CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

UAV Inspections with novel fuzzy-PID control strategy

Ihannah Pinto Guedes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadores:
Milena Faria Pinto
Fabio Augusto de Alcântara Andrade

Rio de Janeiro,
Setembro, 2022

UAV Inspections with novel fuzzy-PID control strategy

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Ihannah Guedes

Ihannah Pinto Guedes

Aprovada por:

Milena Faria Pinto, D.Sc.

(Orientadora, CEFET-RJ)

Fabio Augusto de Alcântara Andrade, Ph.D.

(Orientador, University of South-Eastern Norway)

Alessandro Rosa Lopes Zachi, Dsc.

(CEFET-RJ)

Ana Lucia Ferreira de Barros

Assinado de forma digital por Ana Lucia Ferreira de Barros
Dados: 2022.09.26 11:24:53 -03'00'

Ana Lucia Ferreira de Barros, Dsc.

(CEFET-RJ)

Mario Antônio Ribeiro Dantas, Dsc.

(Universidade Federal de Juiz de Fora)

Rio de Janeiro,

Setembro, 2022

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

G924 Guedes, Ihannah Pinto
UAV Inspections with novel fuzzy-PID control strategy / Ihannah Pinto Guedes – 2022.
xiii, 91f. il. (algumas color.) + apêndice , tabs. ; enc.

Dissertação (Mestrado). Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia : f. 84-91.
Orientadora: Milena Faria Pinto
Orientador : Fabio Augusto de Alcantara Andrade

1. Engenharia de controle e automação. 2. Veículo aéreo não tripulado (VANT). 3. Lógica difusa. 4. Controlador PIDI. Pinto, Milena Faria (Orient.). II. Andrade, Fabio Augusto de Alcantara (Orient.). III. Título.

CDD 629.8

Elaborada pelo bibliotecário Leandro Mota de Menezes CRB-7/5281

ACKNOWLEDGMENTS

I would like to thank my advisors for all their dedication and patience throughout this course.

I also want to thank my classmates, who helped me a lot and provided me with enormous learning, and my friends and family, who understood and supported me every time I needed to be absent due to my research.

To the examining board, for the invitation accepted to evaluate this work.

Ihannah Guedes

"A scientist in his laboratory is not only a technician: he is also a child placed before natural phenomena which impress him like a fairy tale."

Marie Curie

RESUMO

UAV Inspections with novel fuzzy-PID control strategy

Ihannah Guedes

Orientadores:

Prof. Milena Faria Pinto, DSc.

Prof. Fábio Augusto de Alcântara Andrade, PhD.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em nome do programa, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Uma boa prática para manter a performance de sistemas de geração de energia é a manutenção planejada, inclusive em usinas solares, cuja implementação vem crescendo. As inspeções podem ser realizadas através de análises de imagens, principalmente sobrevoando os painéis. Nas última década, houve grandes avanços no uso de Veículos Aéreos Não Tripulados (VANTs) em diversas aplicações, inclusive na inspeção visual de usinas. Este rápido crescimento se deve a sua capacidade de realizar diversos tipos de tarefas com alta flexibilidade alinhada a riscos reduzidos à vida humana e custo-benefício. Portanto, este trabalho aplica um sistema VANT para realizar a inspeção de sistemas fotovoltaicos (PVs) como requisito prévio na atividade inicial do ciclo de manutenção. As crescentes demandas por atribuições mais complexas melhoraram as capacidades dos VANTs, impulsionando o surgimento de plataformas com alto grau de autonomia para realizar tarefas simultâneas com menos intervenção humana. Um dos grandes desafios de manobrar um VANT para um voo estabilizado consiste em um sistema não linear altamente acoplado com dinâmica rápida. Portanto, outra contribuição é o desenvolvimento de um controlador híbrido Fuzzy-PID para controlar a estabilidade de altitude e posição do VANT. Os resultados mostraram que o controlador conseguiu realizar o controle de forma satisfatória, diminuindo em até 10% o overshoot e atingindo tempo de acomodação menor que alguns modelos comparados.

Fluxo óptico, VANTs, Painel Fotovoltaico, Imagens Aéreas, Sistema semi-autônomo.

ABSTRACT

UAV Inspections with novel fuzzy-PID control strategy

Ihannah Guedes

Advisors:

Prof. Milena Faria Pinto, DSc.

Prof. Fábio Augusto de Alcântara Andrade, PhD.

Abstract of Master Dissertation submitted to Electrical Engineering Graduate Program, Federal Center of Tecnologic Education Celso Suckow da Fonseca, CEFET/RJ, as partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

A good practice to maintain the performance of power generation systems is planned maintenance, including in solar plants, whose implementation is growing. Inspections can be carried out through image analysis, mainly by flying over the panels. In the last decade, there have been great advances in the use of Unmanned Aerial Vehicles (UAVs) in various applications, including the visual inspection of power plants. This rapid growth is due to its ability to perform various types of tasks with high flexibility aligned with reduced risks to human life and cost-effectiveness. Therefore, this work applies a Unmanned Aerial Vehicle (UAV) system to perform the inspection of photovoltaic systems Photovoltaic Systems (PVs) as a prerequisite in the initial activity of the maintenance cycle. However, the growing demands for more complex assignments have improved the capabilities of UAVs, driving the emergence of platforms with a high degree of autonomy to perform simultaneous tasks with less human intervention. One of the great challenges is maneuvering a UAV for stabilized flight consists of a highly coupled non-linear system with fast dynamics. Therefore, another contribution is the development of a hybrid Fuzzy-PID controller to control the altitude and position stability of the UAV. The results showed that the controller was able to perform the control satisfactorily, reducing the overshoot by up to 10% and reaching a shorter settling time than some compared models.

Optical Flow, UAVs, Photovoltaic Pannels, Aerial Imagery, Semi-autonomous Systems.

Contents

I Introduction **13**

I.1 Initial Considerations and Work Motivation 13

I.2 Objective 14

I.3 Main Contributions 16

I.4 Papers related to this dissertation 17

I.5 Work Organization 20

II Background and Related Works **21**

II.1 UAV inspection in general 21

II.2 ROS and Gazebo 24

II.3 Optical flow 28

II.4 Stereo Camera 33

II.5 Control 36

II.5.1 PID Cascade 37

II.5.2 Fuzzy system 39

III Proposed Methodology **42**

III.1 Hybrid PID-Fuzzy Controller 42

III.2 Motion Control with Fuzzy Tuning of Cascaded-PID Gains 49

III.3 Fuzzy Gain-Scheduling PID for Position and Altitude Controllers 52

IV Results and Discussion **57**

IV.1 Hybrid PID-Fuzzy Controller 57

IV.2 Motion Control with Fuzzy Tuning of Cascaded-PID Gains 62

IV.3 Fuzzy Gain-Scheduling PID for Position and Altitude Controllers 72

IV.4 Optical Flow simulation 87

V Conclusions and Future Work **93**

List of Tables

II.1 Parameters of the Quadrotor Mathematical Model 37

III.1 Parameters of the Quadrotor Mathematical Model 45
III.2 Proportional-Integral-Derivative (PID) configuration used. 47
III.3 Fuzzy Rule Table from [5]. 51

IV.1 Quantitative data acquired from simulation tests. 57
IV.2 Gain for the altitude control law. 72
IV.3 Performance of the altitude controllers. 72
IV.4 Gain for position control laws. 72
IV.5 Controller performance. 77

List of Figures

II.1 Example of solar farm environment and inspection scheme. 24
II.2 TurtleSim Graphical User Interface (GUI) under the "Understanding ROS Topics" tutorial page. (From: [58]) 25
II.3 Example of rqt_graph usage (From: Author) 26
II.4 Example of extended usages of bag file readings 27
II.5 Example of model and RVIZ utilization. 27
II.6 Gazebo software and web application. 29
II.7 Example of optical flow. 31
II.8 Example of using homography in conjunction with optical flow. 32
II.9 Intel RealSense D435i. 33
II.10 Intel RealSense Software Development Kit (SDK) loading all sensors from D435i. 34
II.11 Quadrotor structure. 36
II.12 Simplified cascade PID controller. 38
II.13 Fuzzy controller proposed by Doitsidis *et al.* 41

III.1 Flowchart of data for this project. 43
III.2 The UAV 3DR IRIS used in the tests. 43
III.3 Interface UAV-Robot Operating System (ROS). 44
III.4 Fuzzy Rules in details. 45
III.5 Fuzzy membership functions. 46
III.6 Illustrative diagram of the predefined trajectories for the tests. 48
III.7 Simulation Environment provided by the software Unreal Engine 4 with AirSim. 49
III.8 Fuzzy membership functions. 51
III.9 Simplified diagram of the control stages. 53
III.10 Fuzzy logic implementation with independent objectives. 53
III.11 Fuzzy surface for proportional gain. 55
III.12 Fuzzy surface for derivative gain. 55
III.13 Fuzzy surface for integral gain. 56
III.14 Fuzzy surface for position control-proportional gain. 56

IV.1 Position and velocity for one row of tests by implementing the PID controller. 59
IV.2 Position and velocity for one row of tests by implementing the Fuzzy-gain controller. . . 60
IV.3 Diagonal motion test on Fuzzy-gain controller. 61
IV.4 The comparison among Cascaded-PID tuning methods. (From: [5]) 62
IV.5 ArduPilot and Proposed method compared. 63
IV.6 Absolute Error comparison between ArduPilot and the proposed method. 63
IV.7 Accumulated Error comparison between ArduPilot and the proposed method. 64
IV.8 Measured errors and Fuzzy PID gains. (From: [5]) 65
IV.9 Measured and Commanded velocities. 67
IV.10 Comparison between the proposed method and using the mean values of the fuzzy logic gains. 68
IV.11 Accumulated Error comparison between the proposed method and using the mean values of the fuzzy logic gains. 68
IV.12 Comparison among different methods in the presence of 15 m/s wind to the West. . . . 69
IV.13 The Parrot Bebop 2. 69
IV.14 The UAV height control flight test. 70
IV.15 Control signal and error during the experimental trial. 70
IV.16 Individual control response during the experiment. 71
IV.17 Altitude response to first control law. 73
IV.18 Altitude response to second control law. 74
IV.19 Input Fuzzification. 75
IV.20 Output DeFuzzification. 76
IV.21 Comparison of the proposed method and PID controller for altitude control. 77
IV.22 Detail of comparison of the proposed method and PID controller for altitude control. . . 78
IV.23 Comparison of the proposed method among others for position control in the X-axis. . . 79
IV.24 Comparison of the proposed method among others for position control in the Y-axis. . . 80
IV.25 3D visualization for the comparison of the proposed method among other approaches for position control. 81
IV.26 UAV 3D path during second simulation tests. 82
IV.27 UAV height response details. 82
IV.28 Quadrotor used in the experimental tests. 83
IV.29 UAV height response for the practical experiment. 83
IV.30 Height response without fuzzy scheduling. 84
IV.31 Height response with fuzzy scheduling. 85
IV.32 Position response of the test with the fuzzy-scheduler. 86

IV.33 Simulation environment in Gazebo. 88
IV.34 Trajectory scheme. 89
IV.35 Simulink scheme. 89
IV.36 Lukas-kanade application over a sample images. 90
IV.37 Movement estimation from optical flow. 91
IV.38 Example result for linear movement. 92
IV.39 Example result for rotation. 92

List of Abbreviations

AIRSIM Aerial Informatics And Robotics Simulation 28, 49
AUV Autonomous Underwater Vehicles 13
BRIEF Binary Robust Independent Elementary Features 22
EKF Extended Kalman Filter 14, 15, 42, 87
FAST Features From Accelerated Segment Test 22
FCU Flight Controller Unit 43, 47, 76, 94
FMU Flight Management Unit 66, 93
GCS Ground Control Station 43, 44
GNSS Global Navigation Satellite System 87
GPS Global Positioning System 14, 21, 22, 24, 69, 76, 79, 87, 94
GUI Graphical User Interface 25, 26
IMU Inertial Measurement Unit 17, 21, 33, 34, 42
ODE Ordinary Differential Equation 87
PD Proportional-Derivative 72, 73
PEM Prediction Error Method 21
PGS Projected Gauss-Seidel 87
PID Proportional-Integral-Derivative . 14, 15, 16, 17, 24, 36, 37, 38, 39, 40, 41, 42, 44, 45, 47, 49, 50, 51, 52, 53, 57, 58, 62, 64, 66, 69, 73, 74, 75, 76, 77, 78, 79, 87, 88, 93, 94
PVS Photovoltaic Systems . . . . .
RGB Red-Green-Blue 22, 33, 34
ROS Robot Operating System 24, 25, 26, 27, 28, 34, 42, 43, 44, 62, 66, 76, 88, 93, 94
RPC Remote Procedure Call 25
SDK Software Development Kit 33, 34
SIFT Scale-Invariant Feature Transform 28
SLAM Simultaneous Localization And Mapping 22, 32
TCP/IP Transmission Control Protocol/Internet Protocol 25
UAV Unmanned Aerial Vehicle 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 28, 32, 33, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 49, 50, 52, 54, 57, 58, 62, 64, 66, 72, 73, 74, 75, 76, 77, 78, 79, 87, 88, 90, 93, 94, 95
UDP User Datagram Protocol 25

VSLAM Visual Simultaneous Localization And Mapping 32
WSL Windows Subsystem For Linux 49

Chapter I Introduction

I.1 Initial Considerations and Work Motivation

Recently, autonomous vehicles and robotics weren't technologies well applied in a big picture scenario. This status is increasingly changing to optimize many tasks that previously were performed by applying other tools. Most of those tools demand human resources to perform it, which opens breaches to unsafe working places, neglection of potential worker's development, or exhaustion of the human workers, by assigning repetitive tasks. In order to exemplify this point, some of the tasks where autonomous unmanned vehicles are implemented, taking into account human safety, are described as follows [8, 61, 37].

In a sub-sea environment, some applications demand highly trained workers, and, despite all the competence, there is a high risk due to the task's nature *per se*. For instance, the inspection of risers that sometimes are now developed by Autonomous Underwater Vehicles (AUV) [28], which operate independently of humans. Another application that could overcome health risks very recently is the usage of autonomous trucks for food supply and other resources. In December 2020, the British supply chain was affected due to the closure of roads in France to try to minimize the Sars-CoV-2's new variant's dissemination [82]. This could have been overcome by not having any humans take the supplies. Some other researchers endorse that autonomous vehicles and trains to reduce the risks of fatal accidents [59].

UAV are gaining a growing interest over the past years due to the possibility of enabling new services that help modernize transportation [9], supply chain [24], search and rescue [63], delivery goods [51, 46], among others. The reason for its use in various applications can be explained by its ability of UAVs to perform complex activities with low-cost flight and to maneuver flexibility [12, 60].

Regarding the inspection task, according to [8], the use of UAV can help to reduce the mission's complexity for data gathering due to its high versatility and the possibility of attaching new technologies to it. Besides, as already mentioned, robotic systems have been applied to many applications to move the traditional processes towards more autonomous operations. In this work, this and some other aspects are also taken in place to use as work motivation.

