, pp. 120-125, 2022.

Appendices

Appendix I. Dynamic modelling equations derivation

Inertial Frame $\vec{E}$: $\{\overrightarrow{e_1}, \overrightarrow{e_2}, \overrightarrow{e_3}\}$

Body Frame $\vec{B}$: $\{\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}\}$

Horizon Frame $\vec{H}$: $\{0, 0, \overrightarrow{h_3}\}$

$$\mathbf{q} = [X \quad \omega]^T = [x \quad y \quad z \quad \varphi \quad \theta \quad \psi]^T$$

where $X = [x \quad y \quad z]^T$; $\omega = [\varphi \quad \theta \quad \psi]^T = [Roll \quad Pitch \quad Yaw]^T$ in $\vec{E}$ frame.

$$\mathbf{V} = [V \quad \Omega]^T = [v_x \quad v_y \quad v_z \quad p \quad q \quad r]^T$$

where V – linear, Ω - angular velocity in $\vec{B}$ frame.

Rotation matrices:

$$R_\varphi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\varphi & -s\varphi \\ 0 & s\varphi & c\varphi \end{bmatrix}$$

$$R_\theta = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

$$R_\psi = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The general rotation matrix:

$$R = R_\psi R_\theta R_\varphi = \begin{bmatrix} c\theta c\psi & s\varphi s\theta c\psi - c\varphi s\psi & c\varphi s\theta c\psi + s\varphi s\psi \\ c\theta s\psi & s\varphi s\theta s\psi + c\varphi s\psi & c\varphi s\theta s\psi - s\varphi c\psi \\ -s\theta & s\varphi c\theta & c\varphi c\theta \end{bmatrix}$$

Relation between $\dot{\omega}$ and Ω:

$$\dot{R} = R\Omega = \begin{bmatrix} c\theta c\psi & s\varphi s\theta c\psi - c\varphi s\psi & c\varphi s\theta c\psi + s\varphi s\psi \\ c\theta s\psi & s\varphi s\theta s\psi + c\varphi s\psi & c\varphi s\theta s\psi - s\varphi c\psi \\ -s\theta & s\varphi c\theta & c\varphi c\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Ω is defined as:

$$\Omega = R^T \dot{R}$$

where

$$\dot{R} = \frac{d}{dt}\left(R_\psi R_\theta R_\varphi\right) = \dot{R}_\psi R_\theta R_\varphi + R_\psi \dot{R}_\theta R_\varphi + R_\psi R_\theta \dot{R}_\varphi$$

$$R^T = \left(R_\psi R_\theta R_\varphi\right)^T = {R_\varphi}^T {R_\theta}^T {R_\psi}^T$$

So that

$$\Omega = {R_\varphi}^T {R_\theta}^T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}^T + {R_\varphi}^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix}$$

$$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\varphi & s\varphi c\theta \\ 0 & -s\varphi & c\varphi c\theta \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Relation between **q** and **V**:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\varphi s\theta c\psi - c\varphi s\psi & c\varphi s\theta c\psi + s\varphi s\psi & 0 & 0 & 0 \\ c\theta s\psi & s\varphi s\theta s\psi + c\varphi s\psi & c\varphi s\theta s\psi - s\varphi c\psi & 0 & 0 & 0 \\ -s\theta & s\varphi c\theta & c\varphi c\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -s\theta \\ 0 & 0 & 0 & 0 & c\varphi & s\varphi c\theta \\ 0 & 0 & 0 & 0 & -s\varphi & c\varphi c\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ p \\ q \\ r \end{bmatrix}$$

Considering the forces affecting the flight:

$$ma = F - mg$$

$$F = F_X + F_t$$

where F_X - is the force of translational movement generated by propellers. F_t is the drag forces applied to the quadrotor along $\vec{B}$.

$$F_X = fRe_3 = f \begin{bmatrix} c\varphi s\theta c\psi + s\varphi s\psi \\ c\varphi s\theta s\psi - s\varphi c\psi \\ c\varphi c\theta \end{bmatrix}$$

where f is the sum of the forces generated by propellers.

$$F_t = \begin{bmatrix} -K_{fx} & 0 & 0 \\ 0 & -K_{fy} & 0 \\ 0 & 0 & -K_{fz} \end{bmatrix} \dot{X}$$

where K_{fx}, K_{fy}, K_{fz} – are drag coefficients.

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = f \begin{bmatrix} c\varphi s\theta c\psi + s\varphi s\psi \\ c\varphi s\theta s\psi - s\varphi c\psi \\ c\varphi c\theta \end{bmatrix} + \begin{bmatrix} -K_{fx} & 0 & 0 \\ 0 & -K_{fy} & 0 \\ 0 & 0 & -K_{fz} \end{bmatrix} \dot{X} - mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$m\ddot{x} = (c\varphi s\theta c\psi + s\varphi s\psi)f - K_{fx}\,\dot{x}$$

$$m\ddot{y} = (c\varphi s\theta s\psi - s\varphi c\psi)f - K_{fy}\,\dot{y}$$

$$m\ddot{z} = c\varphi c\theta f - K_{fz}\,\dot{z} - mg$$

Lagrangian of quadrotor:

$$L(q,\dot{q}) = Kinetic\ Energy - Potejtial\ Energy = KE_{trans} + KE_{rot} - U$$

$$L(q,\dot{q}) = \frac{1}{2}m\dot{X}^T\dot{X} + \frac{1}{2}\dot{\Omega}^T J\dot{\Omega} - mgz$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{q}}\right) - \frac{\delta L}{\delta q} = \begin{bmatrix} F \\ \tau \end{bmatrix}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{q}}\right) = \begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \\ J_x\ddot{\varphi} + J_x\dot{\varphi} \\ J_y\ddot{\theta} + J_y\dot{\theta} \\ J_z\ddot{\psi} + J_z\dot{\psi} \end{bmatrix}$$

where $J = diag(J_x\ J_y\ J_z)$ is the inertia matrix, $\tau = \left[\tau_\varphi\ \ \tau_\theta\ \ \tau_\psi\right]^T$ is generalized moments around UAV.

$$\frac{\delta L}{\delta q} = [0 \quad 0 \quad -mg \quad 0 \quad 0 \quad 0]$$

$$J\dot{\Omega} = \Omega \times J\Omega + \sum_1^4 J_r\Omega \times e_3\Omega_i + \tau$$

$$\begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}\begin{bmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\dot{\psi}\dot{\theta}I_y + \dot{\psi}\dot{\theta}I_z \\ \dot{\psi}\dot{\varphi}I_x - \dot{\psi}\dot{\varphi}I_z \\ -\dot{\varphi}\dot{\theta}I_x + \dot{\varphi}\dot{\theta}I_y \end{bmatrix} + J_r\begin{bmatrix} \dot{\theta} \\ -\dot{\varphi} \\ 0 \end{bmatrix}\Omega_r + \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}$$

where $\Omega_r = \omega_2 + \omega_4 - \omega_1 - \omega_3$

$$I_x\ddot{\varphi} = \dot{\theta}\dot{\psi}(I_y - I_z) - J_r\dot{\theta}\Omega_r + L_{roll}\tau_{roll}$$

$$I_y\ddot{\theta} = \dot{\varphi}\dot{\psi}(I_z - I_x) - J_r\dot{\psi}\Omega_r + L_{pitch}\tau_{pitch}$$

$$I_z\ddot{\psi} = \dot{\varphi}\dot{\theta}(I_x - I_y) + \tau_{yaw}$$

the thrust generated by i-th propeller along $\vec{b}_3$ is:

$$f_i = bu_i$$

where b – thrust factor. Moment generated by i-th rotor:

$$\tau_i = du_i$$

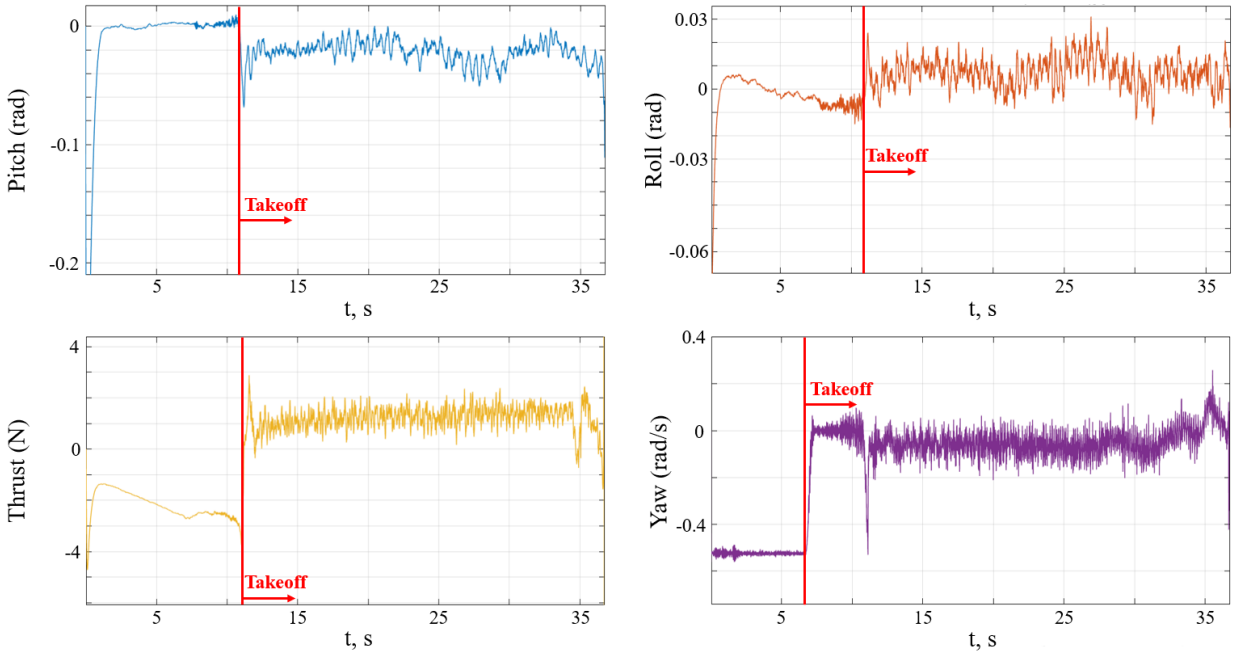
where d – yaw torque constant.

$$\begin{bmatrix} F \\ \tau_{roll} \\ \tau_{pitch} \\ \tau_{yaw} \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ -b\frac{L_{roll}}{2} & -b\frac{L_{roll}}{2} & b\frac{L_{roll}}{2} & b\frac{L_{roll}}{2} \\ b\frac{L_{pitch}}{2} & -b\frac{L_{pitch}}{2} & b\frac{L_{pitch}}{2} & -b\frac{L_{pitch}}{2} \\ d & -d & -d & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

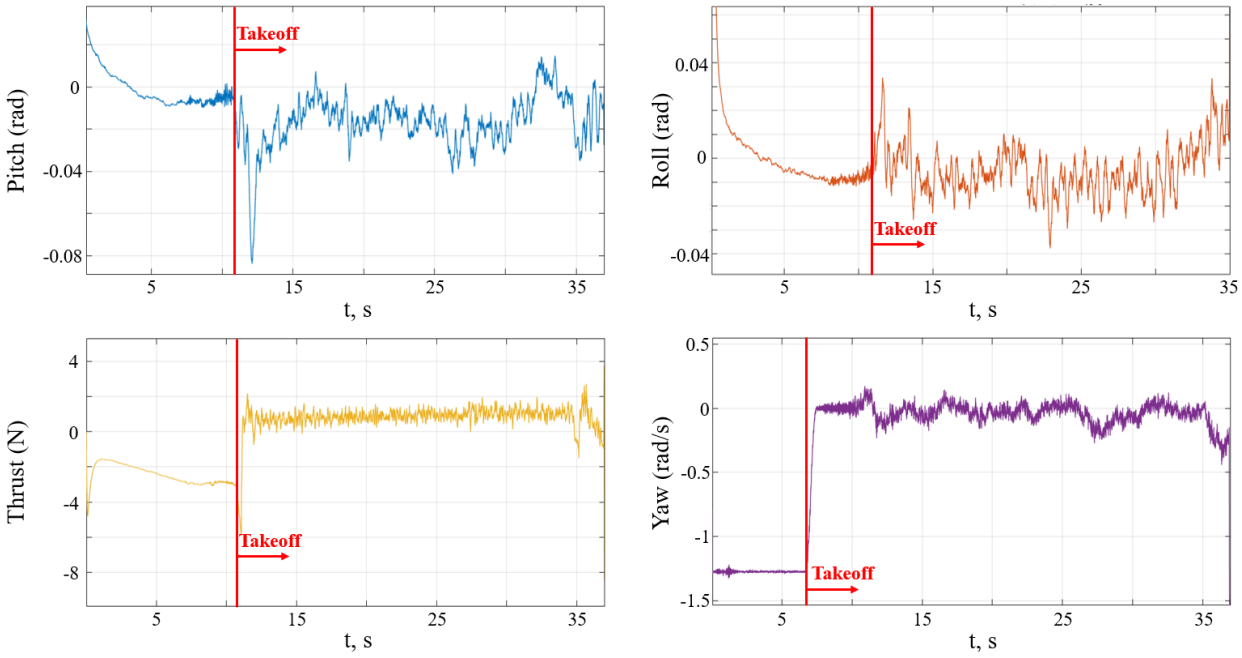
Finally, u is expressed by thrust forces:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & -\frac{1}{2bL_{roll}} & \frac{1}{2bL_{pitch}} & \frac{1}{4d} \\ \frac{1}{4b} & -\frac{1}{2bL_{roll}} & -\frac{1}{2bL_{pitch}} & \frac{1}{-4d} \\ \frac{1}{4b} & \frac{1}{2bL_{roll}} & \frac{1}{2bL_{pitch}} & -\frac{1}{4d} \\ \frac{1}{4b} & \frac{1}{2bL_{roll}} & -\frac{1}{2bL_{pitch}} & \frac{1}{4d} \end{bmatrix} \begin{bmatrix} F \\ \tau_{roll} \\ \tau_{pitch} \\ \tau_{yaw} \end{bmatrix}$$