Nowadays, developed countries are investing in sustainable and alternative energy. Among the clean energy technologies, solar energy has many advantages, such as its decentralized installation, its long life, among others. Note that with the increase in solar farms distribution and installation, efficient and low-cost methods are required for regular inspection to monitor functionality and guarantee performance [1, 41, 64]. In such huge areas to be analyzed, either inland or sea, preventive maintenance is inevitable and highly necessary to keep high efficiency of the energy generation.

In this and in many other scenarios, preventive maintenance must be carried out as often as possible, in the most efficient way, and at the lowest cost to suppliers. It is also imperative that the risk associated with maintenance is reduced and that employee fatigue is minimized. This work aims to implement an efficient solution to improve this type of situation. Notice that how this will be done is described below.

I.2 Objective

Recently, new technologies, such as UAVs equipped with sensors, have been used to substitute human workers inspecting solar farmers to reduce maintenance costs and intervention times [1, 2].

In the present work, three control strategies are developed to provide alternatives to manual control and also give more reliability to the UAV's flight control. In one of them, a Cascaded-PID is combined with a fuzzy-logic controller, which modulates the PID gains in a simulation environment. The second control was developed by changing some designs of the first one, aiming to solve the issues that remained, and enhance ever further the control with the fuzzy-PID also controlling the altitude, but with the focus of the content exclusively on the controllers. The third work endorses the control's enhancement, with another approach of developing a fuzzy-scheduler, which alternates the altitude and position control, depending on the priority of each situation.

Global Positioning System (GPS) can have a localization error varying from 10 to 15 meters [68], because of this, an imaging system was developed to give extra reliability to the position and route that the UAV is performing. In this scenario, the images are collected and enhanced by a homography method, so this data can be properly used. After that, the optical flow takes place using the Lucas-Kanade method, and to merge all the information along with the GPS, the Extended Kalman Filter (EKF) is used. This is further explained in section II, but for a greater perspective, the Optical Flow (along with the other methods) reduces the error from the GPS and the EKF merges all the data to give the corrected value. At the moment of the presentation of this work, the EKF has already been

developed, but not yet merged to the Optical Flow script, which is presented further. The optical flow by itself delivers a good response on what is proposed, but the EKF has proven in the state of art that it has the potential to enhance the results obtained by the other data acquisition methods.

Given the route, the UAV should also be capable of performing this stably. Therefore, there is also great importance in controlling the UAV's movement itself. For example, the authors of [34] have used a self-tuned PID control method to deal with external disturbances in a quadrotor UAV. In [80], the authors have proposed a hybrid PID control strategy to overcome sensor noise and strong wind disturbances. Several works in the literature have been proposed to make UAVs more robust to disturbances, and parametric uncertainties, among other problems. For example, the control method proposed by [85] is a fault-tolerant strategy considering system uncertainties and actuator failures. In [88], the authors have proposed a flight control for a quadrotor UAV for hovering with a slung load attached to it. The mathematical model was simplified to several controllable linear subsystems via reasonable assumptions. A robust H_∞ controller was designed by utilizing the estimated states of a state observer. Other works have proposed robust control techniques for compensating for the effects of external disturbances and uncertainties in the UAV model parameters [42], [40], [22] and [31]. This control stabilization is also discussed further with the developed works.

Regarding flight stabilization, it is widespread to use a navigation system to feed a classic PID controller, which has a simple structure, good stability, and less dependence on the exact system model. Although the PID controller has a simple structure to be implemented, adjusting its parameters deserves attention from the designer, particularly when nonlinearities of the model are present. This is an issue that has been receiving contributions until today. Such dynamical characteristics force the PID design and tuning to become even more complex, which requires an additional control approach. The Fuzzy Logic theory is proposed to deal with systems that are not easy to be modeled because of their nonlinearities. In this sense, many researchers have applied fuzzy controllers to obtain improved performance and robustness compared to those that use pure classical control algorithms [19, 21, 86, 52].

For this, in this work, the hybrid approach between a cascaded-PID and Fuzzy logic controller was implemented in different ways. This model provides to the system the adaptive possibility by being analyzed by the fuzzy logic and also the robustness of a cascaded-PID, its variations and also the fuzzy-scheduler approach, which switches priorities depending on the altitude and position error.

It is also the project's objective to implement not only a stable control but one that adapts to the scenario,which will help in the perception of the desired route. Different simulation and controller

combinations are also described to compare the performance between models and possible simulation issues.

In summary, the work objectives can be pointed as follows:

General Objectives

- Development of a controller capable of stabilizing an UAV flight
- Development of an Optical Flow observer

Specific Objectives

- Development of a Hybrid Fuzzy-Cascaded-PID position controller
- Development of a Hybrid Fuzzy-Cascaded-PID altitude controller
- Development of a Fuzzy scheduler
- Generating visual analyses from optical flow using Lucas-Kanade method
- Obtaining images and IMU data from RealSense Camera
- Way-point development

I.3 Main Contributions

- **Development of a hybrid Fuzzy-PID control structure that enables the UAV to stabilize its attitude when performing an autonomous flight**

  The main idea of this hybrid model of the controller is to create a controller that has the robustness of a PID, but also that could be applied to many different scenarios. PIDs are widely used in the context of machine control and stabilization. However, the values of the Proportional, Integrative, and Derivative variables rely directly on the model itself that will be controlled. It can be very challenging to choose values that will fit the best way in all situations that this kind of robot can use. In order to amplify the range of use for UAVs, the addition of a fuzzy controller was implemented. In this case, the fuzzy would not control the movement speed and position itself, as usually seen in state-of-the-art, but would provide the system's adapted PID gains and develop an optimized new PID controller to the system.

- **Development of a comprehensive georeferencing algorithm for aerial images collected with UAVs**

When capturing aerial images with drones, it is usually desirable to georeference an area of interest. This area of interest can be a tracked object, a defect found during an inspection, among other applications. The process of georeferencing aerial images is challenging due to the many rotations and translations that are needed through the reference frames, such as image, camera, gimbal, UAV, NED, and ENU frames. Some of these rotations are not orthogonal and depend on the Euler angles orientation of one frame concerning the other frame. In order to evaluate the developed comprehensive algorithm, data from a camera equipped with an Inertial Measurement Unit (IMU) was collected. The camera was used to take a picture of a target object that was put on the floor. Its calculated location was compared with its known location.

- **Development of an scheduler for prioritizing the control strategy**

  This work proposes a novel strategy to prioritize the control strategy of the UAV. At the same time that it relies on a combination of a PID and fuzzy controller, it also possesses two fuzzy schedulers, which define the priority of which controller should be used and with which intensity. It dictates not only the PID's gains but considers if the error of the UAV's altitude is inside the acceptable boundaries or if it is too high, providing more power to the systems that will need to have better performance to respond faster towards the altitude compensation. Once the altitude error is smaller, the scheduler compensates and directs the power for the other systems, with the needed percentage for each, so it can also reduce and stabilize the flight not only in altitude but also in position.

I.4 Papers related to this dissertation

Until the presented moment, this research work has generated a report in the form of an international conference paper and two journal papers, such as follows:

- **Published Conference Paper:**

  Guedes, Ihannah, P.; Pinto, Milena, F.;Carvalho, Guilherme ;Zachi, Alessandro, R.;Almeida, Luciana, F.;Andrade, Fabio, A.A.. "Hybrid PID-Fuzzy controller for autonomous UAV stabilization". 14th IEEE International Conference on Industry Applications INDUSCON (2021). [27].

In the last two decades, there have been great advances in using Unmanned Aerial Vehicles (UAVs) in several applications. This rapid growth is due to their ability to carry out several types of tasks with high flexibility aligned with reduced risks to human life and cost-effectiveness. The increasing demands for more complex assignments have improved UAVs' capabilities, propelling the rise of platforms with a high degree of autonomy for performing simultaneous tasks with less human intervention. One of the great challenges of maneuvering a UAV for a stabilized flight consists of a highly-coupled nonlinear system with fast dynamics. Thus, attitude control is an essential issue for stabilizing the vehicle or for keeping it in the desired orientation. The PID controller is often used due to its simple structure, good stability properties, and less dependence on the exact system model. However, its tuning process may become complicated in the face of the system nonlinearities. In this scenario, an adjustment error can cause it to lose its flight stability. Therefore, the main contribution of this work is the development of a Fuzzy-PID hybrid controller to control a quadrotor UAV's height stability. The work also considers a classic PID controller for comparison purposes with the proposed hybrid controller. The ROS/Gazebo platform was used to carry out the experiments with both control structures. From the results, it was possible to verify that both PID and Fuzzy-PID controllers could perform the attitude control of the UAV. However, the hybrid control strategy obtained some advantages, such as self-adjustment through system variations.

- **Published Article Paper:** Melo, A. G., Andrade, F. A., Guedes, I. P., Carvalho, G. F., Zachi, A. R., & Pinto, M. F. (2022). Fuzzy Gain-Scheduling PID for UAV Position and Altitude Controllers. Sensors, 22(6), 2173. Available at: [49].

Unmanned aerial vehicle (UAV) applications have evolved to a wide range of fields in the last decade. One of the main challenges in autonomous tasks is the UAV stability during maneuvers. Thus, attitude and position control play a crucial role in stabilizing the vehicle in the desired orientation and path. Many control techniques have been developed for this. However, proportional integral derivative (PID) controllers are often used due their structure and efficiency. Despite PIDs good performance, different requirements may be present at different mission stages. The main contribution of this research work is the development of a novel strategy based on a fuzzy-gain scheduling mechanism to adjust the PID controller to stabilize both position and altitude. This control strategy must be effective, simple, and robust to uncertainties and external disturbances. The Robot Operating System (ROS) integrates the proposed system and the flight control unit. The obtained results showed that the proposed approach was successfully applied to the trajectory tracking and revealed a good performance compared to conventional PID and in the presence of noises. In the tests, the position controller was only affected when the altitude error was higher, with an error of 2% lower.

- **Published Article Paper:** Andrade, F. A., Guedes, I. P., Carvalho, G. F., Zachi, A. R., Haddad, D. B., Almeida, L. F., ... & Pinto, M. F. (2021). Unmanned Aerial Vehicles Motion Control with Fuzzy Tuning of Cascaded-PID Gains. Machines, 10(1), 12. Available at: [5].

  One of the main challenges of maneuvering an Unmanned Aerial Vehicle (UAV) to keep a stabilized flight is dealing with its fast and highly coupled nonlinear dynamics. There are several solutions in the literature, but most of them require fine-tuning of the parameters. In order to avoid the exhaustive tuning procedures, this work employs a Fuzzy Logic strategy for online tuning of the PID gains of the UAV motion controller. A Cascaded-PID scheme is proposed, in which velocity commands are calculated and sent to the flight control unit from a given target desired position (waypoint). Therefore, the flight control unit is responsible for the lower control loop. The main advantage of the proposed method is that it can be applied to any UAV without the need of its formal mathematical model. Robot Operating System (ROS) is used to integrate the proposed system and the flight control unit. The solution was evaluated through flight tests and simulations, which were conducted using Unreal Engine 4 with the Microsoft AirSim plugin. In the simulations, the proposed method is compared with the traditional Ziegler-Nichols tuning method, another Fuzzy Logic approach, and the ArduPilot built-in PID controller. The simulation results show that the proposed method, compared to the ArduPilot controller, drives the UAV to reach the desired setpoint faster. When compared to Ziegler-Nichols and another different Fuzzy Logic approach, the proposed method demonstrates to provide a faster accommodation and yield smaller errors amplitudes.

- **Published Article Paper:** Correia, C. A. M., Andrade, F. A., Sivertsen, A., Guedes, P. I., Pinto, M. F., Manhães, A. G., Haddad, D. B. (2022). Comprehensive Direct Georeferencing of Aerial Images for Unmanned Aerial Systems Applications. Sensors, 22(2), 604. Available at: [16]

  Optical image sensors are the most common remote sensing data acquisition devices present in Unmanned Aerial Systems (UAS). In this context, assigning a location in a geographic frame of reference to the acquired image is a necessary task in the majority of the applications. This process is denominated direct georeferencing when ground control points are not used. Despite it applies simple mathematical fundamentals, the complete direct georeferencing process involves much information, such as camera sensor characteristics, mounting measurements, attitude and position of the UAS, among others. In addition, there are many rotations and translations between the different reference frames, among many other details, which makes the whole process a considerable complex operation. Another problem is that manufacturers and software tools may use different reference frames posing additional difficulty when implementing the direct georeferencing. As this information is spread among many sources, researchers may face difficulties on having a complete vision of the method. In fact, there is absolutely no paper in the literature that explain this process in a comprehensive way. In order to supply this implicit demand, this paper presents a comprehensive method for direct georeferencing of aerial images acquired by cameras mounted on UAS, where all required information, mathematical operations and implementation steps are explained in detail. Finally, in order to show the practical use of the method and to prove its accuracy, both simulated and real flights were performed, where objects of the acquired images were georeferenced.

I.5 Work Organization

Chapter II will address the theoretical foundation for developing this research work. In this chapter, UAV system modeling and the algorithms for controlling and optical flow estimation will be found.

Chapter III presents the methodological foundation for UAV stabilization technique by using the hybrid Fuzzy-PID cascade, as well as other simulation models and controllers.

The methods' results are shown in Chapter IV. In addition, the necessary discussions will be held about the results in order to demonstrate the efficiency of the obtained methods.

The conclusions obtained throughout this dissertation, as well as the various possibilities for continuing the research, are presented in Chapter V.

Chapter II Background and Related Works

This chapter will present the background and all necessary mathematical foundations to support the understanding of the proposal of this master's qualification. Besides, at the end of this Chapter, it will be present the related works to support the contributions claimed in this work.

II.1 UAV inspection in general

The demand for services performed remotely has been increasing, especially when it involves human risk in carrying out the task/mission. This kind of demand for remote access to physical places is faced within the most varied scenarios: be it the search for survivors in extreme hypotheses of landslides or other weather events, or even in the analysis of solar panels and wind turbines. For this, the use of UAVs is increasing in different areas. Applications are extremely varied and the tools implemented in each case depends on the needs and costs of the project. For example, [36] and [69] performs a hazardous gas leak evaluation, thus resorting to specific sensors that are not needed to evaluate a solar panel.

Inspection using UAVs, in general, requires the UAV to be able to efficiently perform the inspected trajectory, perform an image capture, or detection of components and failures, well enough so that the analysis is actually possible and of a good quality. For this, some factors must be taken into account, such as the possibility of using the available tools, such as environments that make it impossible to use a GPS, cost of implementation and application of tools, computational cost of the chosen tools versus benefit tied to implementation.

For the inspection of industrial environments, for example, it is possible to mention [54] which implemented an inspection routine inside an industrial boiler plant of approximately 50 meters. As in the present work, the proposal of these authors was to implement a UAV that could cover areas of human risk to perform visual inspection tasks in environments where it would be difficult to obtain information through GPS. For this, both use data collected from IMUs and cameras coupled to the UAV. A common feature is the preset trajectory, however, the authors use the Prediction Error Method (PEM) trajectory prediction method to allow more aggressive evasive maneuvers while maintaining image stability, using

tools known as Features from Accelerated Segment Test (FAST) and Binary Robust Independent Elementary Features (BRIEF) which can be observed in detail in [70] and [10] respectively.

With the increase of renewable energy use, solar panels are being placed in several structures, such as houses' rooftop [90], buildings [17], power plants [83], among others. In the presence of anomalies and over time, the panel's efficiency can be deteriorated, bringing the need for regular inspections. For the specific scenario of inspections of solar panels, in [84] the main problem of understanding the edges of solar panels, due to the shadows and reflections of the environment, was addressed. To improve detection, the authors used Red-Green-Blue (RGB) and thermal images, in addition to two methods for comparative purposes: contrast correction and edge detection, and a convolutional neural network optimized for texture description vectors. For the first approach, compared to the present work, corners are identified through the optical flow without requiring processing costs to the project. In the case of the neural network, it is still an interesting approach, but it would also be necessary to evaluate the procedural cost-benefit for the implementation of the project. At first, it was not shown to be necessary and extremely beneficial to the project.

As can be seen, UAVs are applied to inspect a wide range of environments, such as slopes and dams [8], building structures [45], wind turbine [11], road surface [7], etc. Note that in most of these applications, the path is generated by a set of GPS way-points. However, in some places, the GPS signal can not be available or is not strong enough. Over the years, several different vision-based localization techniques were developed. According to [76], a great advantage of this approach is the possibility of using the UAV onboard camera for real-time localization.

A well-known method from the literature is Simultaneous Localization and Mapping (SLAM). This method is very commonly used for exploring unstructured environments, building a live map of the surroundings, and at the same time, sensing the robot's locations [50]. This method uses the salient key features in the image to estimate the camera location. However, in some inspection places, those salient features may not be available. Additional processing is still needed to locate the in-flight positions of UAV in the world coordinates.

In order to address this problem, [76] proposed a vision-model-based real-time self-positioning method for UAVs in GPS-denied places to support inspection applications. The authors claimed that compared to other localization methods needing additional onboard sensors, their proposed solution uses a single camera to continuously estimate the in-flight UAV's poses.

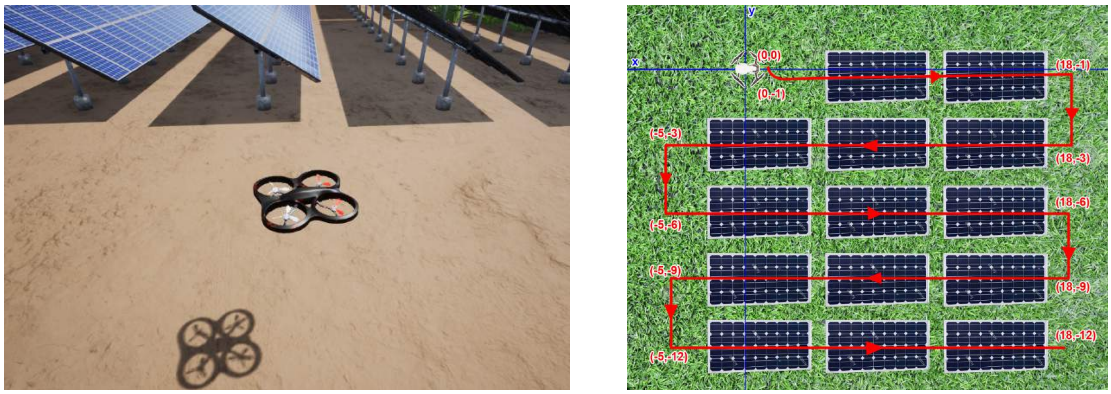
It is worth mentioning that model-based method approaches have been used for tracking and localization in different kinds of applications and robots, such as aerial robots [81], autonomous ground [48], manipulators [15], etc. However, there are still few types of research applying this kind of approach for aerial robots.

Being the route stabilization of great importance, it is essential that the UAV has a extremely satisfactory attitude control. In solar panels' inspection task specifically, but not restricted to, the image of the inspection must be stable so the inspector, either Neural Network, either human, can perform the image analysis. In real scenarios, many disturbances can prevent the inspection to be stable, such as wind gusts, loads added to the UAV, either intentional or small objects being blown by the wind and attaching to the UAV. The attitude controller must be capable of stabilizing the path as soon as the disturbance enhance the position error of the UAV. Not only the attitude, but also the altitude should be stable, to prevent incidents and damages to the UAV and environment, and also keeping the image in the desired distance from the observer. If the image is too close to the object, or too distant, the inspector won't be able to identify accordingly the maintenance points of interest, and take-off and wind gusts could cause this altitude to vary more than acceptable.

Another aspect worth mentioning is that there are different ways of performing such inspections with an UAV to fly above such installations: an autonomous, semi-autonomous, and manual process. In these all different flying situations, the UAV can fly very close to the object at a slow speed, producing valuable and reliable information about the solar module's condition.

The manual process requests attention for the image inspection and the expertise to operate the drone simultaneously. In that sense, using human resources to perform such repetitive tasks can cause incidents or decrease the inspector's quality of work due to the burthen this repetitive task causes.

The semi-autonomous method could use, for instance, a way-point to generate the route that the UAV would perform with defined high speed, visualization angle, points of interest, and where there might be a turn for it. This already gives a better perspective to the worker, that can pay full attention to the images being analyzed. From another perspective, this method also gives the employer a wider range of options for hiring workers that can perform the analyses but don't have the training to operate the UAV manually. The autonomous inspection would consist of an UAV capable of flying through the desired path with high stability and identifying any maintenance point or attention marks. The autonomous inspection is not considered in this work, as it relies on specific machine learning processes for the solar farm maintenance aspect. In II.1(a), an example of simulation environment contemplating



(a) Simulation Environment (From [5])

(b) Trajectory scheme. (From: Author)

Figure II.1: Example of solar farm environment and inspection scheme.

solar panels and in II.1(b) the proposed way-point scheme to optimize the UAV's trajectory during the inspection above the solar panels at a chosen height.

As can be seen, there is still space for improvements regarding both the stabilization process of the UAV and visual-based localization. Therefore, this work proposes some different approaches to support these two mentioned issues. First, we proposed a set of Fuzzy-PID controllers approach to deal with attitude and altitude stabilization control. Second, we show a solution to GPS-denied environments to support solar panel inspection by presenting a Optical-flow algorithm for aerial images collected by UAVs. All the necessary background to understand the proposed methods are given in the next sections.

II.2 ROS and Gazebo

ROS is the acronym for Robot Operating System, which is a flexible, open-source framework for robotics' software development. This system gathers a series of libraries, tools, and conventions to simplify tasks and yet create complex and robust robot behavior along with various platforms.

To work with such a system, it is required a Linux-based computer and additional knowledge in python is desirable. However, Open Robotics provided a great amount of learning articles from simple things, such as installation and firsts steps using it, available at `wiki.ros.org`. All information regarding ROS and Gazebo in this section is available at the library, with descriptions, images, examples and tutorials, as it can be seen in II.2

ROS communicates with the hardware through nodes and topics. Each node represents a process that performs computation [58], they can communicate with each other by streaming topics, Remote

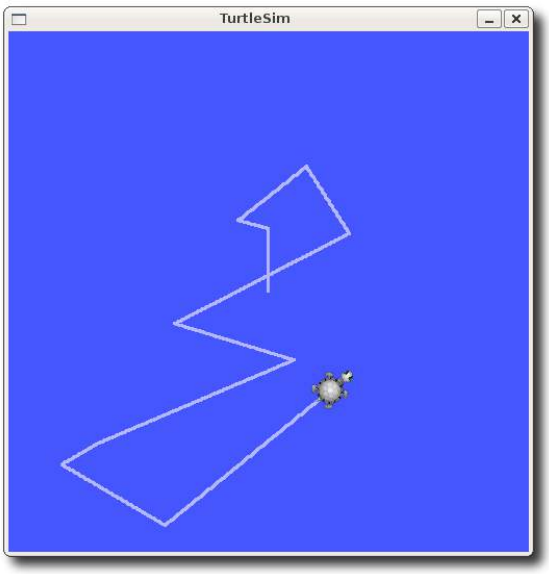


Figure II.2: TurtleSim GUI under the "Understanding ROS Topics" tutorial page. (From: [58])

Procedure Call (RPC) services, and parameter servers. According to the developer, using such nodes provide several benefits to the overall system, such as additional fault tolerance and decreased code complexity. Topics are named buses through which the nodes send and receive messages by publishing or subscribing to the desired topic. However, they are not aware of who they are communicating with and are intended to be unidirectional. ROS currently supports transport of two types of messages: Transmission Control Protocol/Internet Protocol (TCP/IP) and User Datagram Protocol (UDP) based.

Although the concept of communication for robotic systems may seem abstract at a first view, the tools provided to guide the user through which topics are available are very straight-forward. The most common commands used for topic reading are *rostopic list* and *rostopic echo*, which show in the command prompt, respectively, the list of all topics in use at the moment and the live value of the selected topic. This ensures that the list of topics available is easy to find and also to check if received data is according to the expected values.

Another way to use ROS for control is using services. Services are different from topics in that nodes can communicate with each other. Services allow the nodes to send a request and receive a response. Also *rosservice* has many commands that can be used on services, for example, list, call, type, and find. The *list* command prints information about active services, while *call* command calls the service with the provided arguments. *Type* prints the service type, and *find* shows the services by type [58].

ROS also provides a very interesting tool, namely *rqt_graph* that works as a GUI plugin for visualization of the graph of the computations that are running at the moment [58]. With this, it is much

easier to exemplify how the nodes and topics interact with themselves and to find communication errors or programming issues. An example is shown in Figure II.3 below.

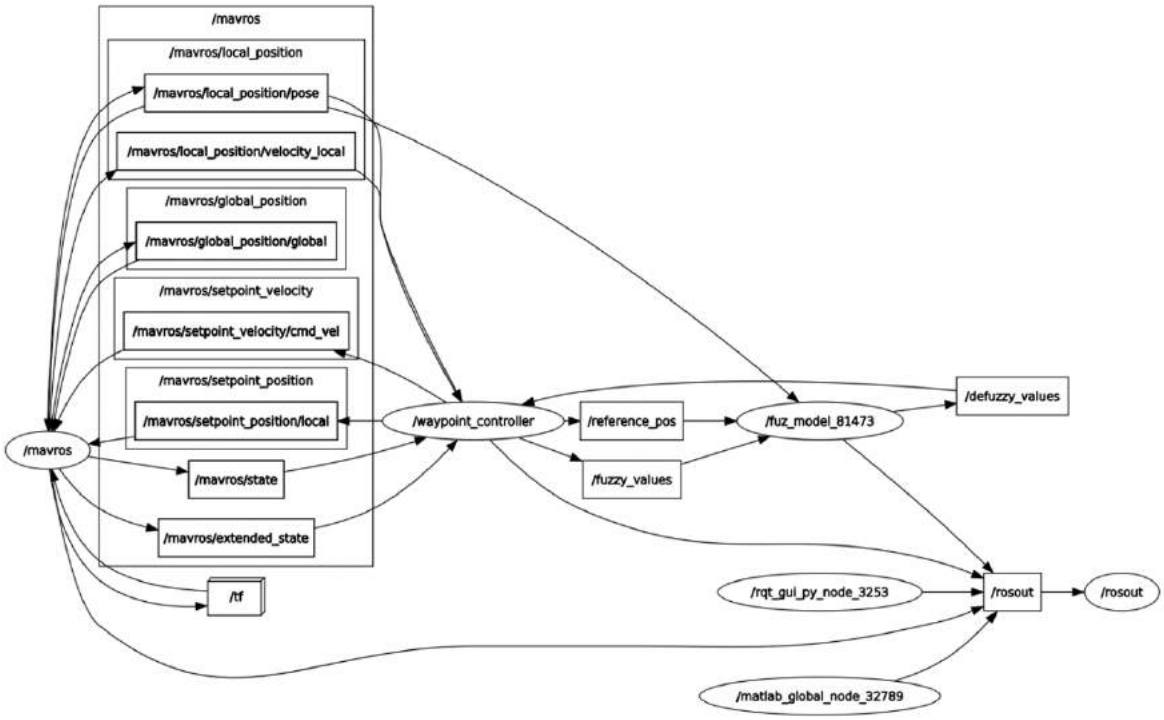


Figure II.3: Example of rqt_graph usage (From: Author)

Another great tool for friendly visualization and filtration of the ROS messages is *rqt_console*, which provides a GUI plugin that shows a viewer that displays messages being published to *rosout*. It collects messages over time and allows the user to view them with more detailed information, as well as allows to filter messages by many means. In addition to that, to keep track of the given information over time, it is also possible to use *rqt_plot*, which lets the user visualize scalar data published to ROS topics.

To save the collected information, ROS also provides *bags*, which are file format on ROS for storing message data. The bag files have a very important role as a variety of tools have been written to allow the user to store, analyze, visualize and process them. In order to store data into a bag file, the command *rosbag* is used. It stores into a bag file from the time that the command was activated until the user says to stop by pressing $Ctrl + C$.

By meshing different packages' usages, there is also the *rqt_bag*, that provides a GUI to display and replay the information within the bag files by, for example, showing message contents, displaying image messages, plotting configurable time-series of message values, and so on. The *rqt_bag* can also be extended via a plugin mechanism. There are core plugins contained in the *rqt_bag_plugins* package available in the *rqt_common_plugins* meta package to allow such implementations as shown in the

figure II.4 below.