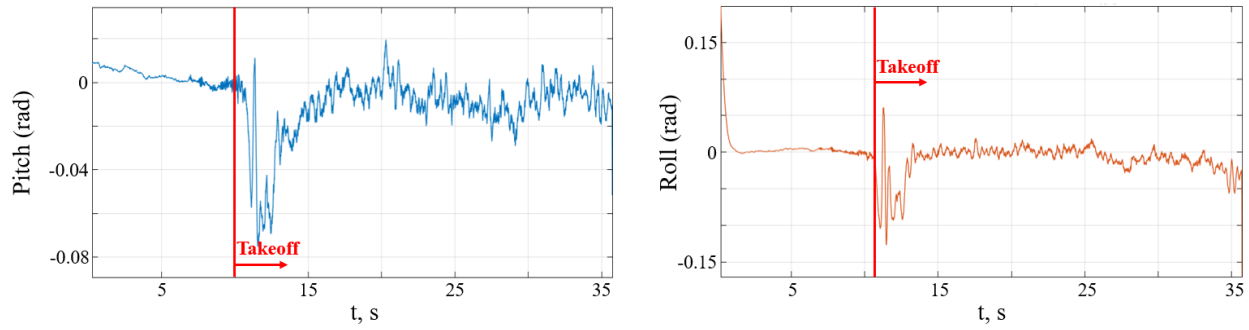
Appendix II. Fuzzy vs PID comparison results



a) Disturbance is not applied

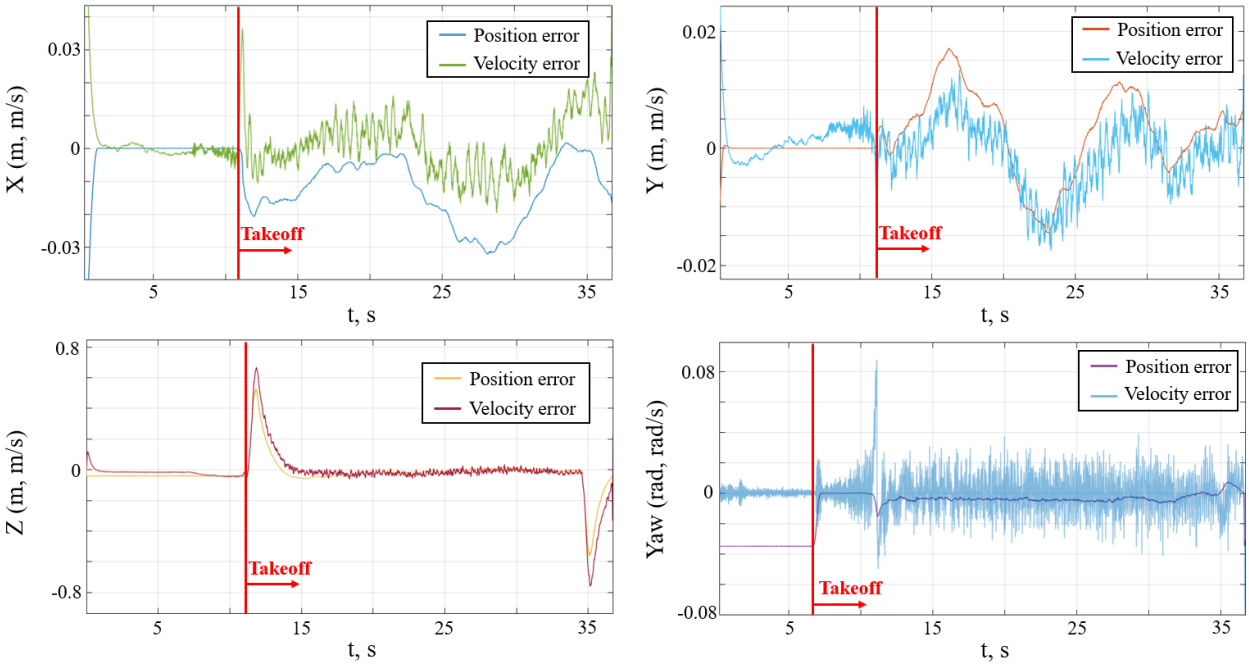


b) Disturbance applied at 0deg angle

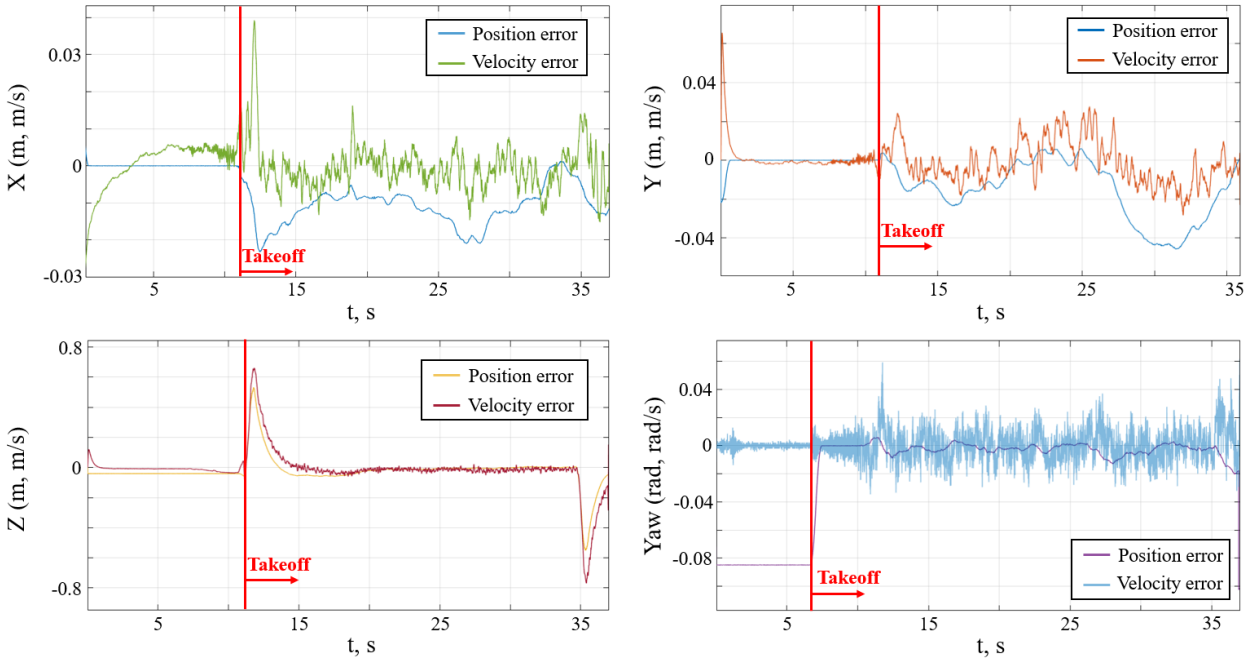


c) Disturbance applied at 45deg angle

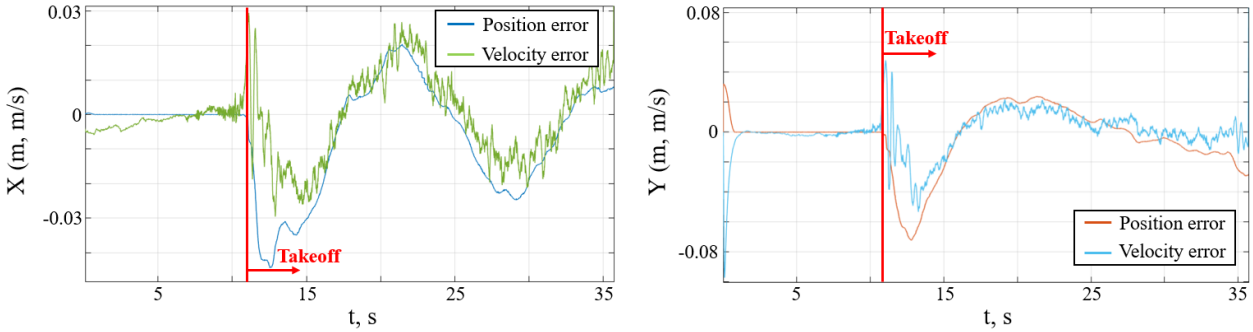
Figure A.1: Unsaturated controller commands vs time (Hover test, PID)



a) Disturbance is not applied

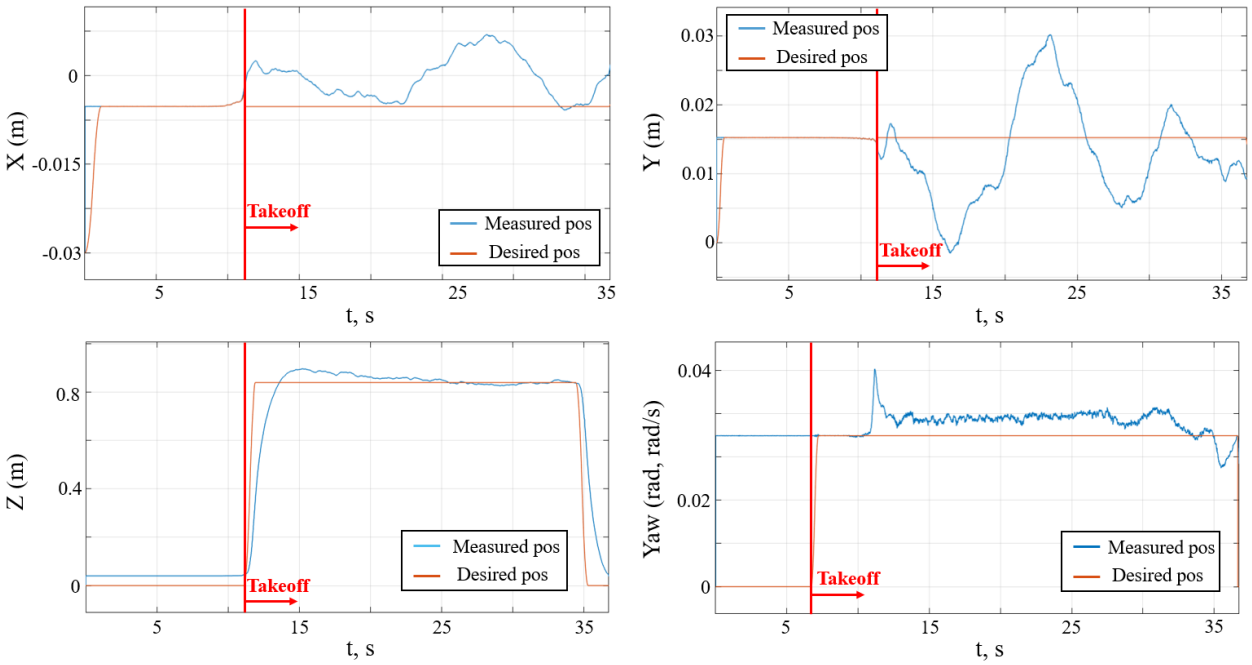


b) Disturbance applied at 0deg angle

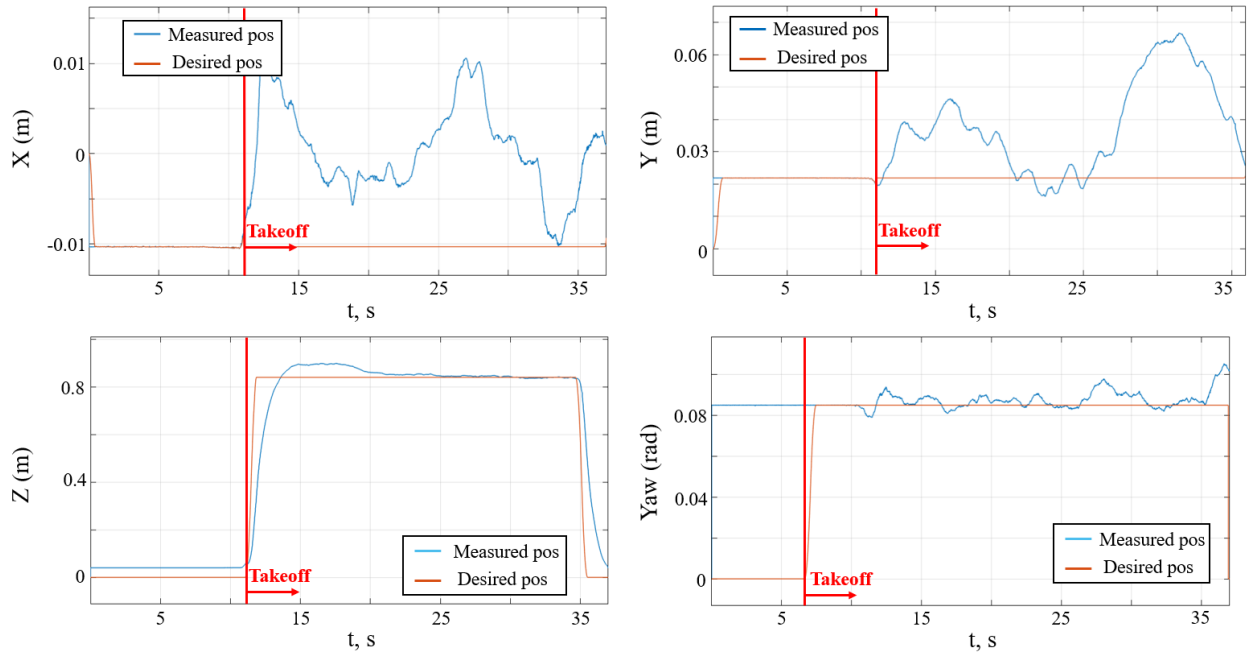


c) Disturbance applied at 45deg angle

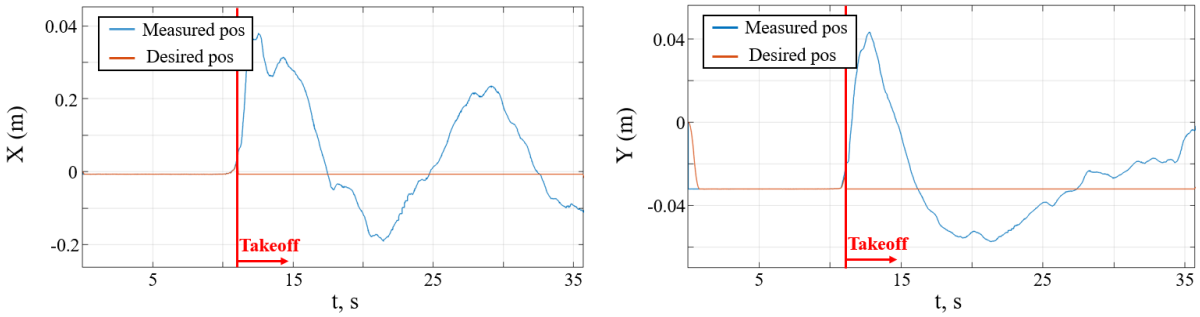
Figure A.2: Position error and Velocity error vs time (Hover test, PID)