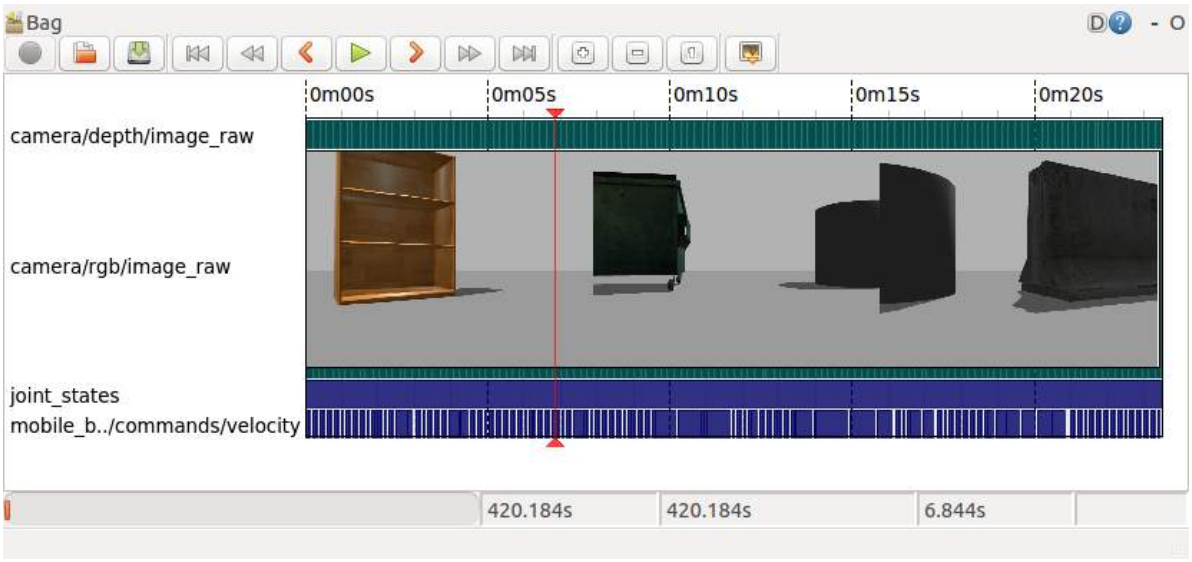


Figure II.4: Example of extended usages of bag file readings
Published in *wiki.ros.org/rqt_bag*

To visualize live information, it is also available the RViZ tool, which is a 3D visualization environment that allows the user to combine sensor data, robot model, and other 3D data into a combined view. It is also possible to send 3D markers into RViZ from the user's software.

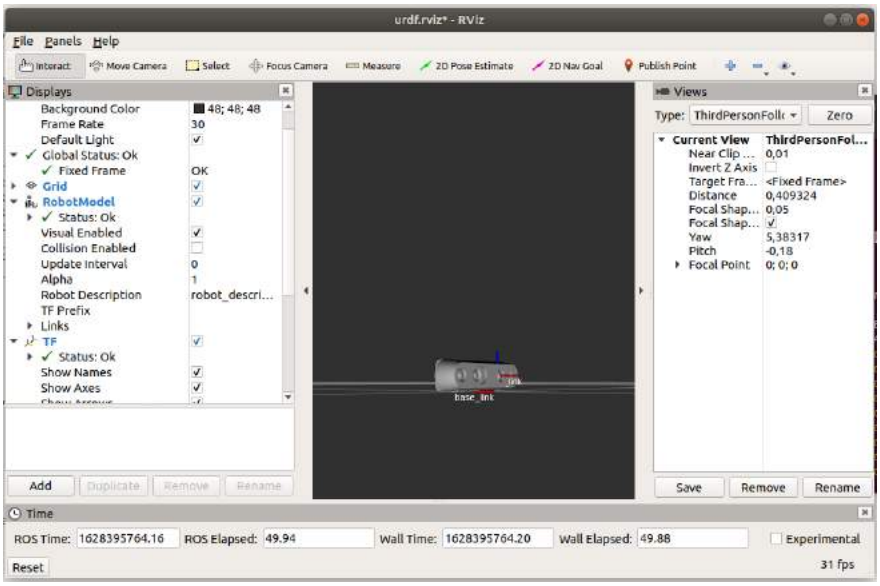


Figure II.5: Example of model and RVIZ utilization.
From: Author

ROS can be applied to many types of robots. In the documentation researched in July 2021, there are 166 available robots and components that support using the Operational System [57]. However, since the information on the website relies on the vendors or users to populate the list of available robots

and components, several are likely unlisted.

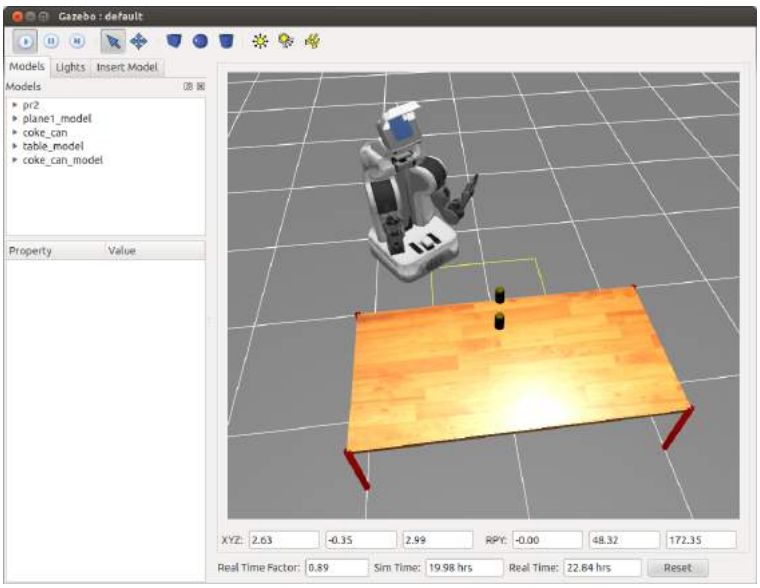
For instance, Intel RealSense is not listed on the above documentation, but the integration between ROS and several RealSense cameras is widely documented. It is possible to see in the wiki itself that there are Intel RealSense packages to enable the use of the R200, F200, SR300, and D400 cameras with ROS, but other derived cameras such as D435i can be used with the librealsense2 package. Note that the published topics differ according to the device and parameters.

The main simulation environment used in this work was Gazebo, which is also developed by Open Robotics and brings great compatibility with ROS, its open-source and has developed features for simulation, world creation, models imports, transport of asynchronous message passing, extensive set of sensors and noise models, physics and rendering engines [56]. Gazebo has developed so many features, that its difficult to summarize, however in II.6(a) its possible to see a preview of a simulation on the app, and in II.6(b) the web application, where the community populates the library with models, worlds and collections. In one of the researched works, which will be further presented, it was used Unreal Engine 4 (UE4), developed by Epic Games, along with the Aerial Informatics and Robotics Simulation (AirSim) plugin, developed by Microsoft. AirSim has several add-ins that are useful when simulating a flight operation, such as 'Gathering training data', 'Computer Vision mode', 'Weather Effects'.

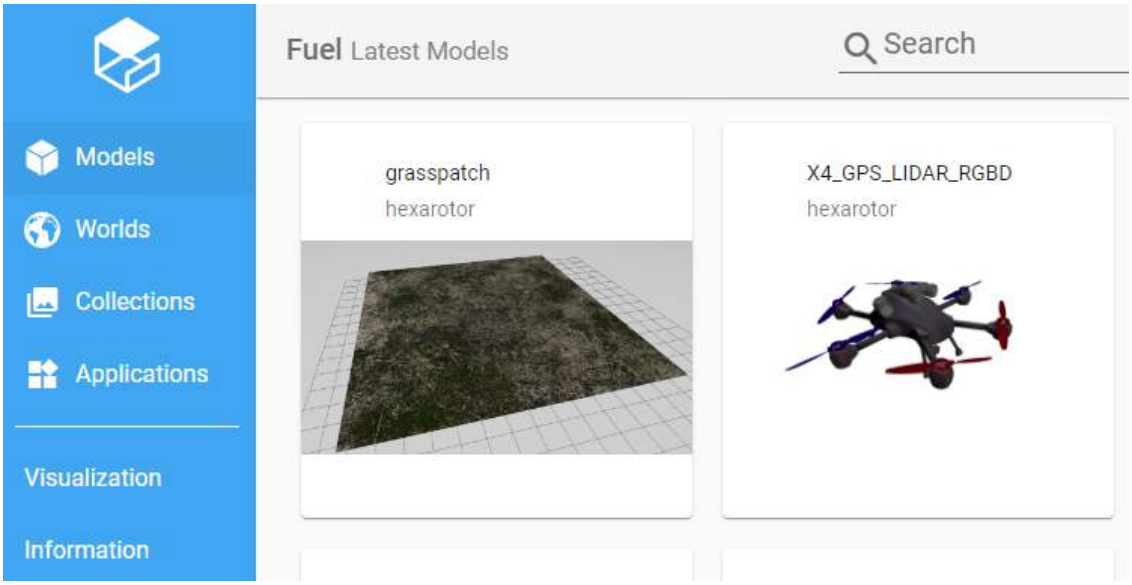
II.3 Optical flow

The optical flow method is an interesting solution to Robot navigation. A comparative study of Optical Flow and traditional sensors in UAV Navigation is provided by [14]. Over the years, different tasks have been accomplished with optical flow, which can be cited the distance estimation [26], altitude hold and obstacle avoidance [92], velocity and height estimation aided by inertial sensors [18], vertical landing [29], Stabilization of an Eight-Rotor UAV [67], among others. A great survey regarding the Optical Flow techniques for UAV navigation is proposed by Haiyang Chao [13].

Acquiring motion information from robots is a challenging task. One way to estimate motion from a sequence of image frames is by using optical flow algorithms. For instance, it is also a tool used for motion segmentation, detection, scene reconstruction, and robot navigation. Many methods for optical flow computation were proposed, such as gradient-based method (e.g., Lucas-Kanade [44] and Horn-Shunck methods [43], feature-based methods (e.g., Scale-Invariant Feature Transform (SIFT) [43]), and interpolation techniques [78].



(a) Gazebo simulation example (From: answers.ros.org)



(b) Gazebo Web Application (From: app.gazebosim.org)

Figure II.6: Gazebo software and web application.

A well-known optical flow algorithm was proposed in 1981 by B.D. Lucas and T. Kanade [44] became a widely used technique in computer vision since it is classified as a gradient method and has reasonable analytical speed keeping good results for several applications. This approach assumes that the flow is constant in a local pixel neighborhood. It solves the basic optical flow equations for all the pixels in that neighborhood through the least-squares criterion [77].

However, the pre-filtering step has a great role in gradient-based optical flow implementations [77]. Some parameters may be taken into account, for instance, the object brightness invariance, and spatial coherence, that is, if the image motion is uniform and if the movement of the surface changes over time.

The intensity of a pixel that has horizontal and vertical plane coordinates u, v is defined by $I(u, v, t)$ and it also expresses a feature that moves in $\delta u, \delta v$ and δt, as described in Equation II.1.

$$I(u, v, t) = I(u + \delta u, v + \delta v, t + \delta t) \tag{II.1}$$

A time derivation lead to Equation II.2, and I_u and I_v are the derivatives of the pixel located at point (u, v) at the instant t. The $\dot{u}$ and $\dot{v}$ are the variables of the optical flow *per se* also at time t, and points (u, v).

$$I_u\dot{u} + I_v\dot{u} + I_t = 0 \tag{II.2}$$

The aforementioned gradient method aims to solve Equation II.2 by calculating (u, v). Nevertheless, for each pixel, there is a scalar equation with tree values (I_u, I_v, and I_t), but only two unknown scalar $\dot{u}$ and $\dot{v}$. This clarifies that there is an undetermined system, this problem is known by *aperture problem* [47].

The Lucas-Kanade algorithm is a Gauss-Newton gradient descent non-linear optimization algorithm, and it aims to align a template image T(x) to an input image I(x), where $x = (x; y)^T$ is a column vector that contains the pixel coordinates. If the Lucas-Kanade algorithm is being used to compute optical flow or to track an image patch from time t $= 1$ to time t $= 2$, as it is in this project, the template T(x) is an extracted sub-region, a window of defined dimensions, of the image at t $= 1$ and I(x) is the image at t $= 2$ [6]. It aims to minimize the sum of the squared error between two images.

In other words, the Lucas-Kanade method "guesses" the displacement of a neighborhood by looking

at changes in pixel intensity by analyzing the gradients of the image and the neighborhood. The result of this analysis and the post-processing is a set of optical flow vectors over the image, giving the trajectory that the chosen pixel did [66].



Figure II.7: Example of optical flow.
Published in *nanonets.com/blog/optical-flow*

Parallel to the tests using the D435i, which will be further described, and preliminary studies were carried out on the functioning of the optical flow. For these tests, it was necessary to perform the previous recording of three videos of sensor movement and add them to the optical flow code, together with the implementation of the Lucas-Kanade method, to determine the velocity vector of the points of interest located in the videos. The code in Python was tested in JupyterLab, inside the Windows 10 operating system, and successfully printed the delta x and delta y matrices of the analyzed points.

As previously mentioned, optical flow identifies and tracks points of interest, which are generally adopted as corners. After identifying the points in the first image of the video, the script considers the next frame and records the position within the video where that point is located at the moment [53]. This analysis is carried out frame by frame and the outputs of the script symbolize the positions of the points, which is the variation in the position in which they are found frame by frame. From this, it is possible to create a vector of the traversed motion, which is known as an optical flow [30].

The Lucas-Kanade method is widely used to optimize the adequacy of the optical flow in certain scenarios [55]. In many studies, it shows itself superior in its tracking ability by analyzing not only the point in question but also a 3x3 dot matrix around the point of interest. This ensures that interference from light variation, lack of focus, or blurring in the image interferes less in the analysis than in other

models.

In addition, it is also interesting to use image optimization tools, such as Homography, for example, which can perform different types of optimization, whether it is lighting improvement, creating "three-dimensional" environments, panoramic images, or reducing image distortion. This would facilitate, for example, the identification of points of interest in the optical flow [4].

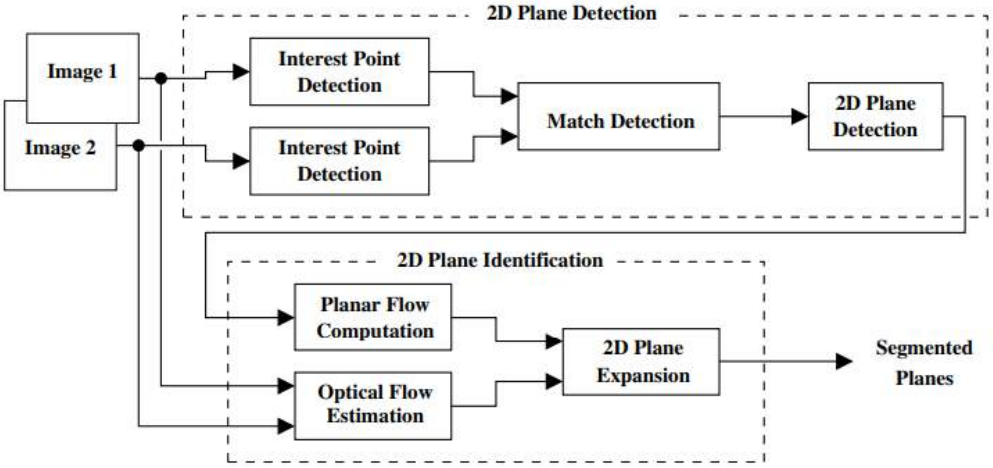


Figure II.8: Example of using homography in conjunction with optical flow.
Published in [4]

Other studies also suggest that the graphical analysis be completely performed through neural networks [32], both for image processing as well as for the detection of points of interest and formation of optical flows.

Still dealing with graphical analysis, another possible alternative for implementation in the research project would be Visual Simultaneous Localization and Mapping (vSLAM), which satisfactorily accommodates environmental changes such as movement of objects or people and also changes in lighting for example [35]. This method combines odometry and mapping analysis performed by standard SLAM, with video-oriented object detection and trajectory prediction.