a) Disturbance is not applied

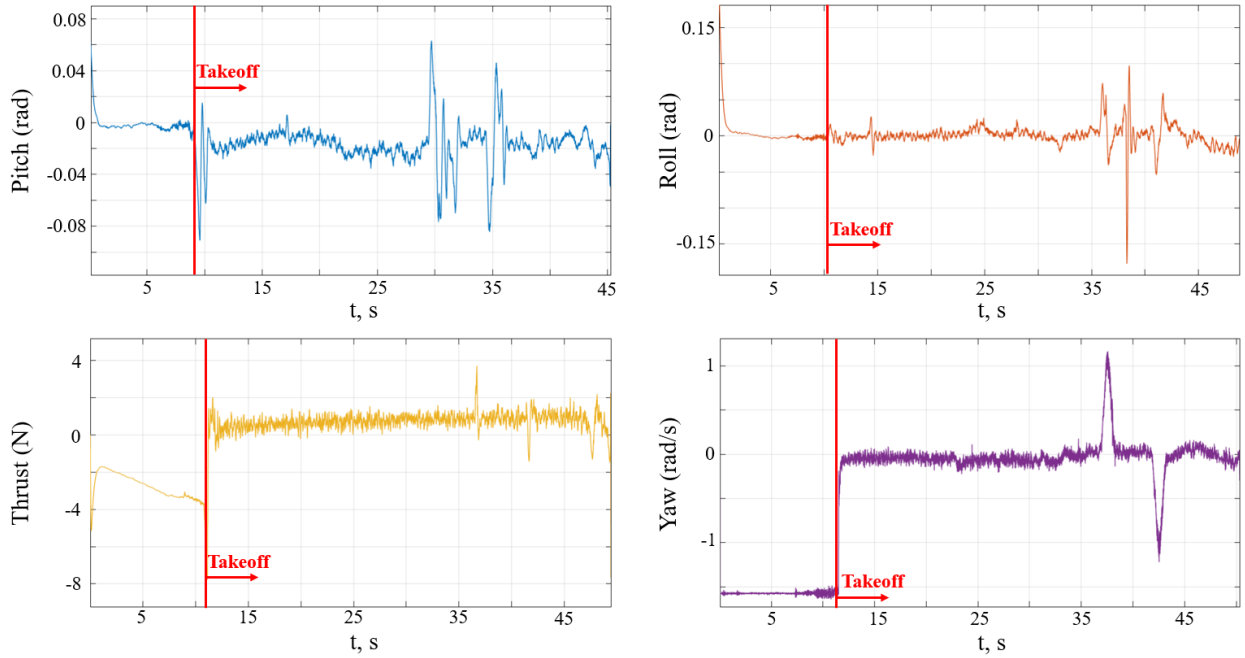


b) Disturbance applied at 0deg angle

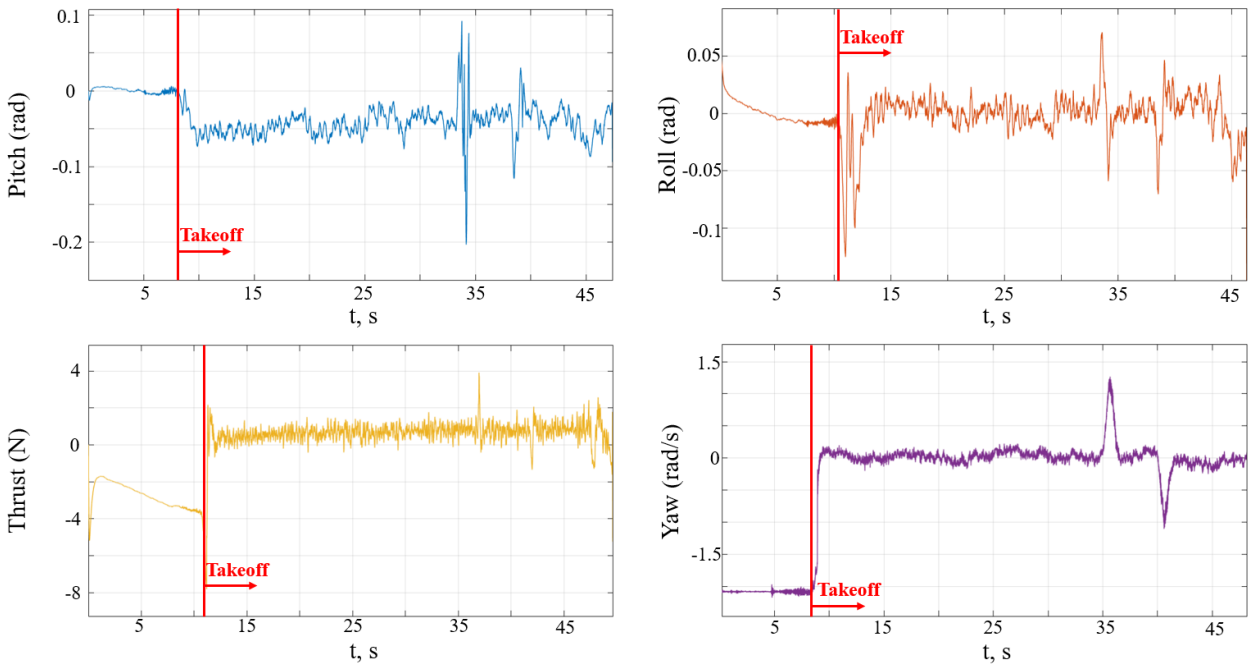


c) Disturbance applied at 45deg angle

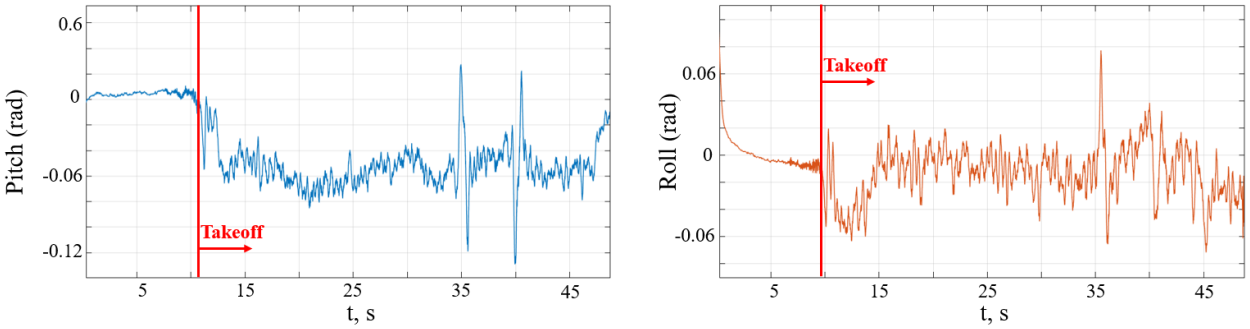
Figure A.3: Position vs time (Hover test, PID)



a) Disturbance is not applied

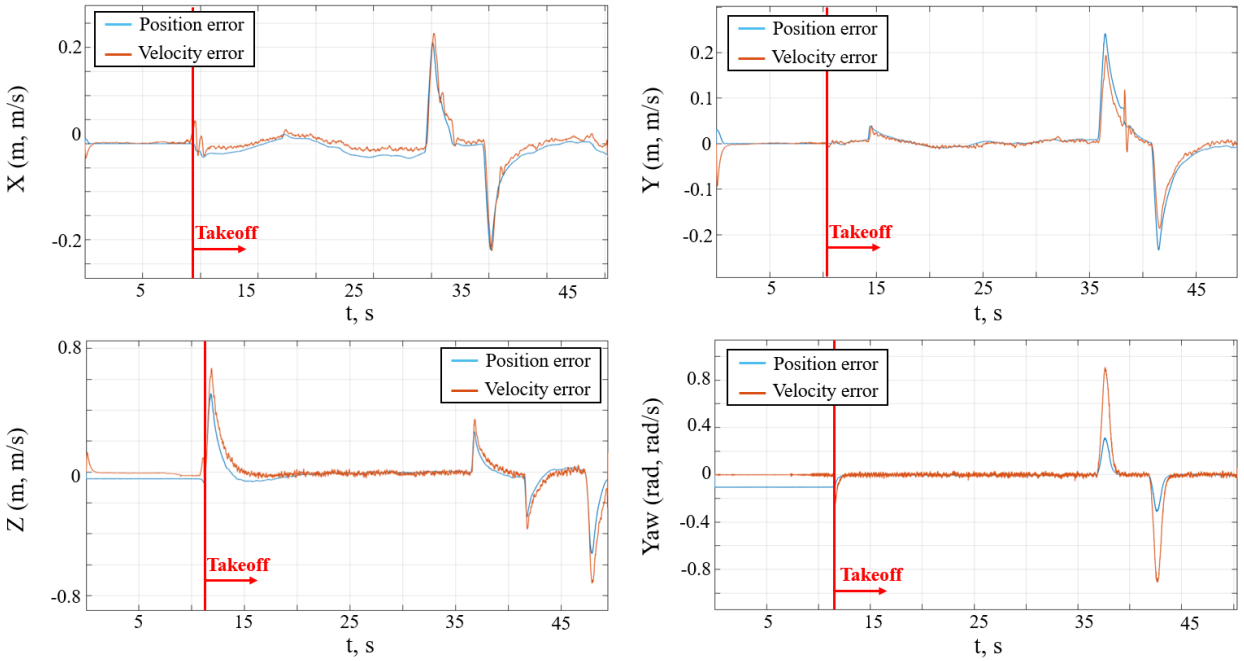


b) Disturbance applied at 0deg angle

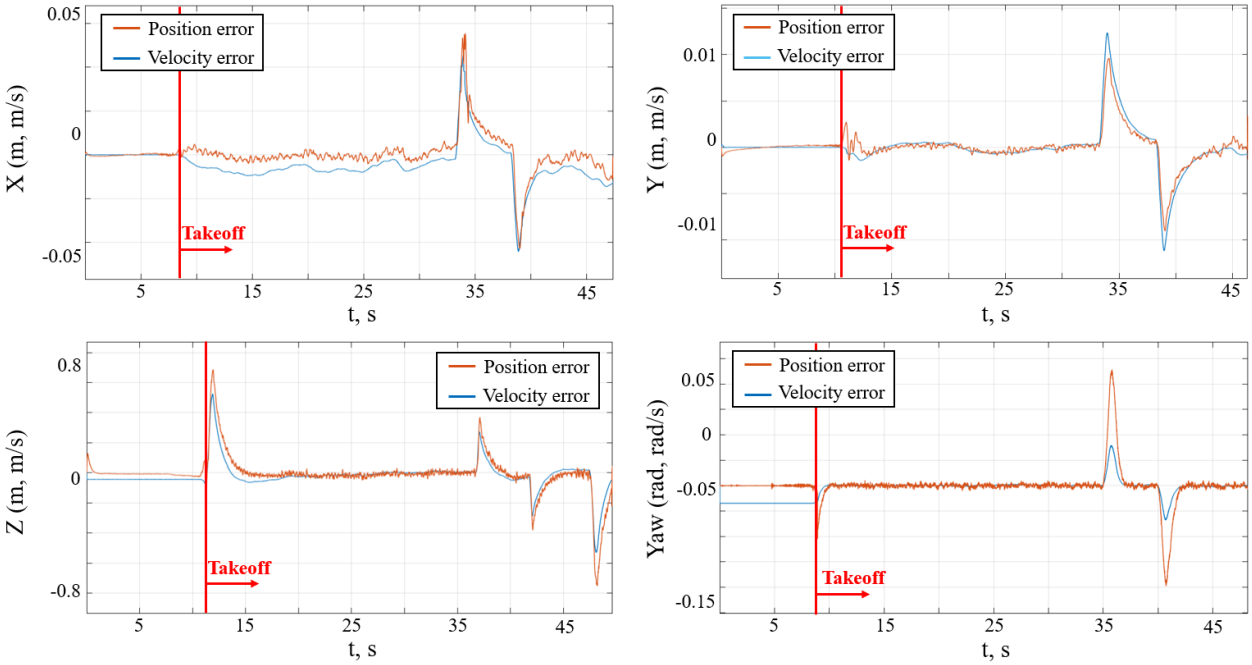


c) Disturbance applied at 45deg angle

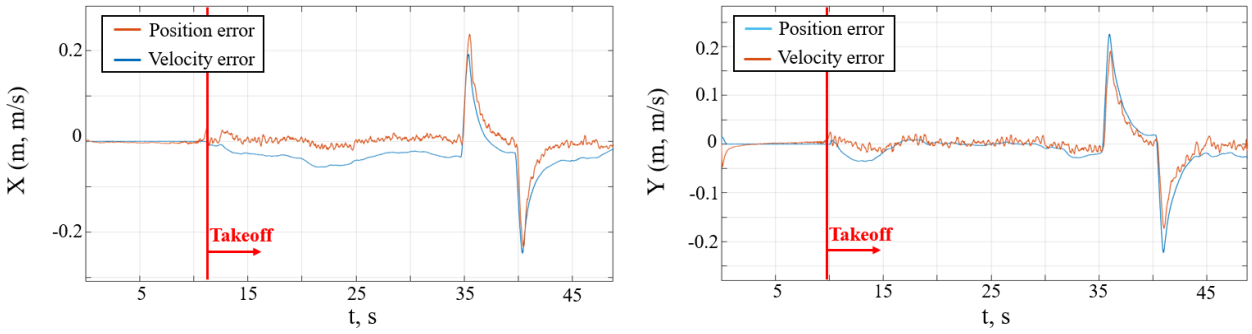
Figure A.4: Unsaturated controller commands vs time (Position control test, PID)



a) Disturbance is not applied

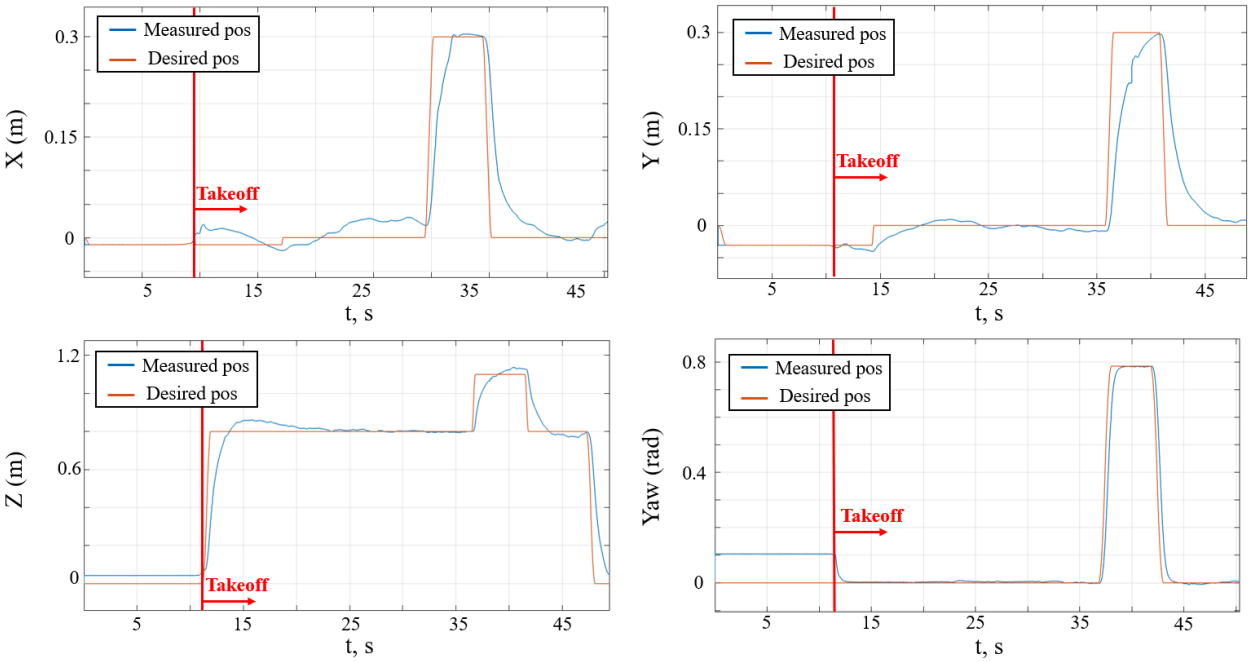


b) Disturbance applied at 0deg angle

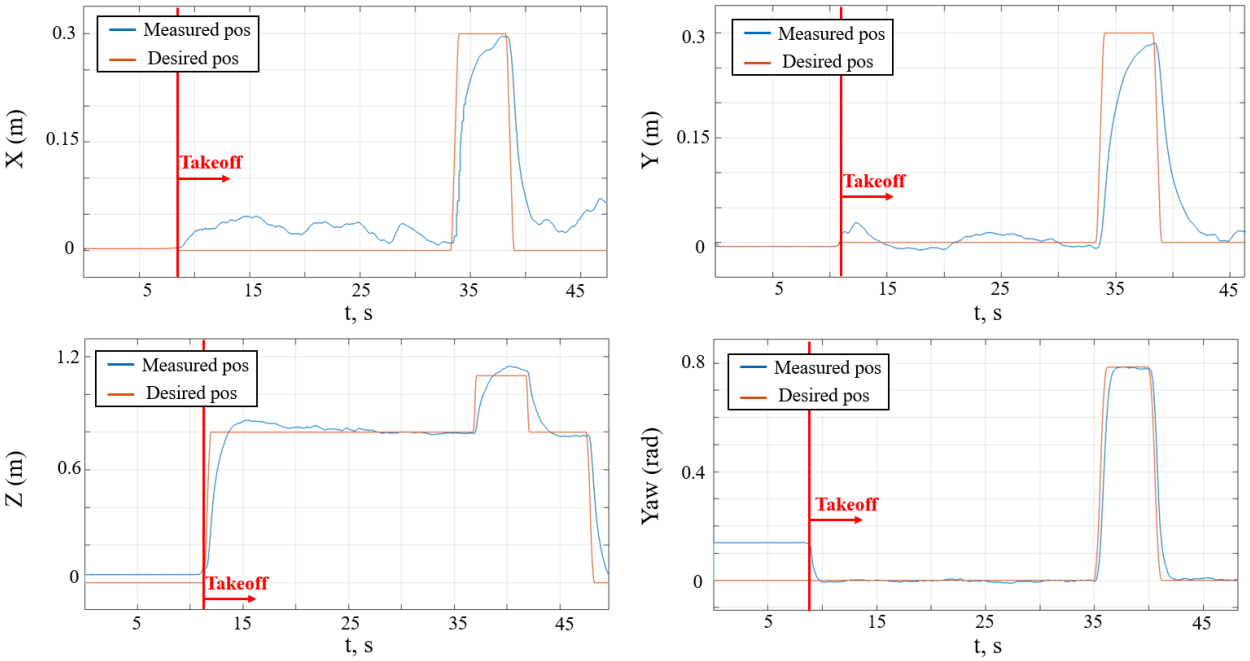


c) Disturbance applied at 45deg angle

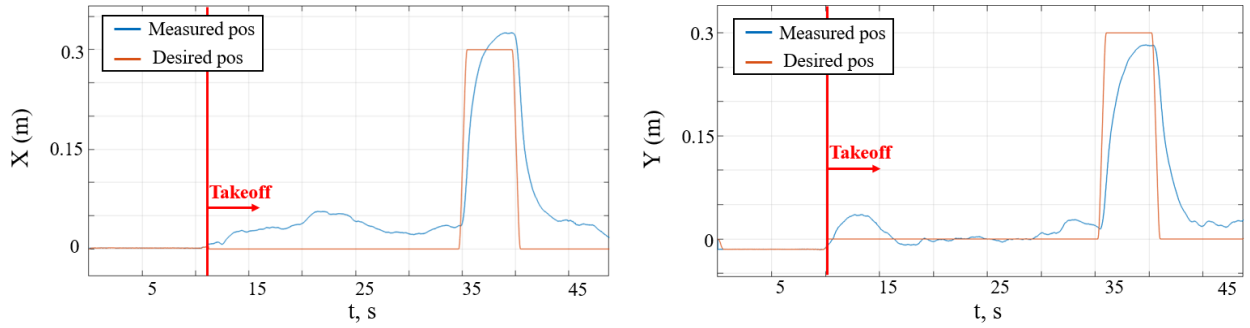
Figure A.5: Position error and Velocity error vs time (Position control test, PID)