For the proposed project, as well as in others previously mentioned, the intention in using optical flow is to collect UAV speed and location information to correct its position and implement the corrected route based on the previous state.

II.4 Stereo Camera

As mentioned briefly, this sensor is a powerful tool capable of capturing not only 2D images but also uses an infrared depth sensor and an inertial measurement unit (IMU). It has a reduced size and weight of 90 x 25 x 25mm and 72g, making it a great candidate for use in UAVs.

This device does not require initial calibration, but this can be done at the user's discretion. The range indicated by the manufacturer varies between 30 centimeters and 3 meters of distance, possibly due to the efficiency of the infrared sensor, which has a minimum distance of 28 cm and reduces the accuracy to less than 2% when the distance is equal to or greater than 2 meters. Despite this, the tests carried out indicate that the sensor accurately identified the presence of objects beyond that distance, in line with the manufacturer's perspective that the sensor would have a range greater than 10 meters. In addition, the depth sensor has an 87° x 58° field of view, a frame rate of 90 fps, and an image resolution of up to 1280 x 720p.

The RGB camera performs 2D image capture and has 2 MP sensor and 1920 x 1080p frame resolution. The camera's field of view is 69° x 42° and the frame rate is 30 fps. The D435i also features an IMU with 6 degrees of freedom.



Figure II.9: Intel RealSense D435i.
Published in unmannedtechshop.co.uk

Further, the configuration aspects regarding the RealSense camera will be presented. Initially, the reading software indicated by the manufacturer was used for learning about the sensor, the Intel RealSense SDK 2.0, also known as Viewer. As a versatile tool with support for several programming languages, frameworks, and operating systems and a pleasant user interface, this was the first step in discovering the sensor's capabilities under ideal conditions.

The software was used on the Windows 10 operating system in order to ensure that any necessary drivers were easily found and installed. First, when testing the sensor, all functions were working perfectly, yet the software suggested a calibration. Calibration was done intuitively, and the measurement

test through the software had very high accuracy. After calibration, two points whose median was 1.5 m distant from the sensor were selected, and, by using the SDK, the distance between the points was measured. The difference between the measurement with the sensor and the measuring tape was 1 centimeter. In addition, the tool has two forms of data acquisition: constant recording in a bag format and the other punctual, such as a snapshot.

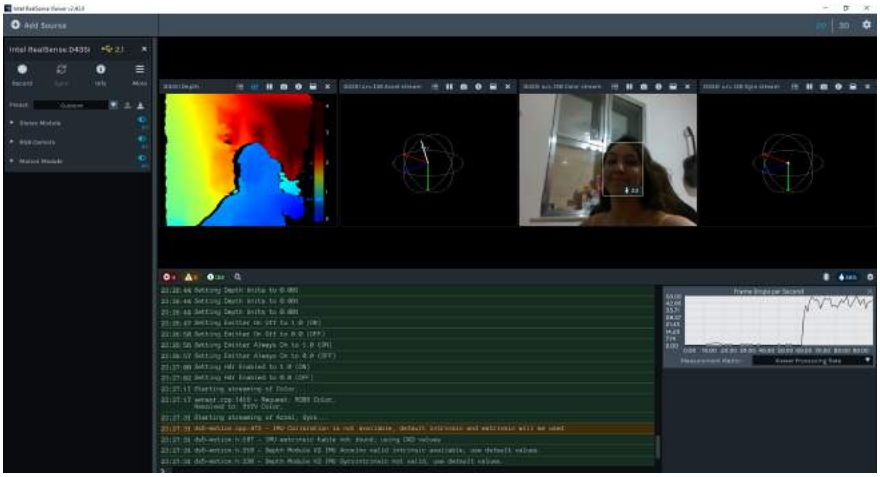


Figure II.10: Intel RealSense SDK loading all sensors from D435i.
From: Author

Later, in order to couple as many tools as possible, the operating system was migrated so that the sensor was used in Ubuntu 16, whose use is indicated for applications using ROS Melodic with greater compatibility. In this scenario, there were minor setbacks due to the sensor configuration related to ROS. The SDK installation, however, took place practically and efficiently. There was no difficulty in using the two tools, regardless of the operating system.

On the other hand, it required considerable research so that the sensor topics were read by the ROS, using the command "*rostopic list*" and also "*rosbag*". At first, topics were not started by the *rs_camera.launch* application, requiring the user to go to the file and manually change some initial values, such as sensor data and triggering of each function. An alternative option was to insert the desired settings manually on the command prompt along with the *rs_camera.launch* application. It was also not possible to analyze the images coming from the sensor using the RViZ software until the completion date of this report.

Furthermore, it should be taken into account that, even using the manufacturer's software recommendations, a USB-C 3.0 cable should be used for data transfer. Tests carried out during this development show that even using the manufacturer's software, there was a constant data overflow problem, even when only the RGB camera and the IMU were active. This is due to the huge amount of

data transmitted, and users report that this problem can be solved only by replacing the cable. However, it is noteworthy that this problem was not what caused the difficulty of data acquisition through the *rosbag* mentioned above.

II.5 Control

In this section will be presented the Quadrotor UAV's system modeling, along with the cascaded-PID and fuzzy system.

The sections of System Modeling and PID Cascade derives from our paper entitled "Hybrid PID-Fuzzy controller for autonomous UAV stabilization", published in INDUSCON 2021 [27].

System Modeling

In the last years, many tasks have been automatically solved by UAVs, which is explained by their maneuver flexibility and rapid growth in usage. These tasks can be for military missions, agriculture, surveillance, mapping, aerial photography, resource exploration, traffic monitoring, and anti-terrorism investigation, among others [3, 62, 61].

Perhaps the most commonly seen among the UAVs are of the quadrotor type configuration, which has four rotors arranged in a squared frame. In this model, two rotors positioned diagonally rotate in the same direction, while the other two have the opposite movement, as can be exemplified in Figure II.11. This gives the possibility to stay in place in mid-air because the motor torques cancel each other when they keep the same angular speed.

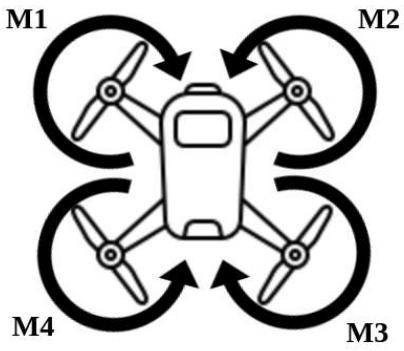


Figure II.11: Quadrotor structure.
Adapted from: [27]

The expression in Equation II.3 represents the mathematical model of the quadrotor with respect to the body frame's axes. The table II.1 show and describes the system parameters. The U_i $(i = 1, \cdots, 4)$ variables are the generalized actuation forces responsible for the nonlinear model's thrust, roll, pitch, and yaw movements. The Ω_r is the engines' residual angular speed, and $\xi_{(\phi,\theta,\psi,h)}$ represents the disturbances that affect the dynamics of the roll, pitch, and yaw angles, respectively.

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \ddot{z} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{zz}}\dot{\theta}\dot{\psi} + \theta\frac{J_r}{I_{xx}}\Omega_r + \frac{l}{I_{xx}}U_2 \\ \frac{I_{zz}-I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} + \phi\frac{J_r}{I_{yy}}\Omega_r + \frac{l}{I_{yy}}U_3 \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{l}{I_{zz}}U_4 \\ g - \frac{\cos(\phi)\cos(\theta)U_1}{m} \\ \frac{U_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \frac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) + \sin(\phi)\cos(\psi)) \end{bmatrix} = \begin{bmatrix} \xi_\phi(t) \\ \xi_\theta(t) \\ \xi_\psi(t) \\ \xi_h(t) \\ 0 \\ 0 \end{bmatrix} \tag{II.3}$$

Table II.1: Parameters of the Quadrotor Mathematical Model

Parameters	Description
Ixx, Iyy, $Izz(kgm^2)$	Moment of inertia about Earth frame
ϕ, θ e ψ	Euler angles *roll*, *pitch* e *yaw*
$m(kg)$	Total vehicle mass
$l(m)$	Quadcopter arm length
$b(Ns^2)$	Impulse coefficient
$d(Nms^2)$	Drag coefficient
$Jr(kgm^2)$	Rotor moment of inertia

Using the Motor Mixing Algorithm, it is possible to analyze how the U_i inputs are related to rotors' angular speeds $\Omega_{(1,2,3,4)}$ II.4, in other words,

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ \Omega_r \end{bmatrix} = \begin{bmatrix} b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ b(-\Omega_2^2 + \Omega_4^2) \\ b(\Omega_1^2 - \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{bmatrix} \tag{II.4}$$

Several control platforms were developed to increase the UAVs' degree of autonomy to allow them to perform tasks simultaneously in the semi-autonomous or autonomous way [73, 23]. Regarding flight stabilization, it is widespread to use a navigation system to feed a classic PID controller, which has a simple structure, good stability, and less dependence on the exact system model. In this work, it was proposed the usage of Fuzzy Logic theory to deal with systems that are not easy to be modeled because of their nonlinearities, such as in [19, 21, 86, 52].

II.5.1 PID Cascade

The cascaded strategy consists of the interaction between two control loops. The external one is responsible for generating a reference signal. In contrast, the internal one, fed by the latter, is responsible for computing signals sent to the actuators.

The diagram of Figure II.12 depicts the simplified cascaded PID controller scheme. The external loop receives as input the position error to generate a setpoint reference value as output that will be used in the internal loop. The internal position control loop is responsible for controlling the UAV motors. Provided that the roll and pitch angles converge to their (generated) reference values, then Equation II.4 is used to compute the rotors' velocity.

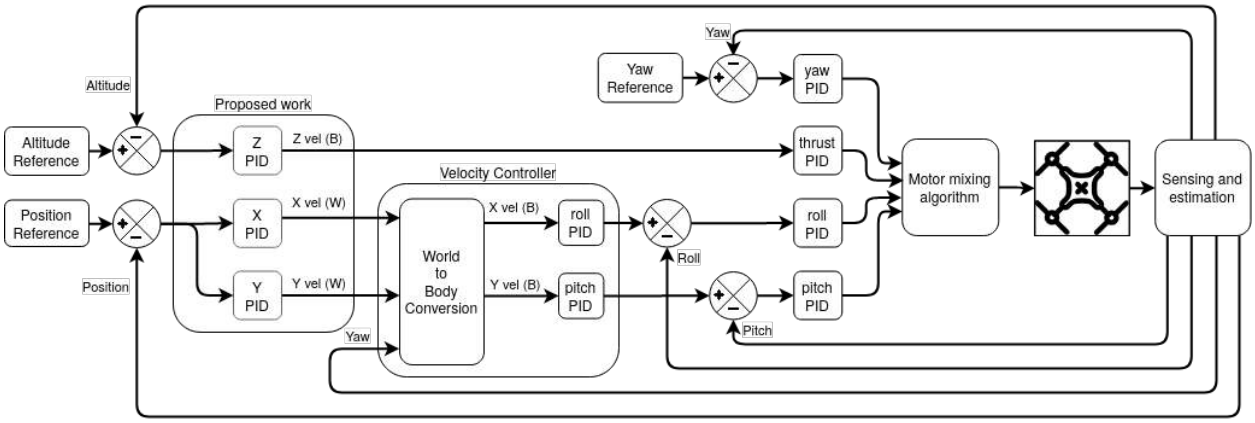


Figure II.12: Simplified cascade PID controller.
Published in [27]

The block named *world to body conversion* converts the external position reference to the UAV frame reference. This block represents the product of a normalized vector of two elements (x and y errors) by a 2×2 homogeneous transformation matrix $R_{Z,\psi}$ in which the parameters are trigonometric functions of the yaw angle ψ, as shown in Equation II.5.

$$\begin{bmatrix} X_{world} \\ Y_{world} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} X_{body} \\ Y_{body} \end{bmatrix} \tag{II.5}$$

This matrix can also be represented as an expanded version to a 3-coordinate system, as follows:

$$\begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{body} \\ Y_{body} \\ Z_{body} \end{bmatrix} \tag{II.6}$$

The matrix II.6 represents the rotation of an angle ψ around the z-axis, which is denoted by $R_{Z,\psi}$,

that is, the yaw rotation in Euclidean space. For the representation of the roll and pitch rotations, there are the matrices $R_{X,\phi}$ and $R_{Y,\theta}$, respectively. In Equation II.6 is shown a complete transformation of the three roll, pitch, and yaw angles. Such matrix product is not commutative, that means that, the product order matters and this transformation is done by parts, from right to left, and it is denoted by $R_{\psi,\theta,\phi} = R_{Z,\psi} \cdot R_{Y,\theta} \cdot R_{X,\phi}$.

$$R_{\psi,\theta,\phi} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{II.6}$$

A way of finding the direction in which the UAV is positioned about the world frame is through the Equation II.7, whose parameters the angles of ϕ, θ, and ψ, which represent the first two lines of the third column of the array $R_{\psi,\theta,\phi}$:

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \end{bmatrix} \tag{II.7}$$

The direction of the thrust in which the UAV is oriented is indicated by this spatial representation. Considering that the UAV z-axis (yaw) is pointing to the top of it from the center and being transposed to the plane (x, y) and then normalized about the axis's total size, there is the vector $[u_x, u_y]$ with the world frame.

II.5.2 Fuzzy system

Intelligent control systems must be able to perceive the environment and its changes over time and process the information to obtain the desired result, thus making real-time decisions based on the information captured at every moment [62]. With the development of autonomous technologies, it is natural that control systems are increasingly present in order to adapt and solve problems related to complex tasks.

A major challenge within the control of UAVs is the management of stability and variations according to environmental factors. Classic controllers do very well for certain uses, but it might be necessary to use two or more controllers occasionally. With this in mind, a fuzzy controller was developed together with cascaded-PID, which re-balances the PID gain values to simulate the implementation of different

controllers according to the system's needs.

As suggested by this work, other authors considered the possibility of performing a mixed controller. In [72] a conventional PID controller was implemented for comparison purposes with a self-tuning PID-based fuzzy controller on adapting to situations with uncertain parameters and maintaining robustness and performance. The premise of these authors is similar to the present project when it tries to increase the usefulness of a PID, which has a good performance, but for specific systems. In this project, however, the fuzzy controller linked to the cascaded-PID was implemented, which already has performance superior to the conventional PID. Both works use the errors as feed values for the controller, but the results obtained by the author differ from those present in this project. The authors did not obtain better performance with the self-adjusting PID, while the present project showed improvements in the fine-tuning of the controller and a decrease in situations where there could be an overshoot.