a) Disturbance is not applied

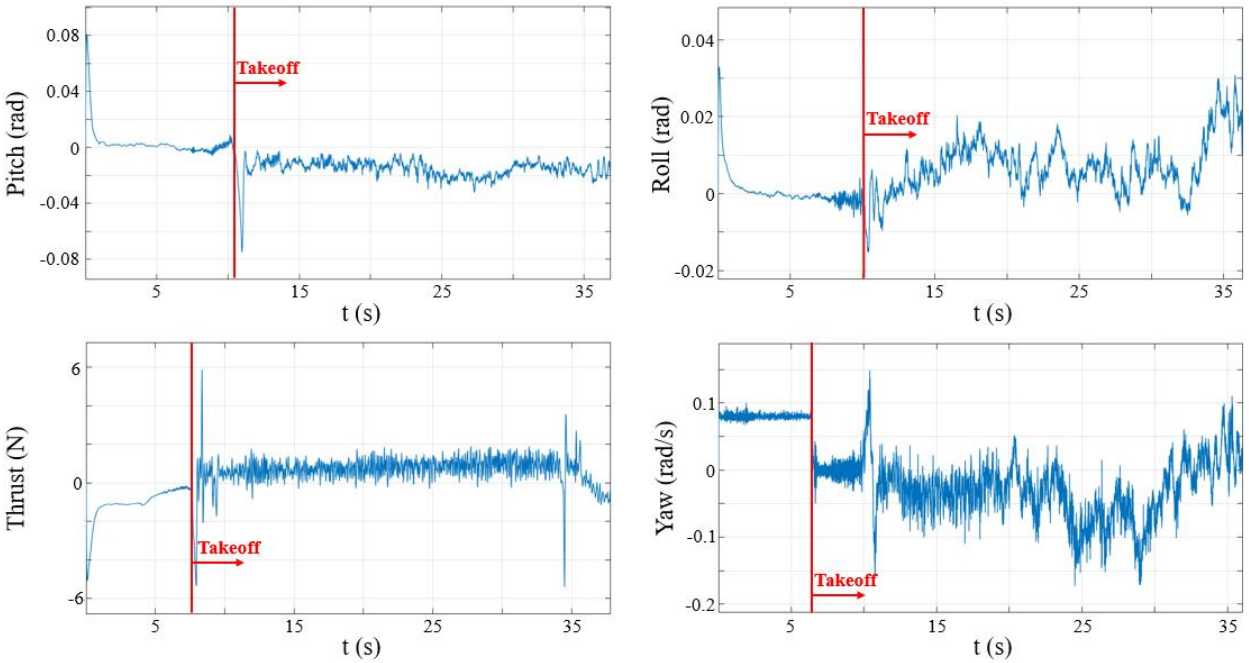


b) Disturbance applied at 0deg angle

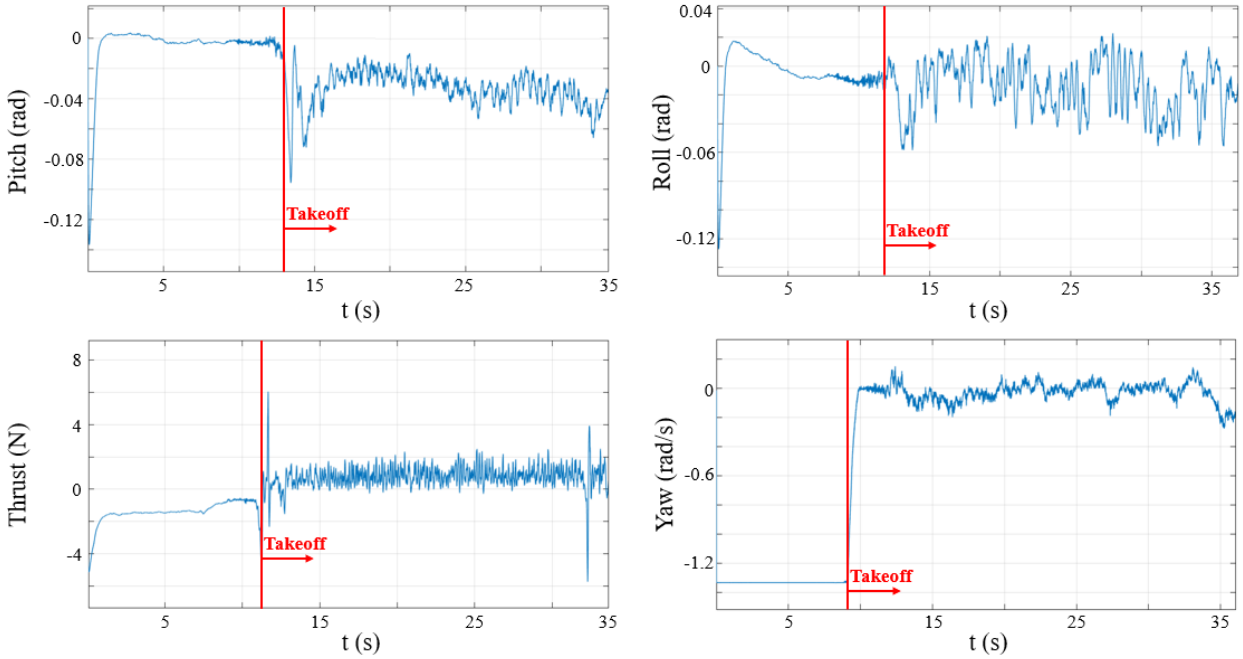


c) Disturbance applied at 45deg angle

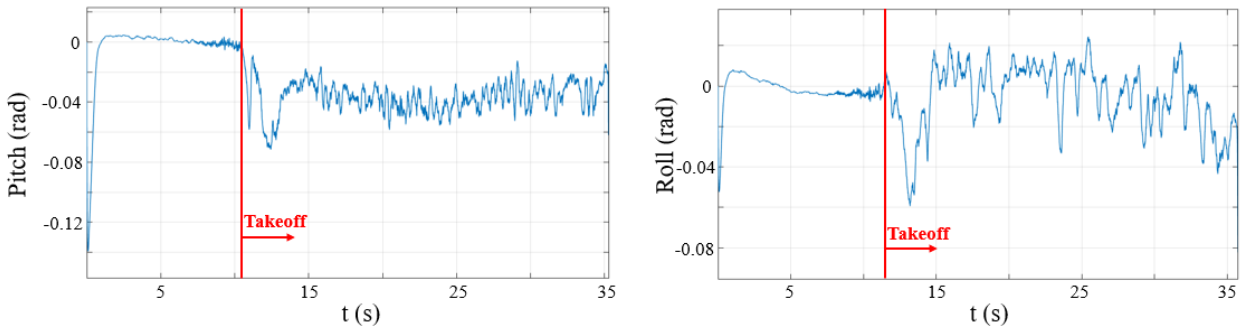
Figure A.6: Position vs time (Position control test, PID)



a) Disturbance is not applied

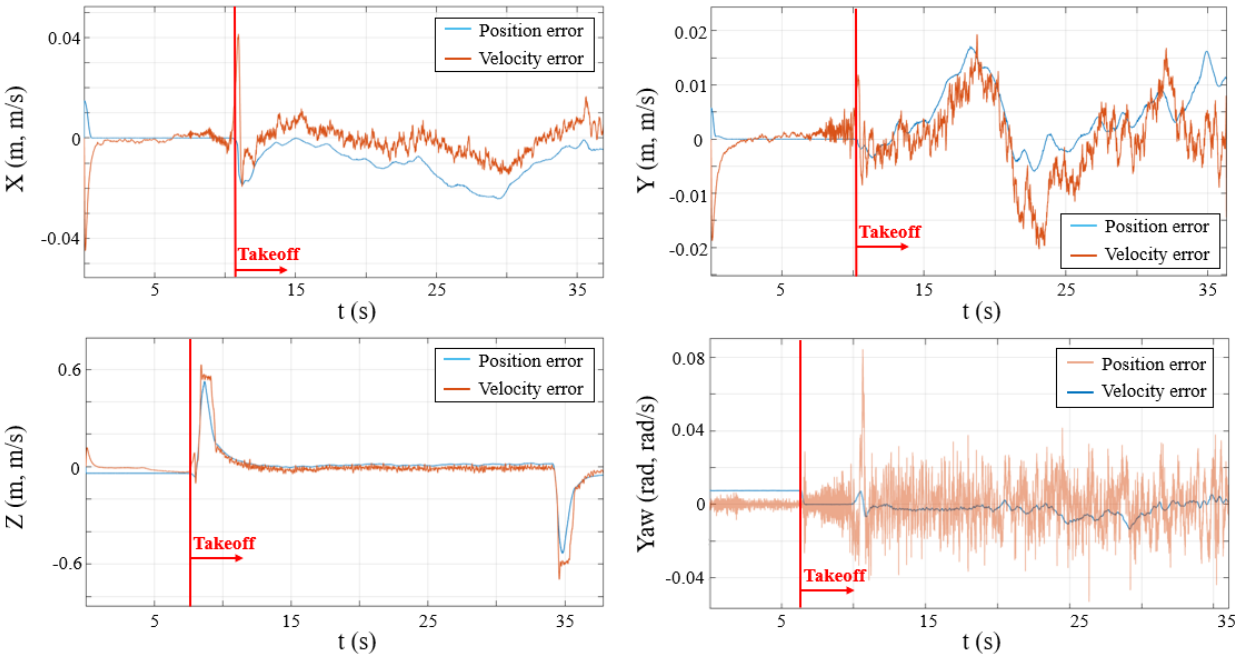


b) Disturbance applied at 0deg angle

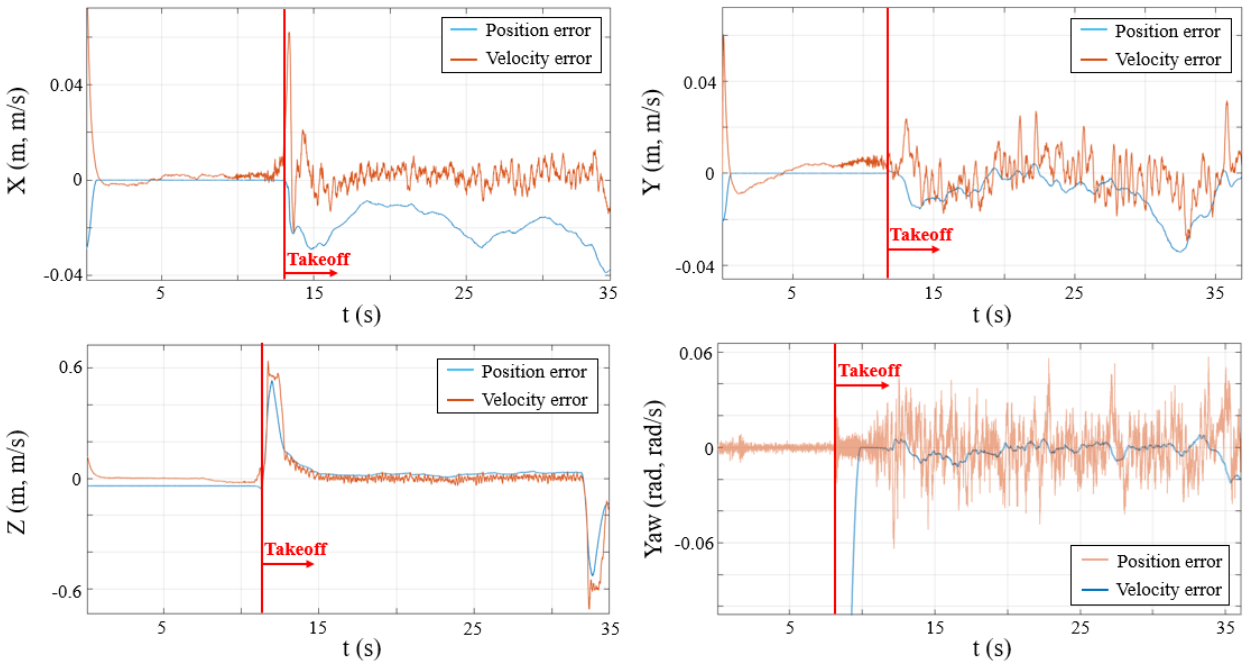


c) Disturbance applied at 45deg angle

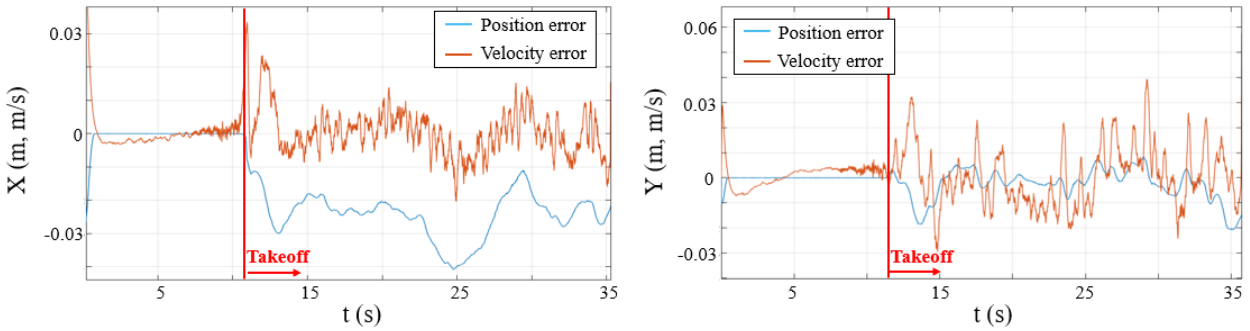
Figure A.7: Unsaturated controller commands vs time (Hover test, Fuzzy)



a) Disturbance is not applied

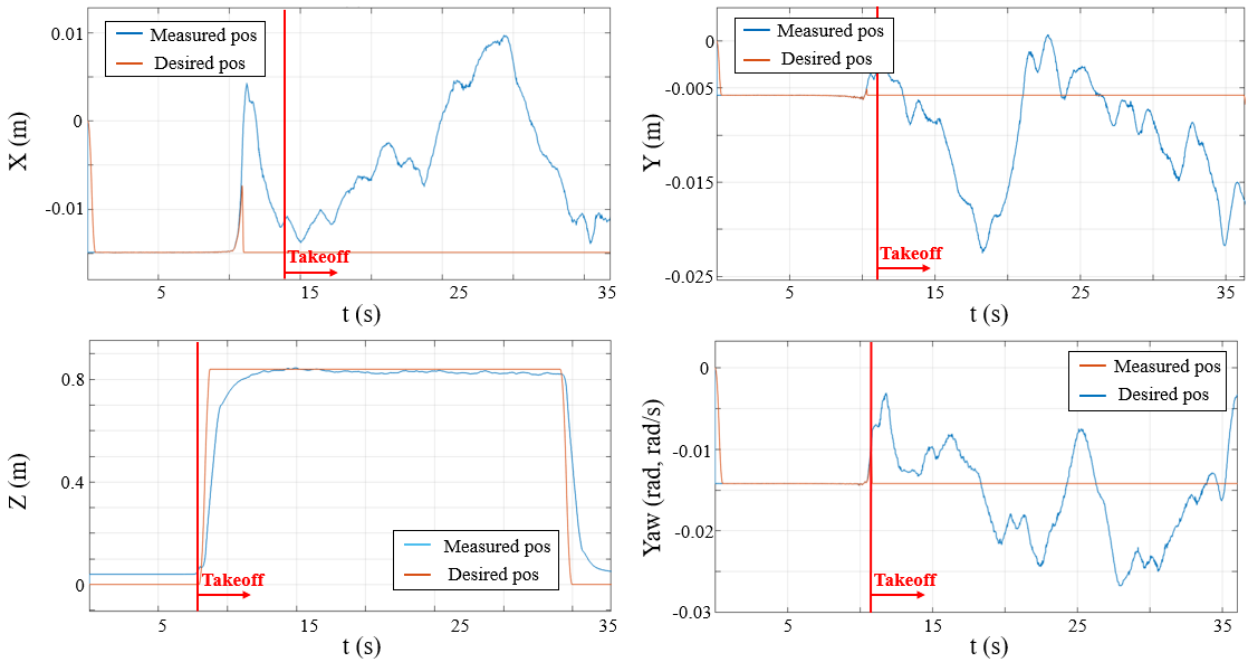


b) Disturbance applied at 0deg angle

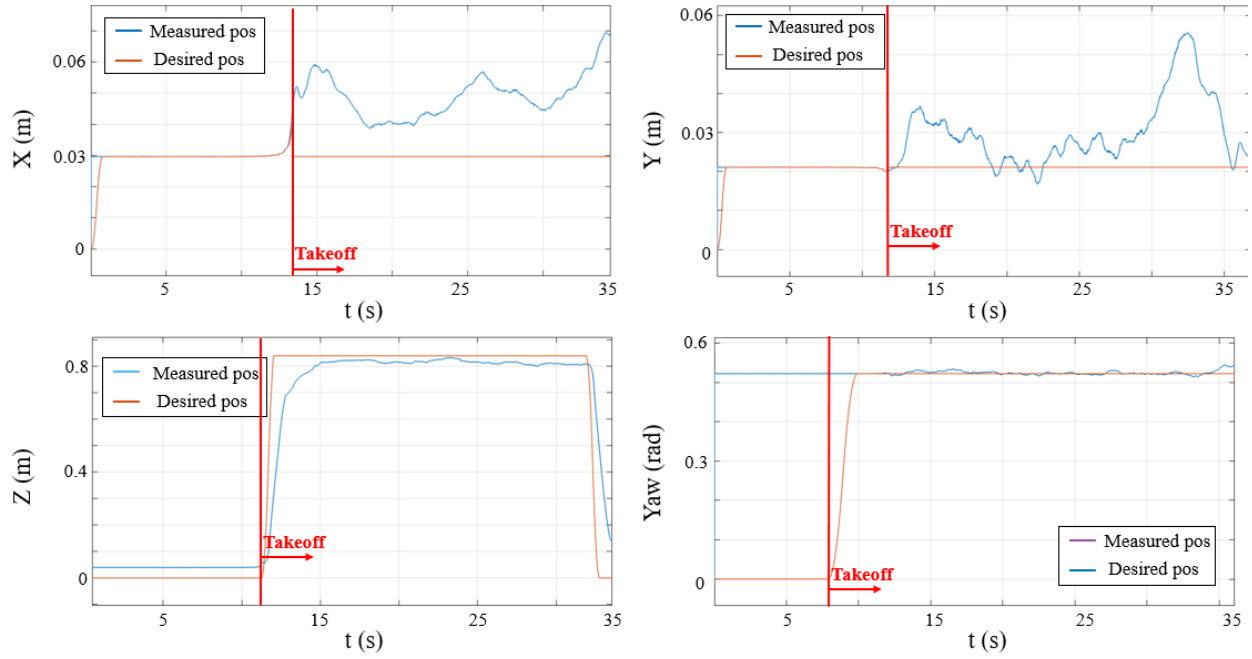


c) Disturbance applied at 45deg angle

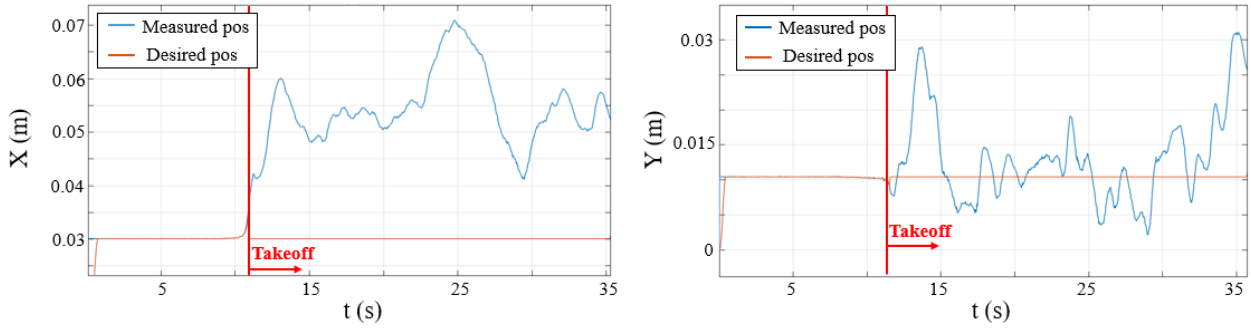
Figure A.8: Position error and Velocity error vs time (Hover test, Fuzzy)



a) Disturbance is not applied

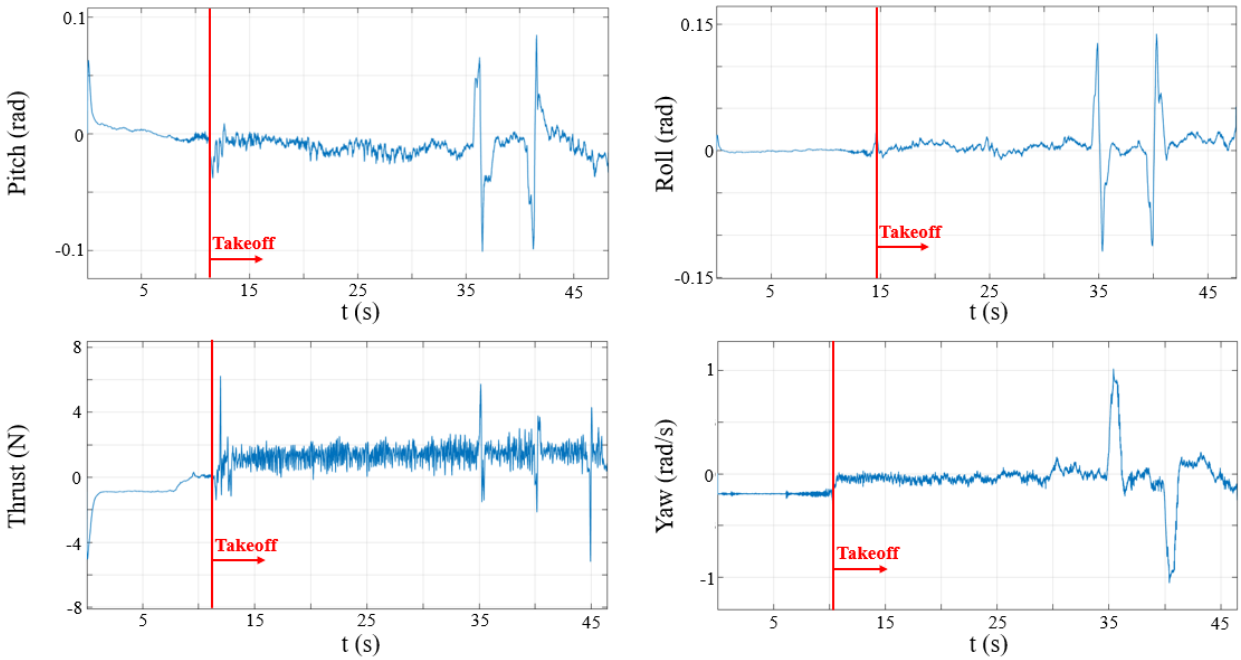


b) Disturbance applied at 0deg angle

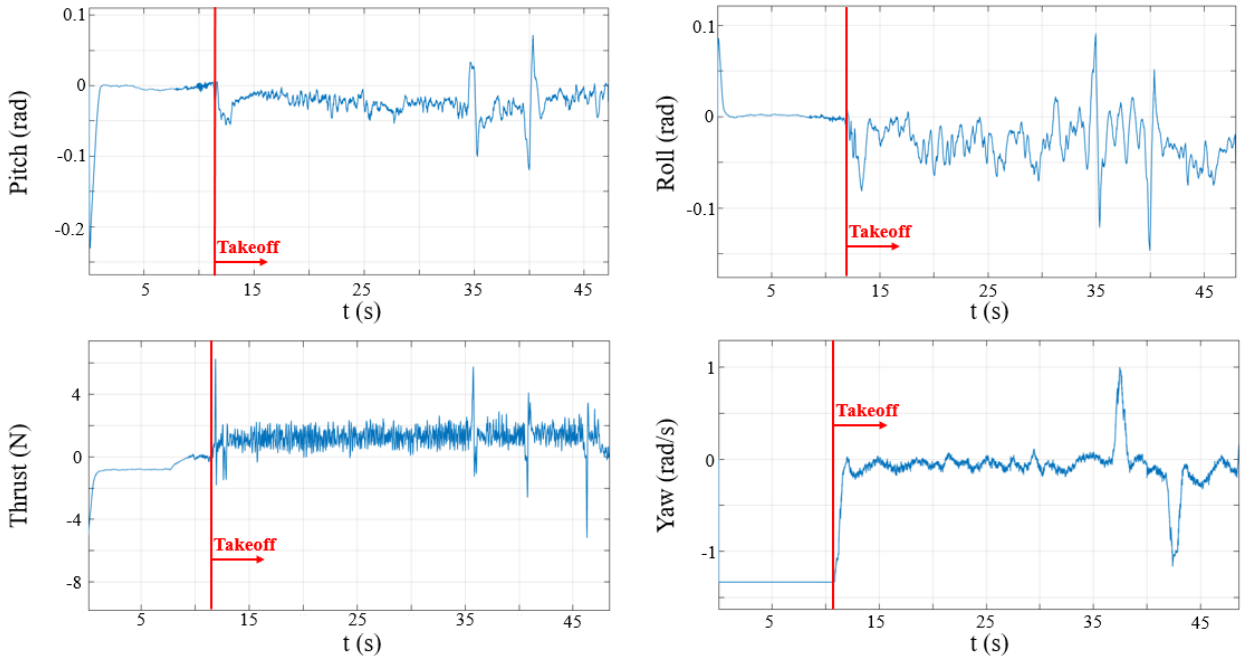


c) Disturbance applied at 45deg angle

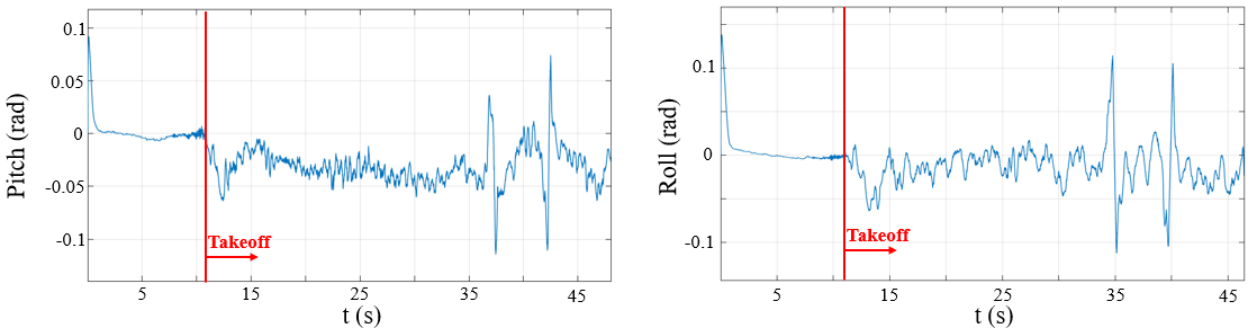
Figure A.9: Position vs time (Hover test, Fuzzy)



a) Disturbance is not applied

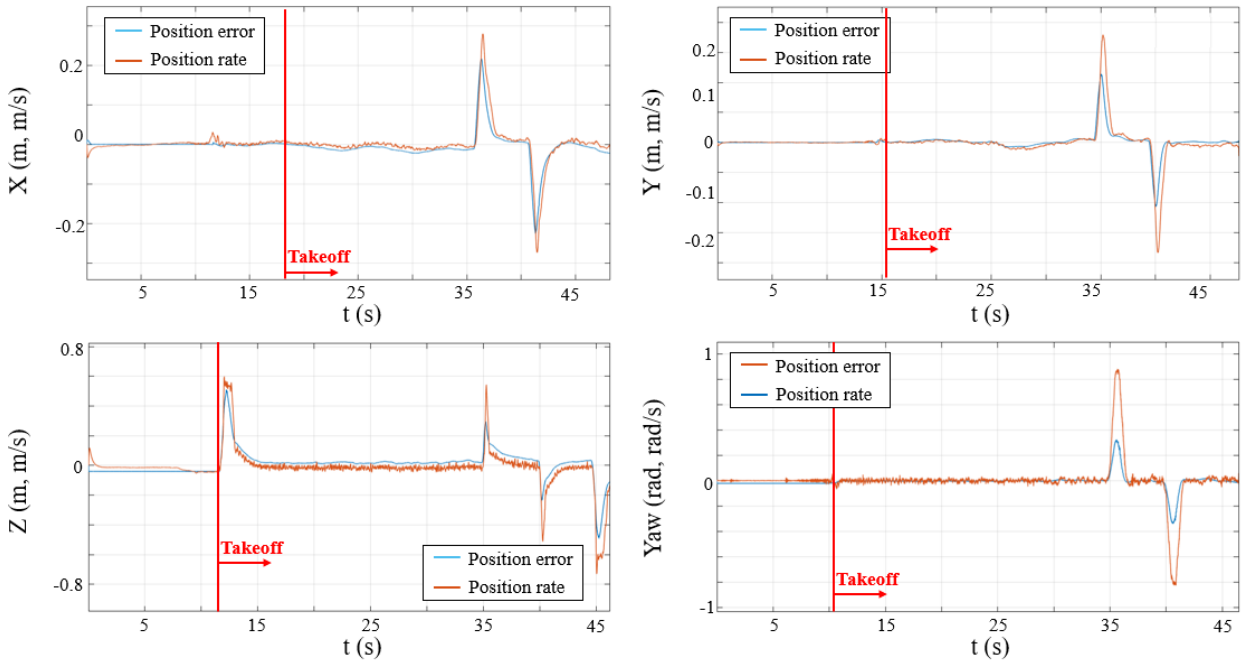


b) Disturbance applied at 0deg angle

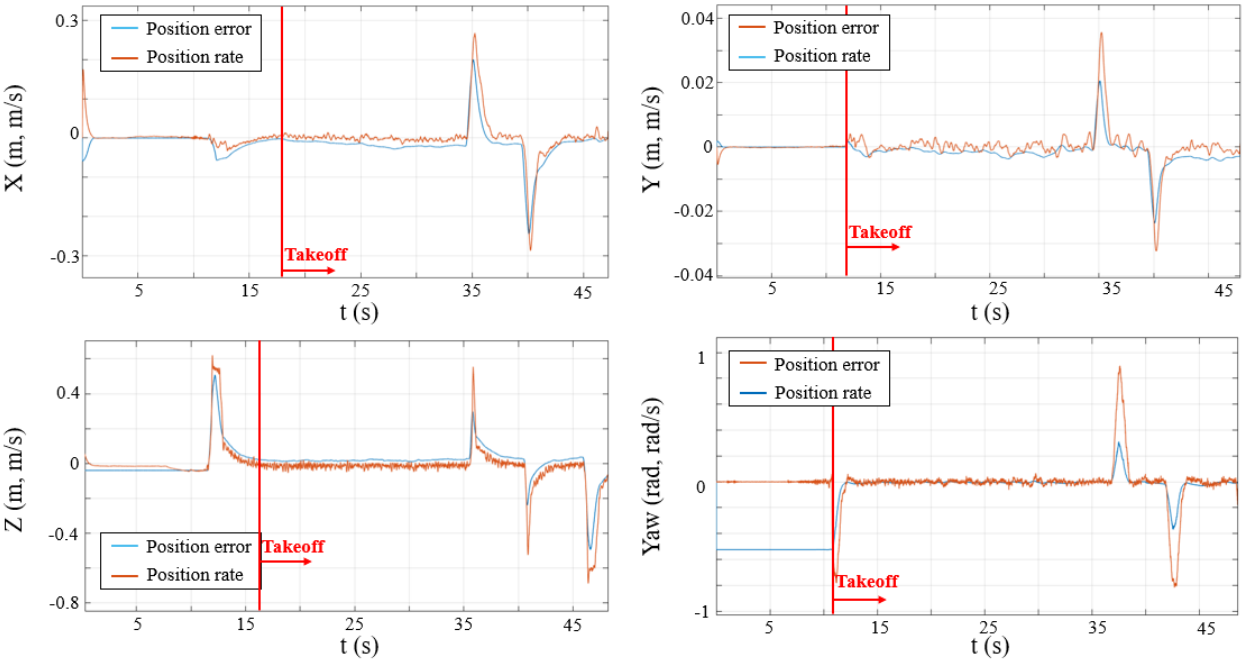


c) Disturbance applied at 45deg angle

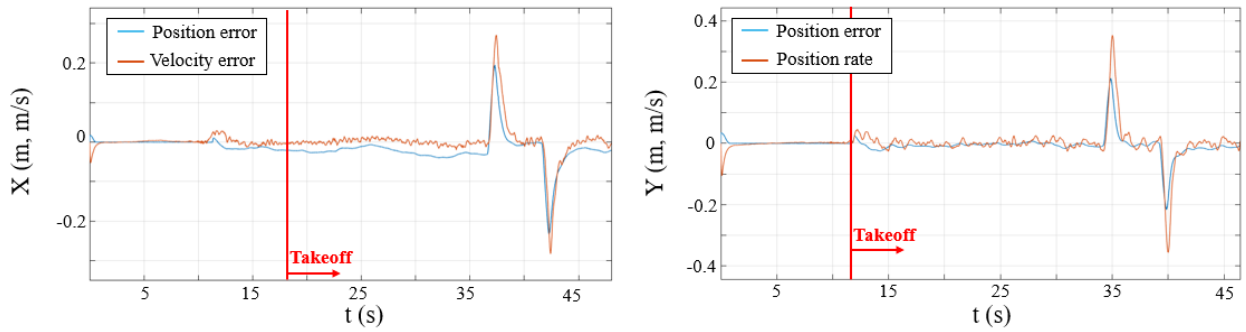
Figure A.10: Unsaturated controller commands vs time (Position control test, Fuzzy)