In [19], on the other hand, the authors implemented a Fuzzy controller that was able to recreate routes to follow a certain element, scan environments using way-points, and trace the route taken. For this, a fuzzy controller with two modules was implemented, an altitude module and a latitude-longitude module, with an additional element that calculates the error, to improve the controller input data. In the aforementioned project, the authors implemented a Mamdani-type controller, as in the present project. The Mamdani type controller shows a statistically similar result compared to the Takagi-Sugeno controller [87]. However, the latter demands that the chosen value be precise, while the Mamdani controller allows for greater malleability of the data because it is a more flexible model. The project authors developed the fuzzy altitude controller to collect the altitude error, change in altitude error, and airspeed inputs, i.e., vehicle speed. This module has elevator and throttle outputs, which perform the UAV's up and down function. The fuzzy latitude-longitude control module takes the heading error and heading change and displays the roll angle output. The modules can be seen in detail in Figure II.13. While the proposed fuzzy controller has the x-axis position error and its derivative as input, it outputs the cascaded-PID gains for re-balancing the controller according to the system's needs.

Another example of fuzzy controller usage for UAV flights is seen in [33]. The authors have developed a fuzzy controller that receives the position error, calculated by the micro-controller that compares the position and the desired values stored within, along with the roll, pitch, and yaw angles to determine the voltage output to be supplied to the tricopter's motor. The results of this experiment compared to a PID controller showed that the fuzzy controller by itself ensures a sufficient degree of adaptability, autonomy, and reliability of the UAV, to improve the movement's speed and accuracy along the trajectory, although it has shown a lack of control when compared to the PID in overshoot situations. This

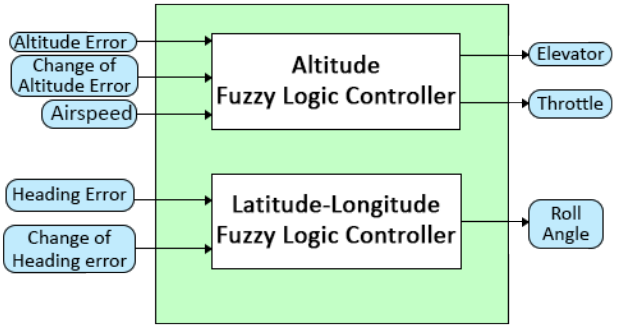


Figure II.13: Fuzzy controller proposed by Doitsidis *et al.*
Adapted From: [19]

study proves the value of the fuzzy controller by itself but also highlights the issues that it alone cannot handle. For this reason, the mixed controller was chosen for the present project, combining two widely used controllers and enhancing the PID results with a cascaded method.

Although the fuzzy set can be used as described above, there are many other ways of developing controllers with the aid of fuzzy systems. In this work, the research of three fuzzy sets was developed. In two of them, the fuzzy set is tested with different strategies, and in the third research, the fuzzy set is also used to apply the correct weight in the decision-making of the UAV.

For the first research work, the fuzzy set is used as a complement of the cascaded-PID, which defines the PID gains for position control [27]. In the second one [5], the fuzzy controller also defines the PID gains, but it is responsible for tuning the horizontal position and altitude control. In the last research [49], it was developed a fuzzy-scheduler, which is responsible for the arrangement of the fuzzy gains of the position and altitude controllers, but it also prioritizes which controller should receive more priority based on the errors of the system for that moment.

Chapter III Proposed Methodology

This chapter will detail the proposed methods used for the UAV attitude stabilized controlling based on the Fuzzy technique. Note that the controlling methods were published in [27], [5], and [49]. Therefore, the sections that refer to this method were derived from published papers.

In [27] and [5], were developed two similar methodologies with some differences. The first will be described in a general aspect, then the interface between the UAV and ROS system, and the second will be described further, following the same description line.

III.1 Hybrid PID-Fuzzy Controller

For the development of the first work [27], as a preview of the further explanation, a route is defined on which the UAV bases itself to carry out the inspection of the solar panels. This route is defined using way-points. Once the UAV takes off, the camera images also start to be acquired, as well as the IMU data. The camera data is sent to the optical flow algorithm using the Lucas-Kanade method, which reads it frame by frame and establishes the speed and direction of the UAV. Optical stream output data and IMU data are then sent to the Extended Kalman Filter (EKF), which performs future position prediction based on current state values. From that, the system receives the new heading value that must be defined to recalculate the route so that it is as faithful as possible to the way-point. In [27], although this studies took place, only the fuzzy-cascaded-PID was published, leaving the rest of the research to be appreciated by the present work.

In this work, the main contribution from this author is the development of the fuzzy controller as a whole, the fuzzy sets, rules, choice of inference method and performance test. The development of the way-point and gathering of IMU data was also a contribution, but not published in the cited article.

In parallel with this approach, the developed controller performs system stabilization, regardless of the sudden trajectory change. After carrying out the mission, the UAV must return to the starting point and land.

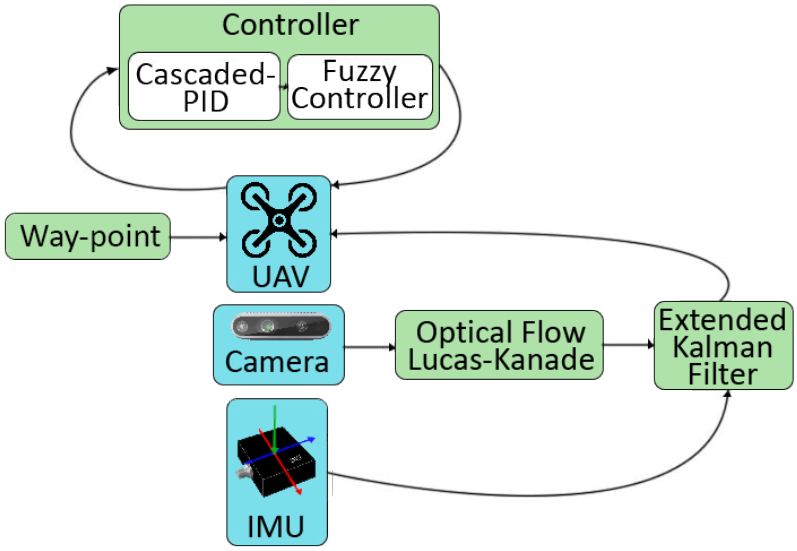


Figure III.1: Flowchart of data for this project.
From: [27]

Initially, a virtual world was created using the software GAZEBO/ROS. By using the MAVROS package, which provides the MAVLink protocol, and a telemetry device, it is possible to access the UAV's current position in the space through its sensors. For this setup, the desired position or speed setpoint was applied to the Px4 flight controller, creating a simple and efficient external control loop.



Figure III.2: The UAV 3DR IRIS used in the tests.
From: [27]

The ROS interface with UAV works as follows. Initially, the Ground Control Station (GCS) has the ROS core processing data in each of the code's pre-configured nodes. The computer receives the information from a telemetry module available on both the computer and the UAV, using the MAVLink communication protocol. This protocol is considered a lightweight messaging protocol for communicating with drones and among onboard drone components.

The PX4 software, which runs on the PixHawk Flight Controller Unit (FCU) hardware, is responsible

for acquiring information from the peripheral sensors and modifying these values in the actuators to which it has access. Figure III.3 illustrates the described interface. In the left part of this figure, the GCS is a computer that runs all the necessary algorithms. Both computer and UAV have a telemetry module that enables the exchanging of data. In this scheme, the UAV performs the necessary movement sent by the GCS. The GCS runs the operating system Ubuntu 18.04.4 LTS 64 bits, and it has an Intel Core i7-5500U CPU @ 2.40GHz x 4, Intel HD Graphics 5500 (Broadwell GT2), and a 15.6 Gb of memory.

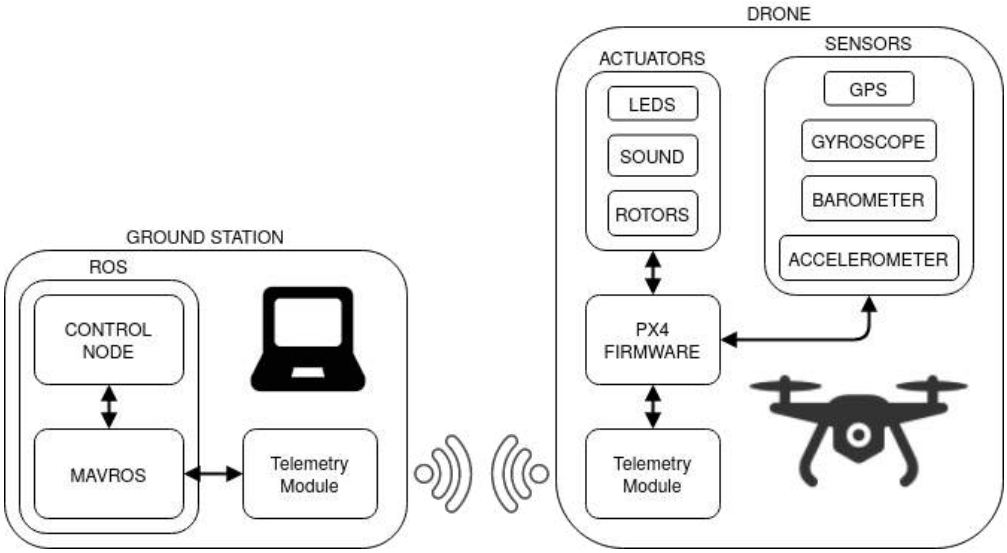


Figure III.3: Interface UAV-ROS.
Adapted from: [27]

The choices of the *Membership Functions* were based on the PID gain values that have resulted in a desired closed-loop response. For instance, if the position error and its derivative are sufficiently small in the present approach, there is no need for large PID gain values. Since no *crisp functions* were used, final gains regularly tend to be slightly different from the ones in the center of the membership function. The fuzzy inference technique adopted in this work is based on the classical Mamdani [74] inference method and the centroid defuzzification method.

The fuzzy rule Table III.1 can be seen in the following. The symbol 'B' represents 'Big', 'M' for 'Medium,' and 'L' for 'Low.' In X_{error}, the indexes 'n' and 'p' (in 'B_n' or 'L_p' of X_{error}) denote 'negative' and 'positive', respectively. Also, the outputs should be interpreted as P, I, and D gains. Hence, the output notation 'LMB' means that the proportional (P) gain is Low, the integrative (I) gain is Medium, and the derivative (D) gain is Big. Additionally, a module component for adjusting X_{error} is recommended for reducing the rule table since multiple outputs repeat themselves based on a positive or negative value of the error input.

Table III.1: Parameters of the Quadrotor Mathematical Model

	X_{error}				
$X_{error}D$	**Bn**	**Ln**	**Z**	**Lp**	**Bp**
Low	BBL	MMM	LLB	MMM	BBL
Med	MML	MLB	LMM	MLB	MML
Big	BBL	MLB	BBL	MLB	BBL
Zero	-	-	MLB	-	-

The rule table was developed based on an individual analysis of each PID gains. Therefore, if the X_{error} and its derivative are sufficiently high, then the controller demands a higher proportional gain. In the same way, if there is an urge to receive a faster response, then integrative gain should be enhanced. The derivative gain is expected to be requested to avoid minimal errors.

For a better understanding, one can also view the rules below.

nº	if X-error =	val	and X-error-D =	val	, then	P	I	D
1	if X-error =	Z	and X-error-D =	Zero	, then	M	L	B
2	if X-error =	Bn	and X-error-D =	Low	, then	B	B	L
3	if X-error =	Ln	and X-error-D =	Low	, then	M	M	M
4	if X-error =	Z	and X-error-D =	Low	, then	L	L	B
5	if X-error =	Lp	and X-error-D =	Low	, then	M	M	M
6	if X-error =	Bp	and X-error-D =	Low	, then	B	B	L
7	if X-error =	Bn	and X-error-D =	Med	, then	M	M	L
8	if X-error =	Ln	and X-error-D =	Med	, then	M	L	B
9	if X-error =	Z	and X-error-D =	Med	, then	L	M	M
10	if X-error =	Lp	and X-error-D =	Med	, then	M	L	B
11	if X-error =	Bp	and X-error-D =	Med	, then	M	M	L
12	if X-error =	Bn	and X-error-D =	Big	, then	B	B	L
13	if X-error =	Ln	and X-error-D =	Big	, then	M	L	B
14	if X-error =	Z	and X-error-D =	Big	, then	B	B	L
15	if X-error =	Lp	and X-error-D =	Big	, then	M	L	B
16	if X-error =	Bp	and X-error-D =	Big	, then	B	B	L

Figure III.4: Fuzzy Rules in details.
From: Author.

The membership functions' output range values were defined based on matching PID values representing neither an inefficient nor unstable controller. The recommended values can be seen as follow:

$$Pmin = 2.600 \quad Pmax = 3.30$$

$$Imin = 0.250 \quad Imax = 5.00$$

$$Dmin = 0.565 \quad Dmax = 0.85$$

$$Pavg = 2.9500 \quad \Delta P = 0.700$$

$$Iavg = 2.6250 \quad \Delta I = 4.750$$

$$Davg = 0.7075 \quad \Delta D = 0.285$$

Also, there is no need to apply any rule for X_{error} different than zero and $X_{error}D$ equals zero since the other rules can outcome further problems in this situation. Nevertheless, adding the rule for both null errors is essential since UAVs should stay in a constant position. The fuzzy membership functions used here are depicted in Figure III.8.