a) Disturbance is not applied

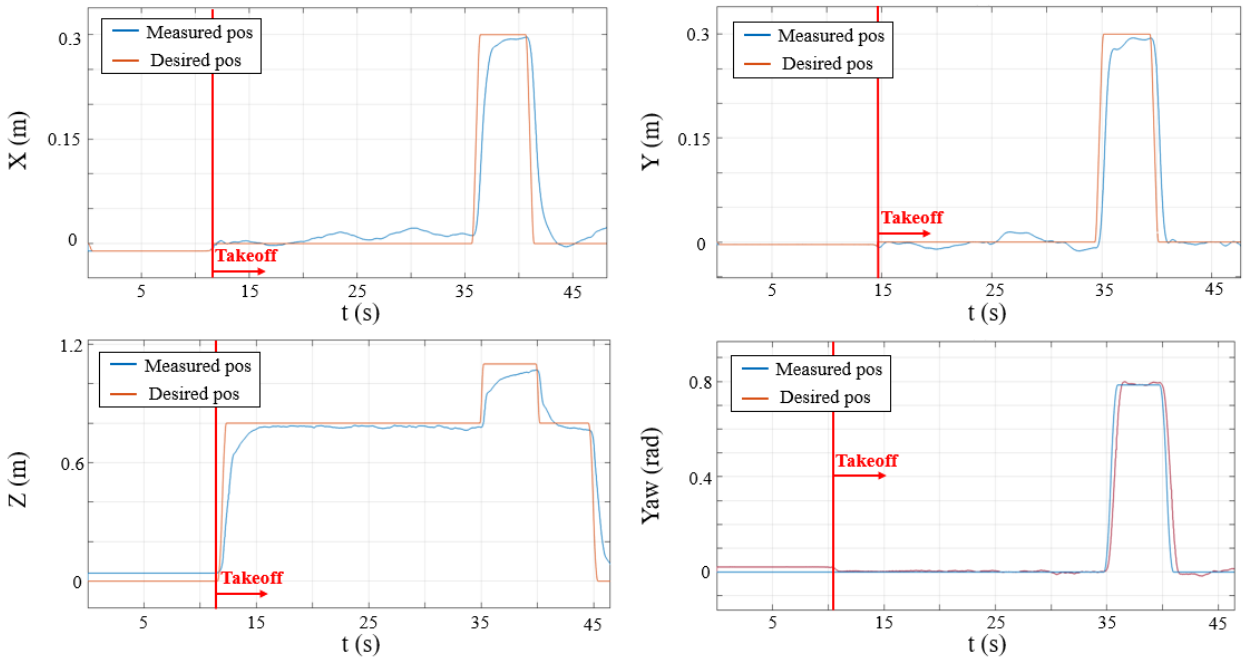


b) Disturbance applied at 0deg angle

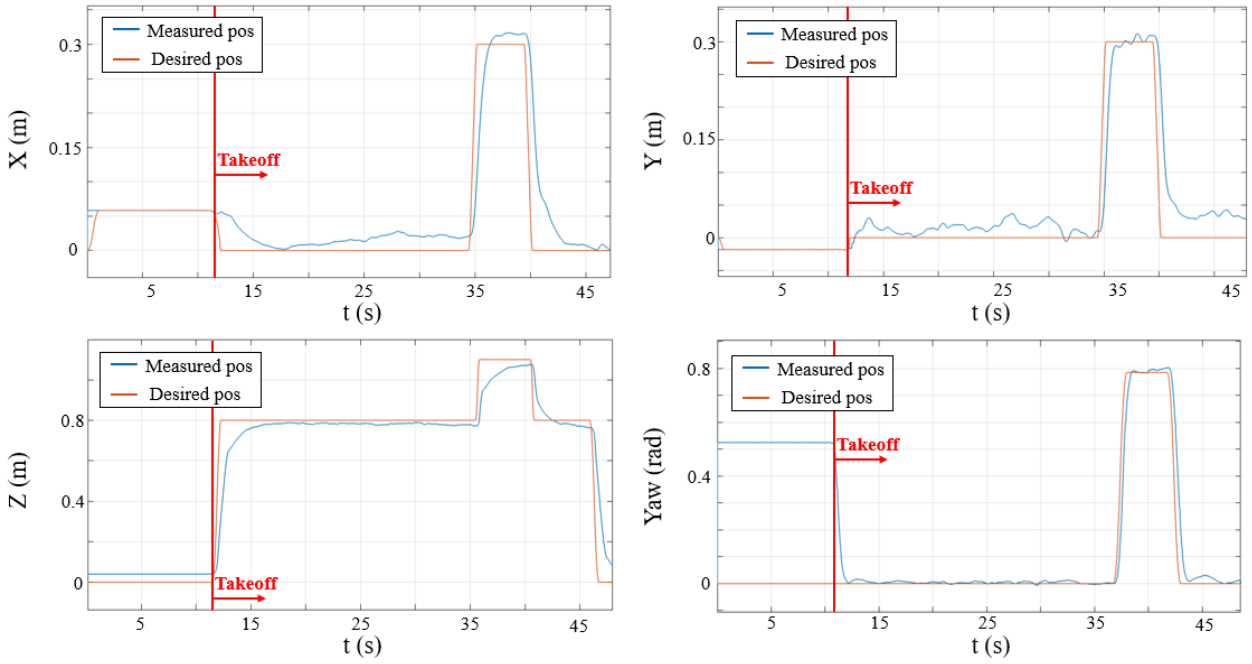


c) Disturbance applied at 45deg angle

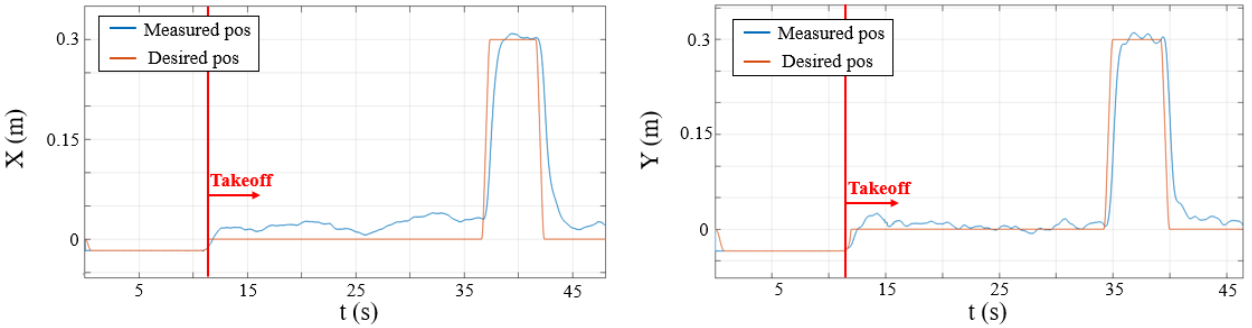
Figure A.11: Position error and Velocity error vs time (Position control test, Fuzzy)



a) Disturbance is not applied

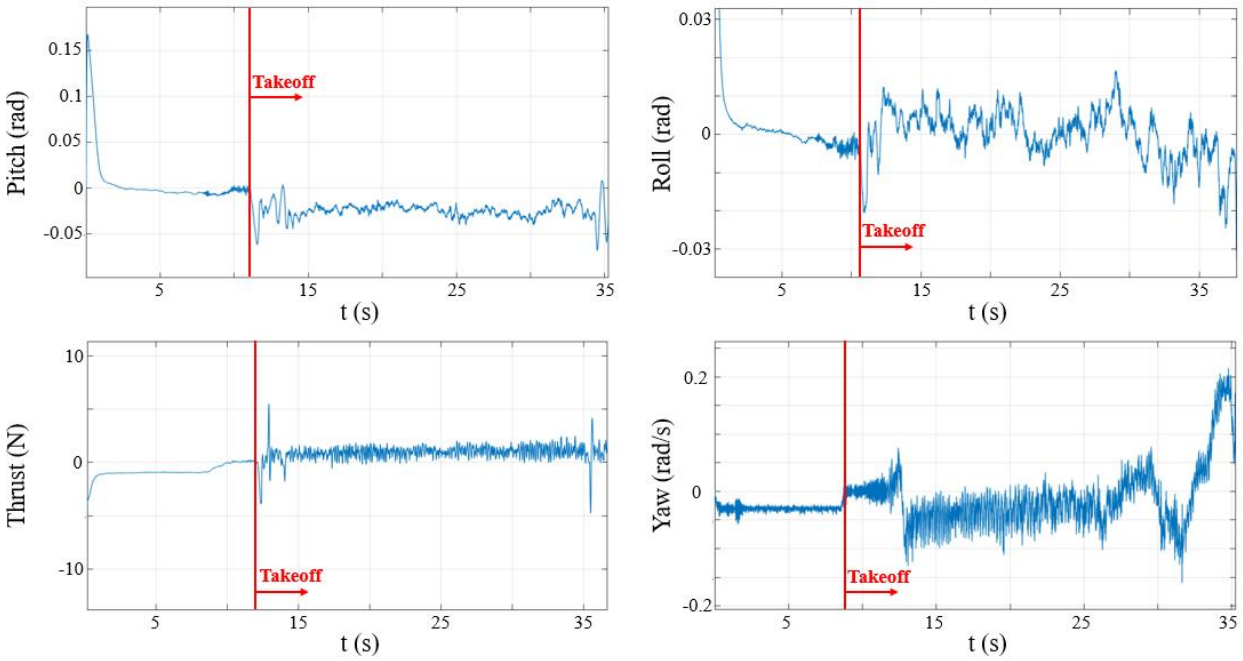


b) Disturbance applied at 0deg angle

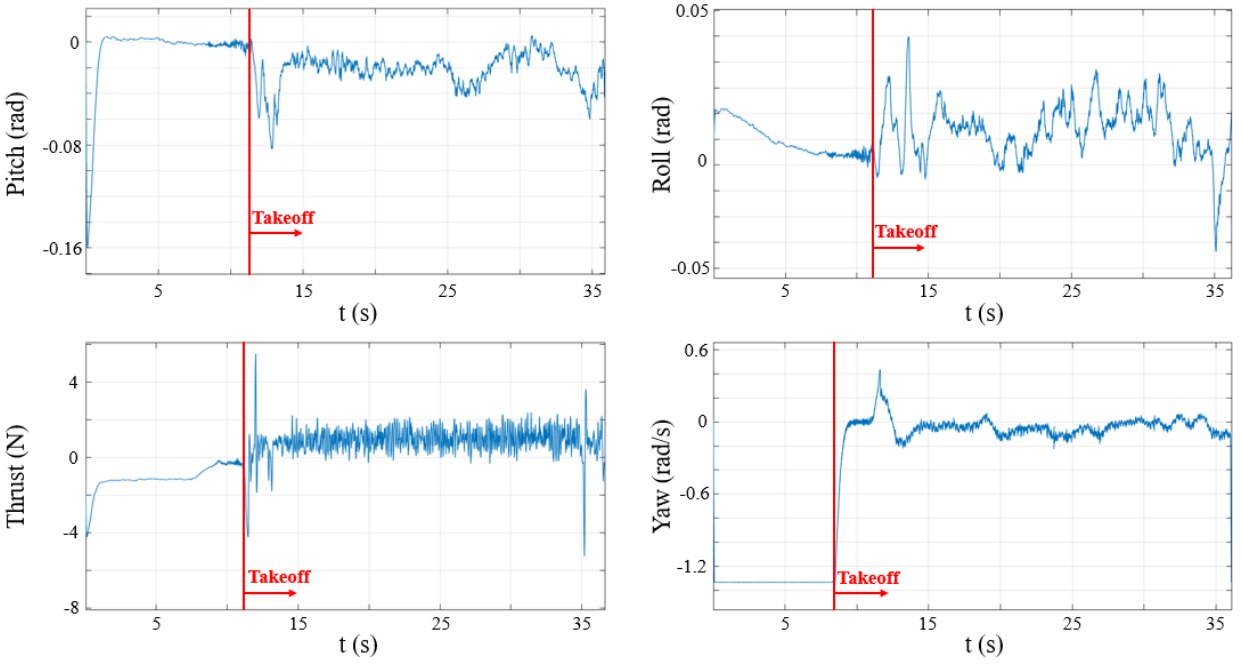


c) Disturbance applied at 45deg angle

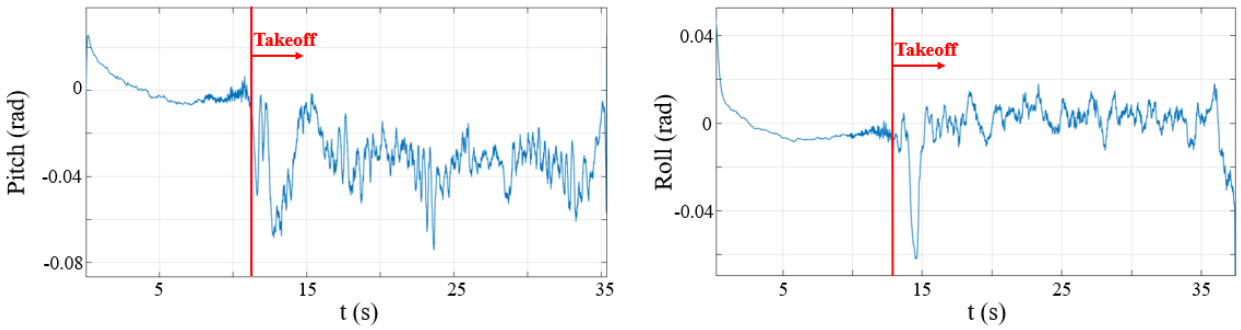
Figure A.12: Position vs time (Position control test, Fuzzy)



a) Disturbance is not applied

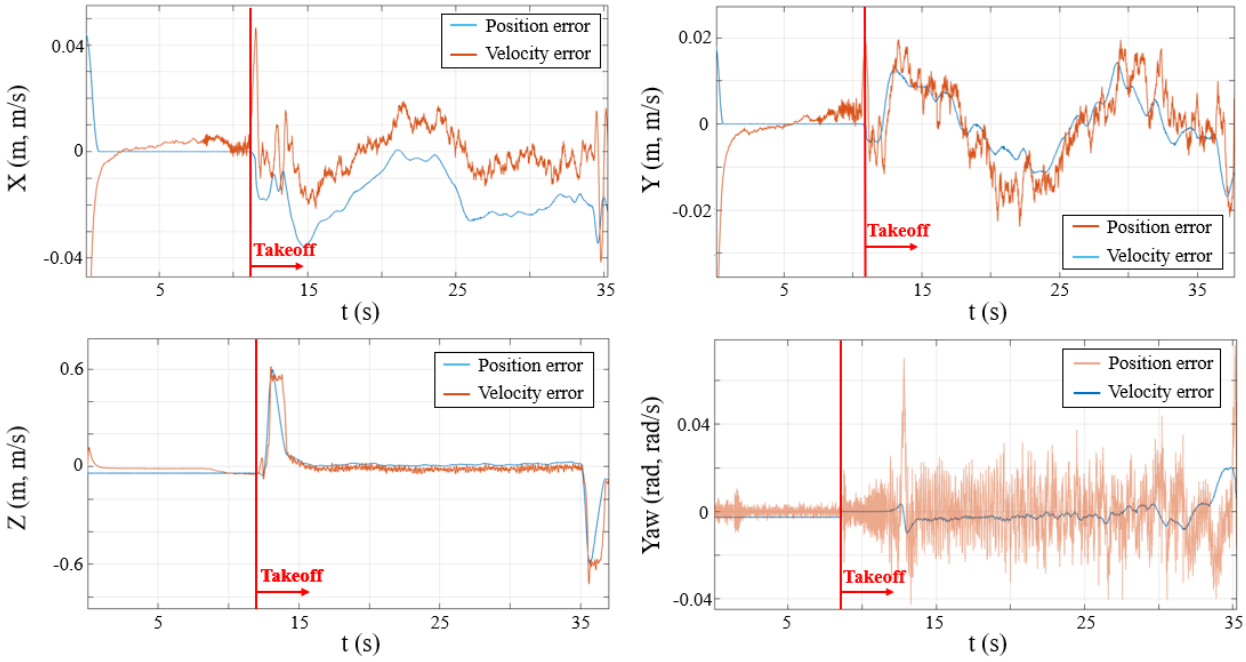


b) Disturbance applied at 0deg angle

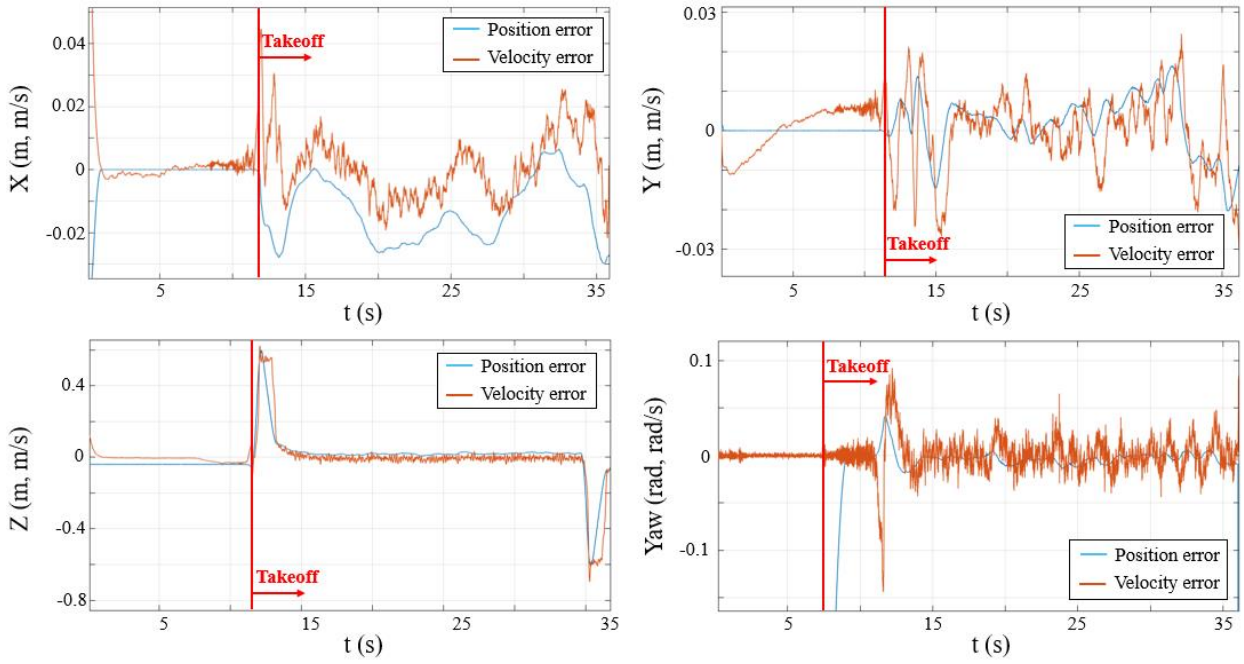


c) Disturbance applied at 45deg angle

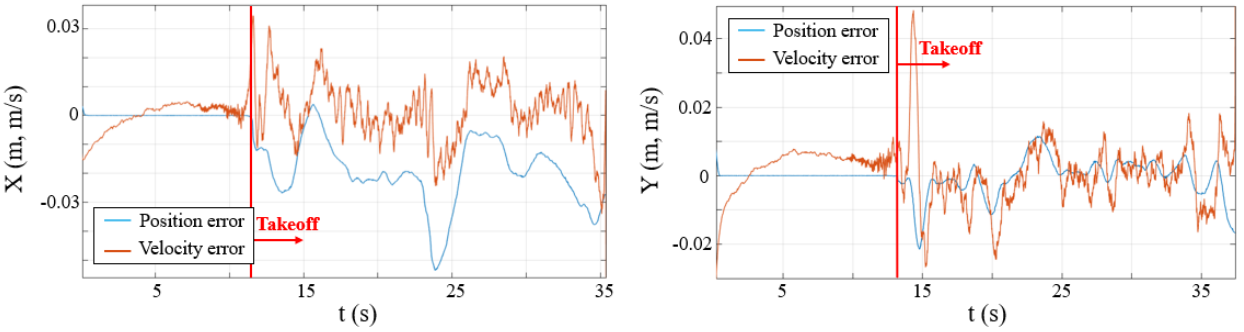
Figure A.13: Unsaturated controller commands vs time (Hover test, Fuzzy controller with prefilter)



a) Disturbance is not applied

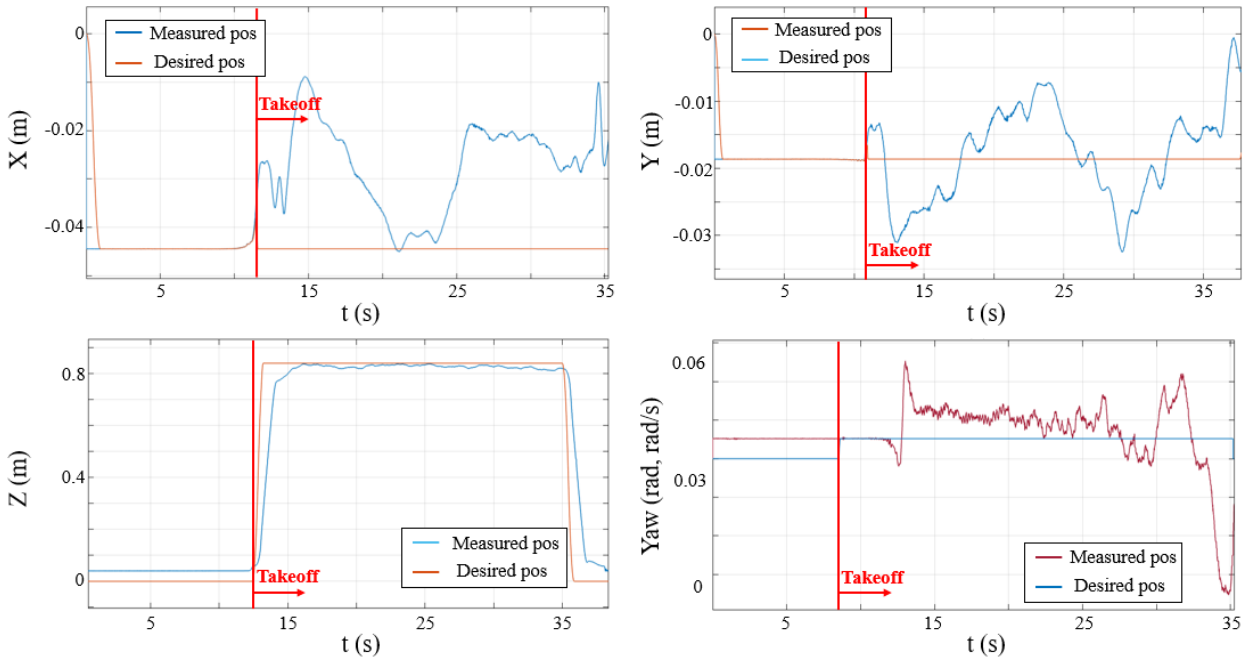


b) Disturbance applied at 0deg angle

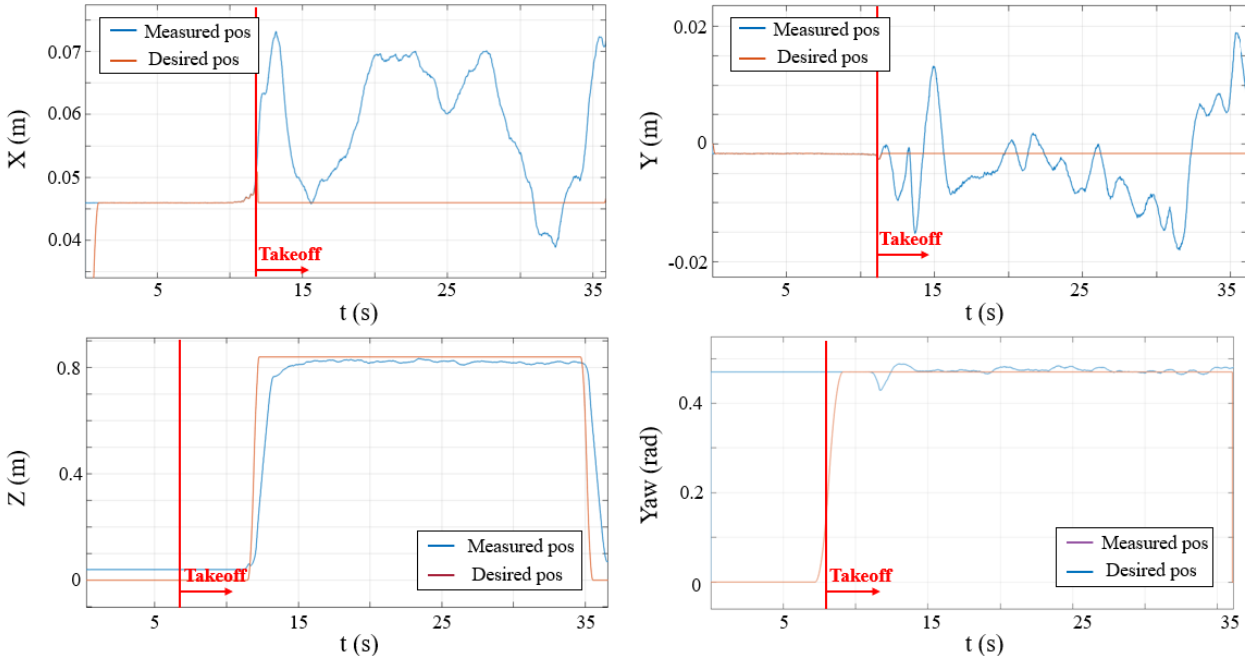


c) Disturbance applied at 45deg angle

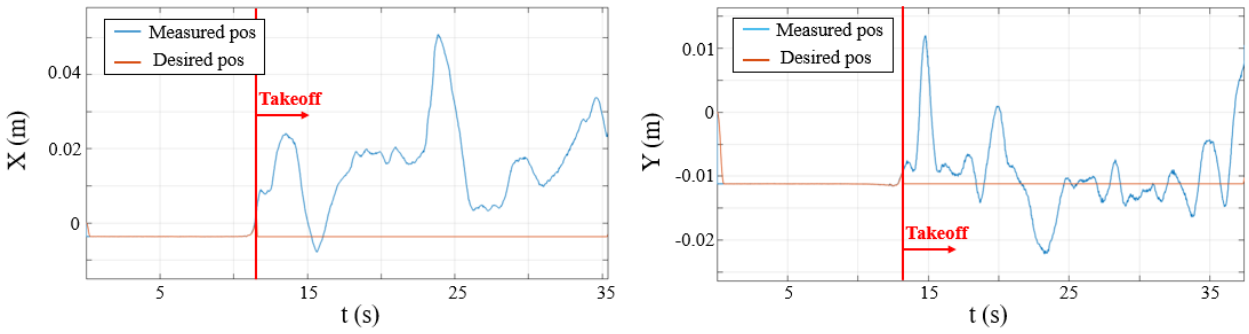
Figure A.14: Position error and Velocity error vs time (Hover test, Fuzzy controller with prefilter)