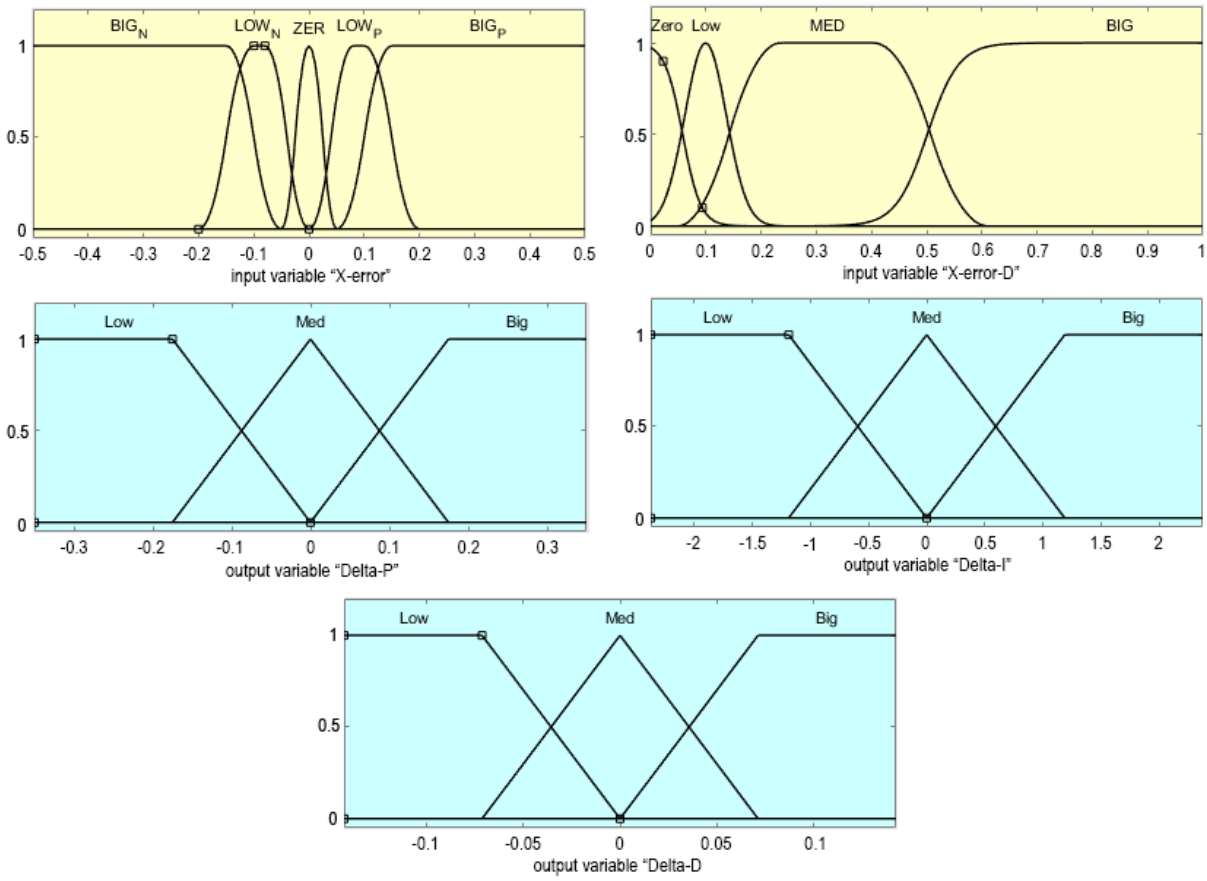


Figure III.5: Fuzzy membership functions.
From: [27]

The testing stage of the controller comprehends the evaluation of stabilization and the time for it

to occur. Four routes were developed for testing the performance and intensity of movements. Each test was performed three times for the classical PID controller and the PID-Fuzzy controller. FCU data were used for state estimation. The PID values used in the testes are shown in Table III.2.

Table III.2: PID configuration used.

	P	I	D
PI	3.2	0.95	-
PID 1	2.6	0.25	0.4
PID 2	3.255	4.912	0.565

Three fuzzy models were evaluated in the test stage, but the proposed one outcomes with the best general results. Figure III.6 shows the performed tests. The white circle represents the desired position, and the black circle underneath the white one is the UAV's position. The desired trajectory can be summarized as follows:

- **T0:** UAV stays in its initial position;
- **T1:** moves from initial position to point 1;
- **T2:** moves from point 1 to point 2;
- **T3:** moves from point 3 to point 4;
- **T4:** moves from point 4 to point 5;

Note that all trajectories defined for the tasks are straight lines between two reference points. The only difference among them is the displacement distance.

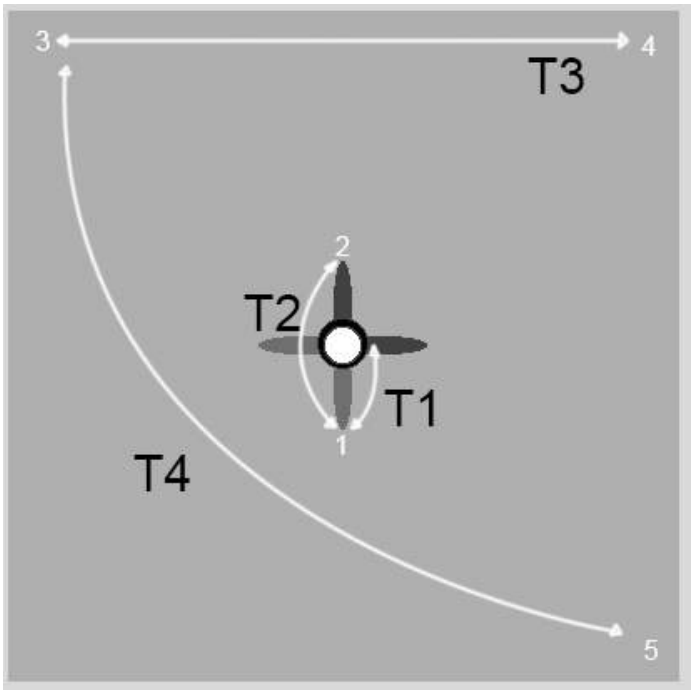


Figure III.6: Illustrative diagram of the predefined trajectories for the tests.
From: [27]

III.2 Motion Control with Fuzzy Tuning of Cascaded-PID Gains

In this work [5], it was also developed a controller that combines a cascaded-PID with a fuzzy algorithm that is responsible for calculating the PID gains, which is the main contribution from this author of this work. This unity between the controllers can provide an effective solution to the system's non-linearity, and, as a result, the system can accurately converge to the desired position in fewer iterations.

While the external loop uses the position error to generate a setpoint reference value for the internal loop, the internal control loop is responsible for controlling the UAV motors. The velocities from the rotors are calculated after demonstrating that both pitch and roll angles converge to their generated reference values. This strategy allows for a more straightforward implementation, and testing tasks that are benefited as more advanced sensors are used.

The tests also considered a simulation environment as depicted in Figure III.7, and compared its results with other solution as ArduPilot built-in guided mode controller, a fuzzy solution tuned by the methods in [89], detailed in [39], and a Cascaded-PID solution tuned by the Ziegler-Nichols method. The simulation environment runs in a Windows operating system with Windows Subsystem for Linux (WSL) Ubuntu 18.04.5 LTS 64 bits. The computer has an Intel i9-9900K 3600 MHz processor with an NVIDIA GeForce RTX 2080Ti graphics card and 48GB RAM.



Figure III.7: Simulation Environment provided by the software Unreal Engine 4 with AirSim. From [5]

The simulation trials were developed using the software Unreal Engine 4 (UE4) along with the AirSim plugin [75], which was developed by Microsoft. This plugin provides important resources for this

work, as the input/output signals, corresponding to the behavior of sensors and actuators, are sent to and received from the UAV flight controllers with a better response and smoothness.

Because the simulation environment was changed, some other minor changes had to be made to solve compatibility issues, as the sensibility and physics applied to the world were slightly different. Many changes were done also to improve the controller, and the methodology for the proposed controller will be described below.

At the first moment, the ultimate gain (K_u) and the oscillation period (T_u) [91] are obtained by removing the integrative and derivative parts of the PID and increasing the proportional gain. This last gain is then increased until the output of the control loop has stable and consistent oscillations. In this work, the measured ultimate gain was 2.2 and the oscillation period of 4 seconds.

With the measured values of K_u and T_u, the Zhao/Larson gains $K_{PI,min}$, $K_{PI,max}$, $K_{DI,min}$ and $K_{DI,max}$ were calculated:

$$K_{PI,min} = 0.32K_u\,, \quad K_{PI,max} = 0.6K_u\,, \tag{III.1}$$

$$K_{DI,min} = 0.08K_uP_u\,, \quad K_{DI,max} = 0.15K_uP_u. \tag{III.2}$$

As the traditional Ziegler-Nichols presented a high overshoot and, noticing that the method mentioned above reduces but does not remove the overshoot, this work proposes the following modification. Such modification uses a compression factor to reduce the fuzzy range of the PID gains, which smooths the control signal and makes it more suitable for applications involving UAVs, in which the convergence is desired but not with great overshoots.

Therefore, the actual adopted gains $K_{P,min}$, $K_{PI,max}$, $K_{DI,min}$ and $K_{D,max}$ are obtained as follows:

$$K_{P,min} = K_{PI,min} - \frac{K_{PI,min}}{7}\,, \quad K_{P,max} = K_{PI,min} + \frac{K_{PI,max}}{7}\,, \tag{III.3}$$

$$K_{D,min} = K_{DI,min} - \frac{K_{DI,max}}{7}\,, \quad K_{D,max} = K_{DI,min} + \frac{K_{DI,max}}{7}\,. \tag{III.4}$$

For the fuzzy set present in this work, the Membership Functions types are chosen as triangular for the outputs and Gaussian for the inputs. This is usually the setup presented in most of the literature cited by this work regarding PID gains tuning with fuzzy logic. The existing works were used as a start point regarding the rules, and the rules were defined according to the UAV's desired behavior. Figure

III.8 below shows the fuzzy set and its limits.

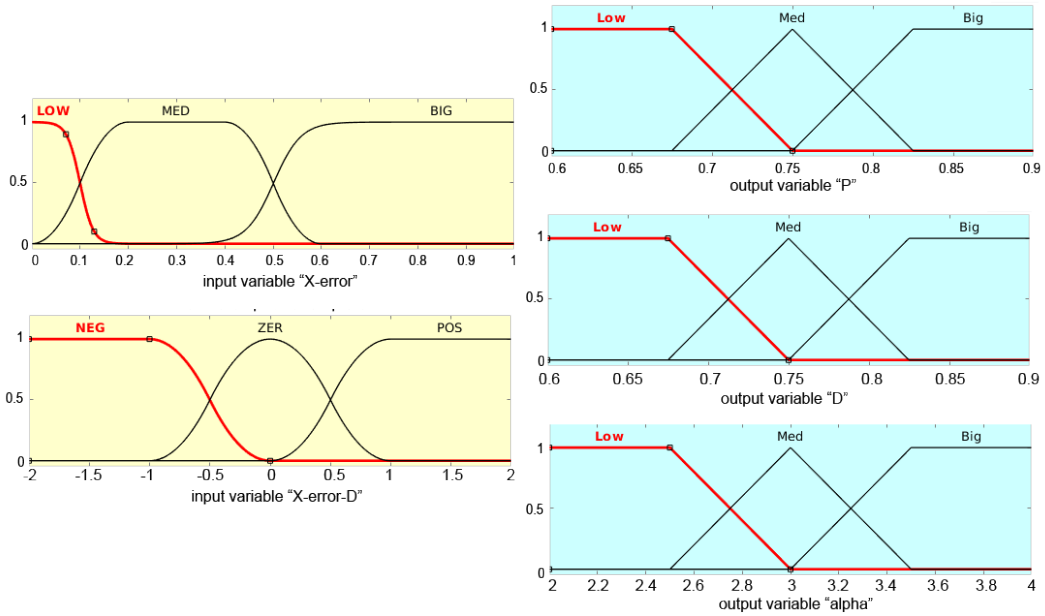


Figure III.8: Fuzzy membership functions.
From [5]

The left membership functions represent the system's inputs of the x-error and its derivative, while the right column exhibits the three outputs, namely the variation on PID gains. The system considers the same fuzzy sets for all X, Y, and Z errors. In the same development line from [27], this work developed its rule table, which can be seen below in III.3.

Table III.3: Fuzzy Rule Table from [5].

	error		
error - D	**Low**	**Medium**	**Big**
Negative	LBB	MMM	BLL
Zero	BBM	BML	BLL
Positive	LBB	MMM	BLL

III.3 Fuzzy Gain-Scheduling PID for Position and Altitude Controllers

Another great aspect of UAV control goals is to provide stability to the horizontal path and altitude control. Despite PID's good performance, different requirements may be present at different mission stages. The main contribution of author in the research [49] was developing a strategy based on a fuzzy-gain scheduling mechanism to adjust the PID controller to stabilize both position and altitude.

UAVs can suffer from environmental disturbances, such as wind, magnetic fields, or uncertain disturbance effects, such as sensor noise or payload during inspections. PID controllers have a simple structure. However, the nonlinear characteristics make the design and tuning of the PID complex. Furthermore, it cannot ensure closed-loop performance in different conditions of flight. Fuzzy controllers applied to UAVs undoubtedly have some advantages in performance evaluation compared to PID controllers, such as dynamic property and steady-state error [38].

In this work, there were presented two fuzzy gain schedulers evaluate the use of fuzzy logic to introduce this effect. The first controller changes the altitude controller gains as the altitude error changes, and the second one adjusts the position controller gains to reduce its action as the height error, where the altitude error is used to control the position performance. This fuzzy-logic controller changes the performance following the UAV application requirements.

Turning the PID gain scheduled can improve the UAV performance during the flight and change the flight's characteristics according to the performed task [79], [33], [20], [5]. For example, moving the UAV in a given direction during an inspection could cause a collision. To avoid such issues, it might be desired that the UAV prioritizes the control in this direction instead of considering every and each direction in case of a wind blow. Another example is during take-off, where the UAV must be well configured to prioritize the flight stability in the vertical direction. In another scenario, while the UAV is carrying a load, the controller can focus on a certain direction in case of instability to enhance flight security.

As this controller uses the same UAV model as the other developed studies, the modeling is the same presented previously from the Quadrotor. Here, it is used a classic UAV control system, composed of different stages, position, orientation, and altitude control, to control the UAV position and orientation. Each PID controller is tuned using classical methods. Once the full UAV model is known, then a fuzzy scheduler is applied to increase or decrease the gain of the altitude control and the position control. This allows the controller to adjust to environmental changes and can be seen in a simplified diagram

of the control stages in Figure III.9.

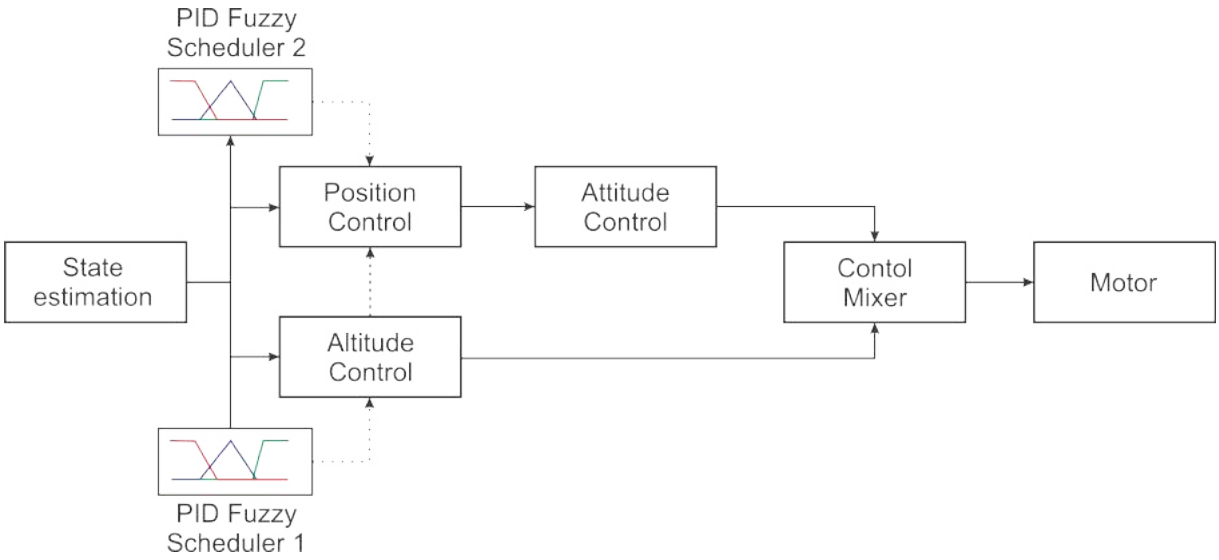


Figure III.9: Simplified diagram of the control stages.
From: [49])

The diagram shows that the PID Fuzzy controllers are independent, and because of this, it is possible to test the first controller separately and then evaluate them all together. The first step is to work on the altitude controller, to achieve an adequate response to the altitude error by adjusting the ideal controller for high and low-altitude error environments. After the altitude controller performance was within the desired objective goals started designing the position controller and fuzzy scheduling for this controller. Each fuzzy implementation had a different objective related to the variable being controlled and its requirements. Each fuzzy controller measured the variable errors to determine the scheduling mechanism, as can be seen in Figure III.10.