a) Disturbance is not applied

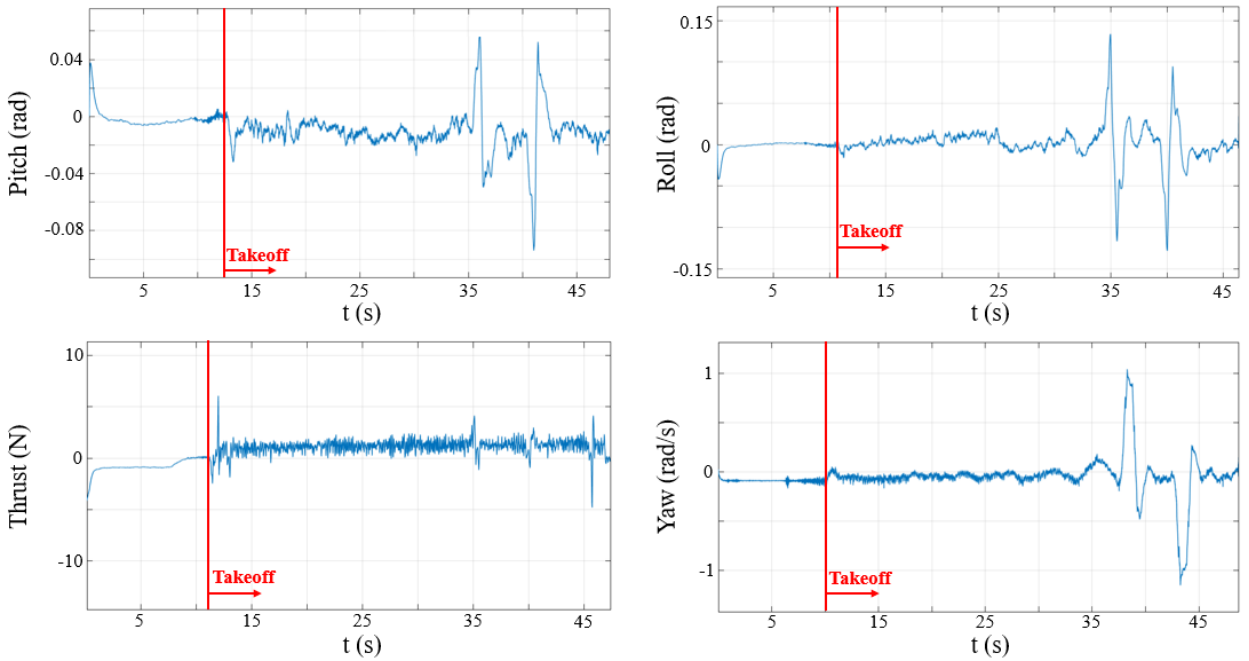


b) Disturbance applied at 0deg angle

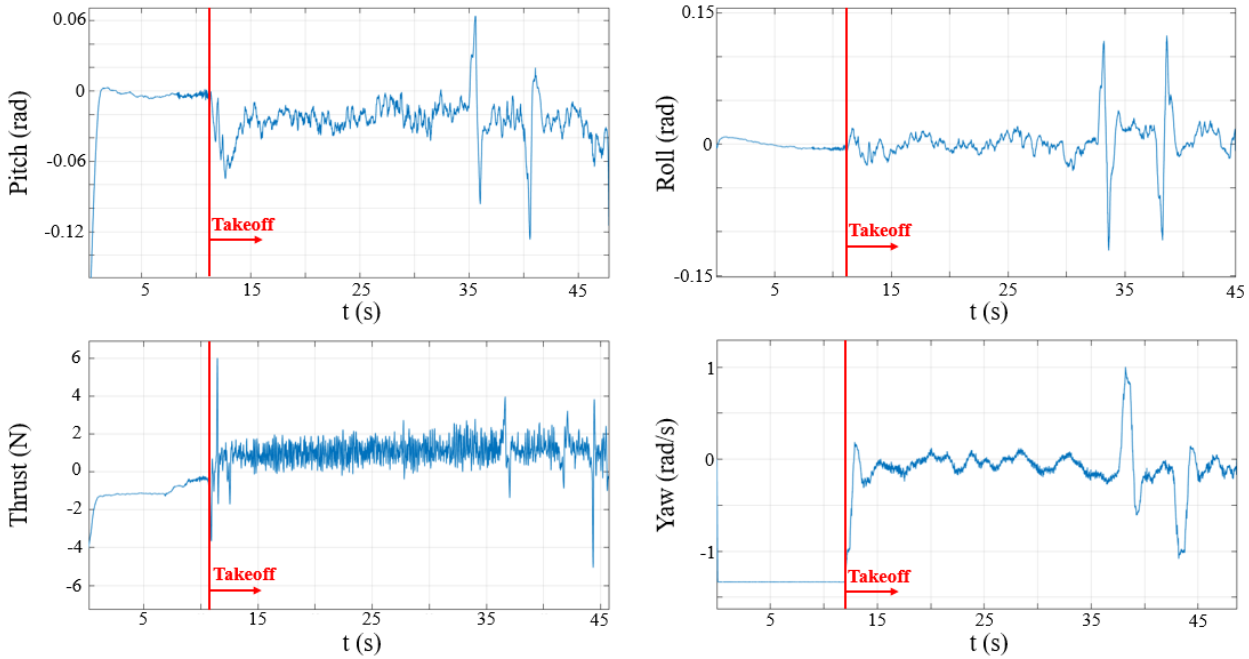


c) Disturbance applied at 45deg angle

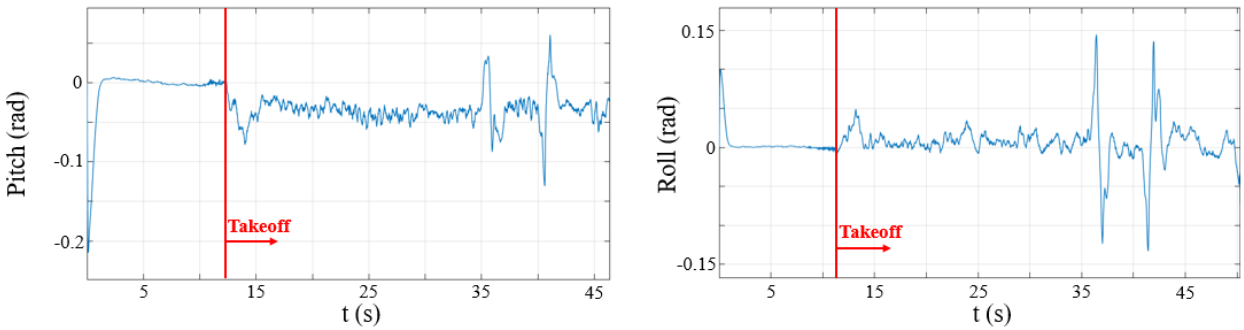
Figure A.15: Position vs time (Hover test, Fuzzy controller with prefilter)



a) Disturbance is not applied

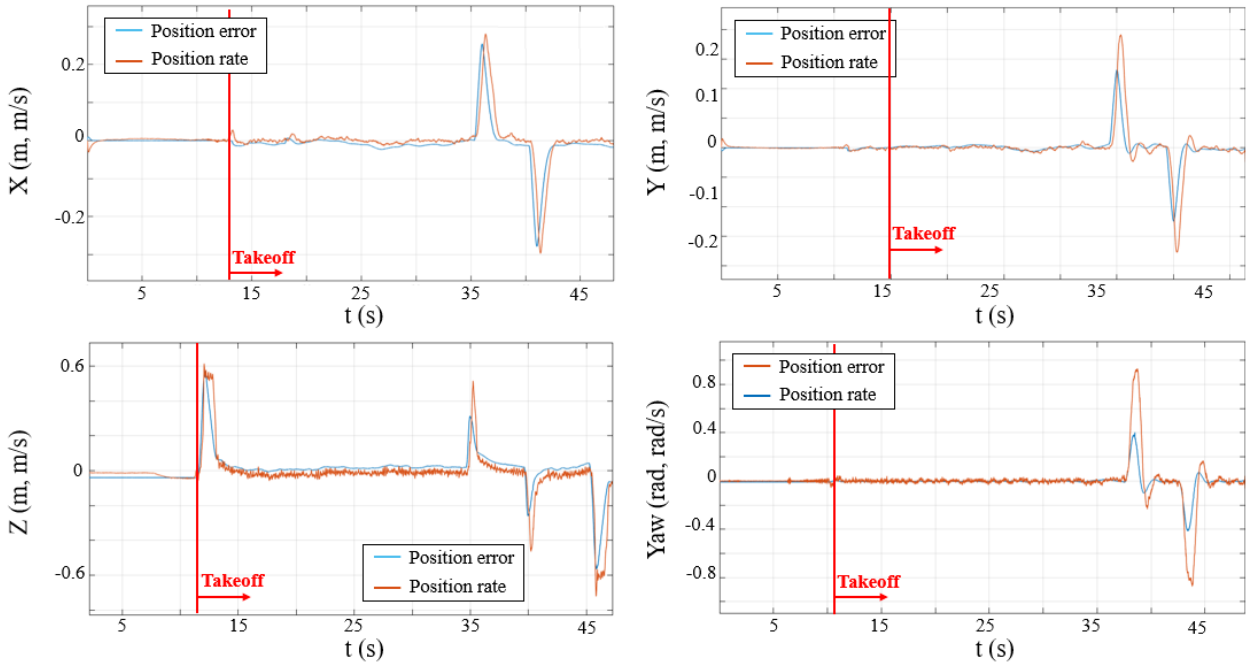


b) Disturbance applied at 0deg angle

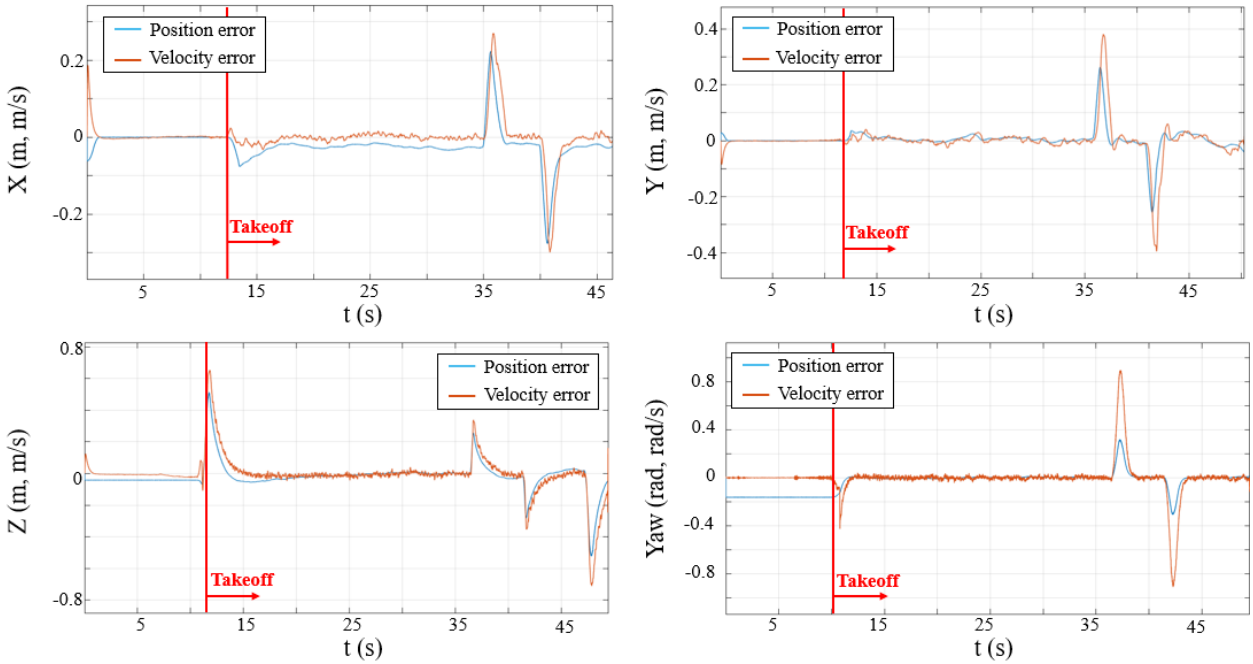


c) Disturbance applied at 45deg angle

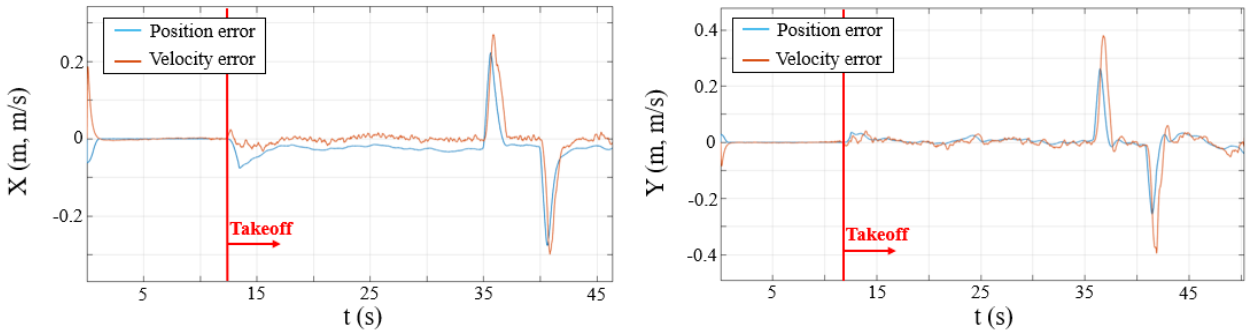
Figure A.16: Unsaturated controller commands vs time (Position control test, Fuzzy controller with prefilter)



a) Disturbance is not applied

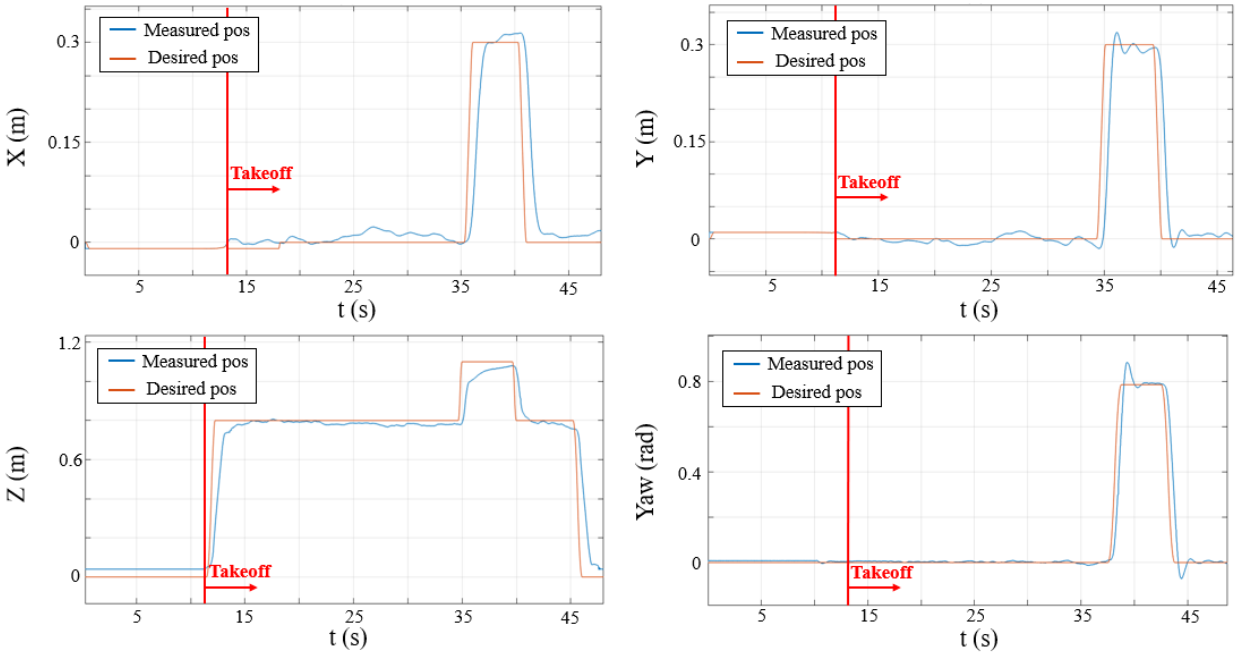


b) Disturbance applied at 0deg angle

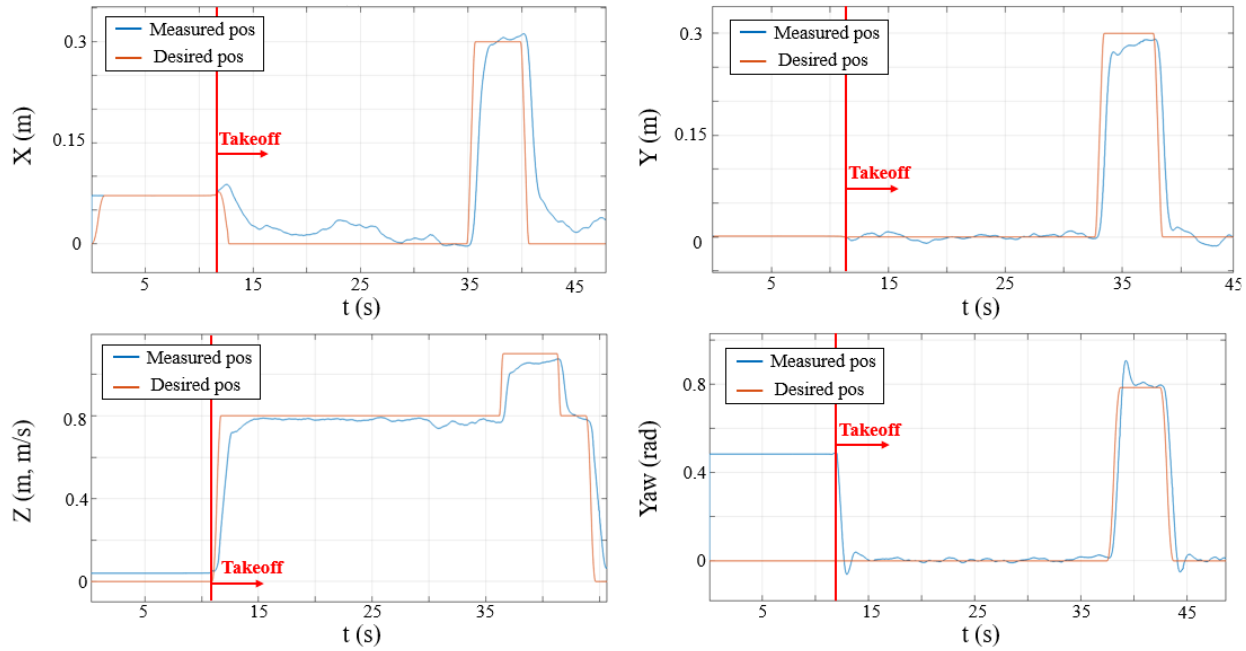


c) Disturbance applied at 45deg angle

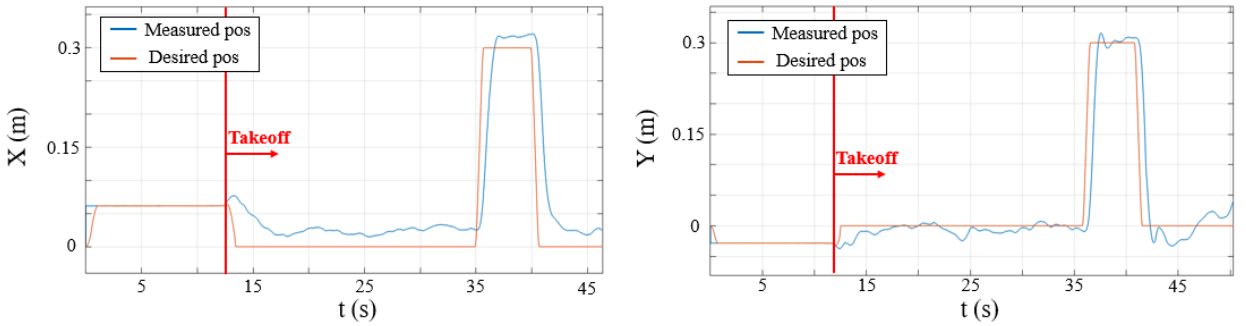
Figure A.17: Position error and Velocity error vs time (Position control test, Fuzzy controller with prefilter)



a) Disturbance is not applied



b) Disturbance applied at 0deg angle



c) Disturbance applied at 45deg angle

Figure A.18: Position vs time (Position control test, Fuzzy controller with prefilter)