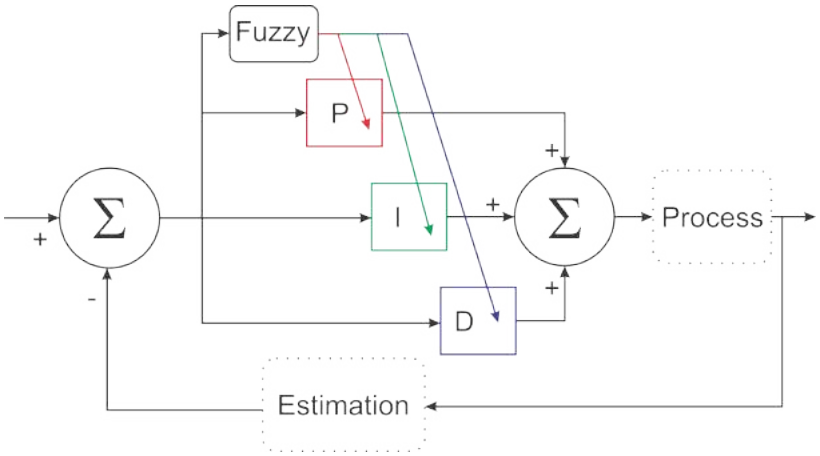


Figure III.10: Fuzzy logic implementation with independent objectives.
From: [49])

The fuzzy schedules are proposed as independent and different because this improves the aircraft's performance and introduces safety characteristics. The primary variable is the aircraft's height and stability in this direction. Once the aircraft is stable, controlling the yaw keeps the aircraft from spinning, and it is essential because it is not possible to perform position control without stable yaw. The last objective is to prevent the aircraft from losing its desired position. If the aircraft has plenty of thrusts available, this priority may not be relevant once enough power is available to control all these parameters. However, in a situation, for example, where the battery may be low, and there is a wind gust or payload change, the power available may not be sufficient to provide a stable response for all the variables. These scenarios may be beneficial to reducing control input for the position, such as allowing more space for height and stability control. In a scenario where there is the presence of a continuous high load, if a good prioritization is performed, the UAV would start losing position, then it would start spinning to lose height at the end.

The quadrotor altitude control design uses a classic strategy which is applied by linearizing the plant loop to obtain its reference model. This simplified model is used to provide reference performances for two operating conditions. The first condition is used as the main control law and is a high-speed and aggressive controller. The second condition is a smoother controller, and the fuzzy logic switches to this following the observed errors. The plant model obtained is shown in Equation III.5.

$$\frac{(1.98 \cdot z + 1.98) \cdot 10^{-4}}{z^2 - 2z + 1} \tag{III.5}$$

The two control laws (L1 and L2) are obtained using the frequency response method in order to achieve a stable controller and good response times. The parameters are shown further in the results' chapter, in Table IV.2.

The fuzzy rules are defined as depicted in Figure III.11 for the proportional gain, Figure III.12 for the integrative gain, and Figure III.13 for the derivative gain. They are built using a Mamdani inference method [71], and are built manually under the expected errors. With these fuzzy sets and rules, the controller can transiation between the two control laws, depending on the error and the error rate. The rules are also evaluated in different scenarios, such as response to the unit degree, ramp, and noisy input.

For the position controller, the fuzzy set applied uses a different approach: it operates as a fast position and robust controller during normal operation, but when the altitude error grows too great, the gains are reduced to prioritize the delivery of power to the altitude controller. The concept behind this strategy is to prevent the control signal from being saturated by the position controller in case of sudden

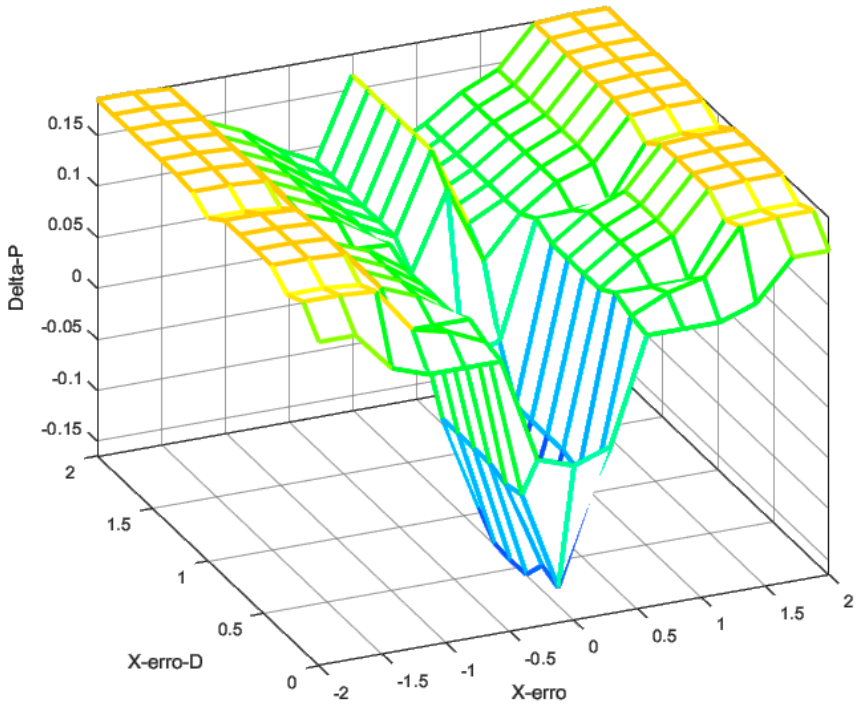


Figure III.11: Fuzzy surface for proportional gain.
From: [49])

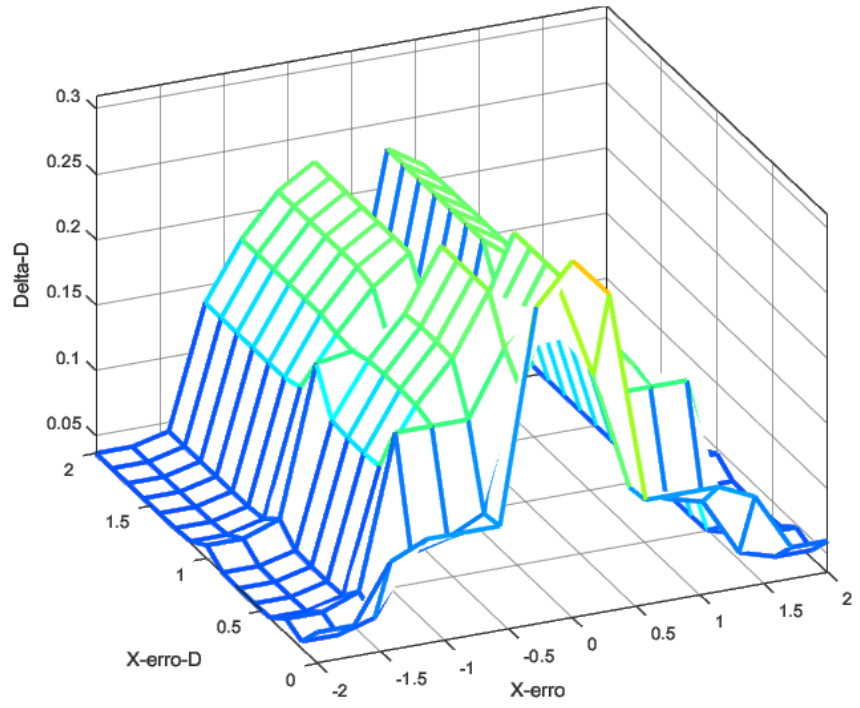


Figure III.12: Fuzzy surface for derivative gain.
From: [49])

altitude disturbances. Figure III.14 depicts the control surface, and, as the altitude errors increase, the surface behavior changes, moderating the controller. In other terms, as the position error increases, the gains from the controller are enhanced.

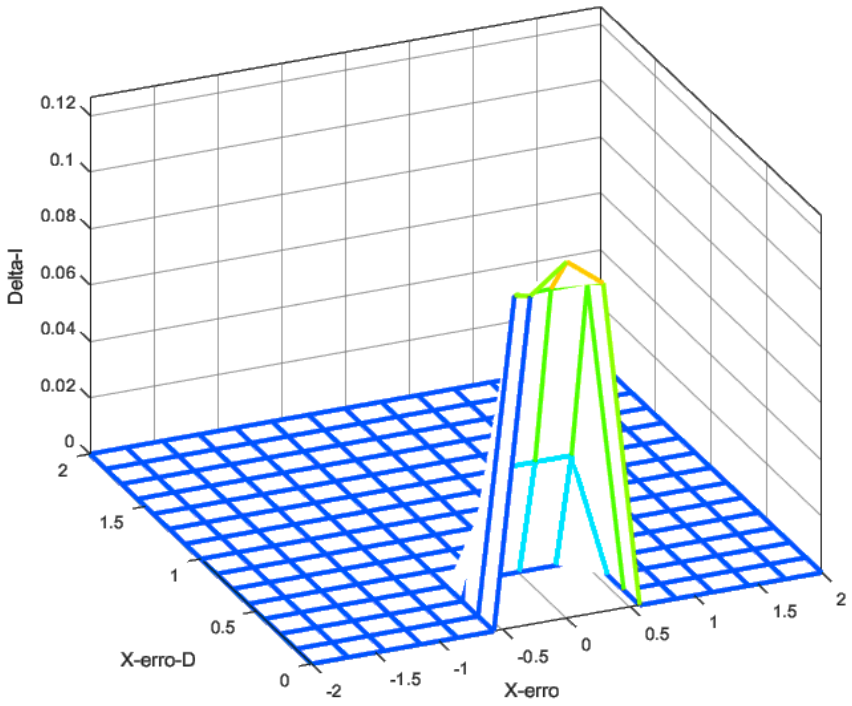


Figure III.13: Fuzzy surface for integral gain.
From: [49])

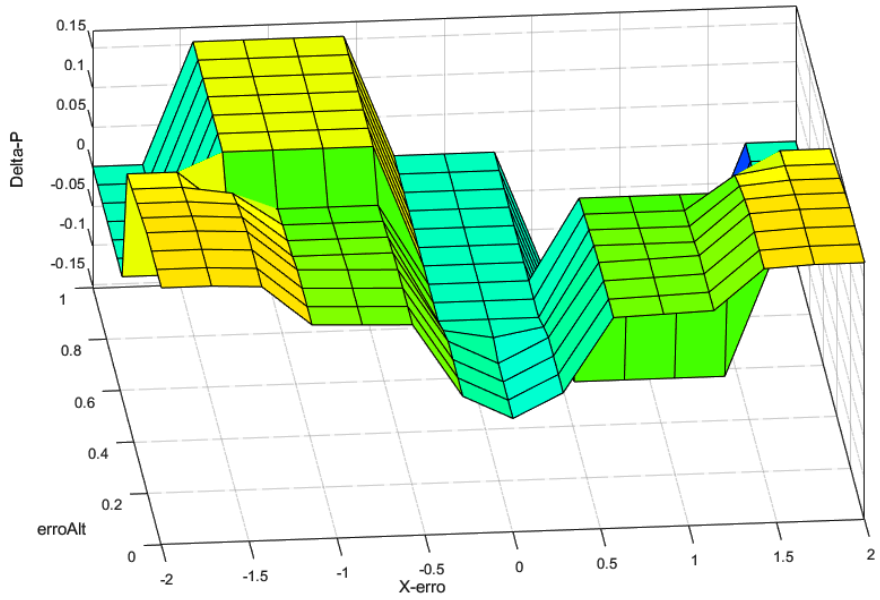


Figure III.14: Fuzzy surface for position control-proportional gain.
From: [49])

In this paper, the evaluation of the developed controllers is done by simulating the flight operation in a virtual world environment, using the same configuration of Induscon's paper (reference [27]), the computer where the simulation was executed was different, however. With a Intel Core i7-5500U CPU @ 2.40GHz x 4, Intel HD Graphics 5500 (Broadwell GT2) and a 15.6 Gb of memory.

Chapter IV Results and Discussion

This section will present the results accomplished in each of the developed works separately. The proposed work 1 refers to [27], proposed work 2 refers to [5], and proposed work 2 refers to [49]. The last section of this chapter deepens on the results not published.

IV.1 Hybrid PID-Fuzzy Controller

For the generation of results in [27], as mentioned before in the last chapter, four tests were taken into consideration, with different distances and paths.

The trajectory test T0, where the UAV is in its initial position, was performed by using both the classical PID and the PID-Fuzzy stabilization controllers. The Test T1 intended to verify the vehicle's performance over small distances. In that case, some fuzzy models and also PID have shown a considerable overshoot in the output response in at least one test. This behavior was not observed after using the proposed controller. Table IV.1 describes the mission times for each test scenario.

Table IV.1: Quantitative data acquired from simulation tests.

	Test Nr.		
PID	**1**	**2**	**3**
T1	2.62 s	1.77 s	1.72 s
T2	2.17 s	2.43 s	2.23 s
T3	3.15 s	2.04 s	3.28 s
T4	3.87 s	3.67 s	3.54 s
Fuzzy-gain			
T1	1.32 s	1.33 s	1.59 s
T2	1.40 s	2.05 s	1.46 s
T3	1.98 s	3.29 s	3.70 s
T4	3.29 s	3.69 s	3.42 s

The test T2 gives an overview of the intermediate trajectory. Here, an overshoot was again noticed in the test when using the PID controller, and a significant amount of time was taken to regulate its position. Since the fuzzy model had no overshoot characteristics, it has not taken extra time to centralize its position also. Thus, when compared to the results obtained with the classical PID controller,

there is a satisfactory gain in the motion performance.

The test T3 aims to evaluate the flight performance for long distances and high velocities. In this scenario, both controllers had overshoot issues. The fuzzy controller took approximately 0.4 additional seconds to stabilize when compared to the PID controller in two tests. Despite the overshoot behavior, the PID controller has taken some time to regulate the UAV's position to the desired point, while the fuzzy-gain controller has not.

The final test T4 uses a diagonal trajectory through which the UAV was guided, which is also the most challenging due to its range and velocity. The classical PID and the fuzzy-gain controller had similar results, such as the ones in Test T3, both with a considerable amount of overshoot in the response. In the PID implementation, an additional time period was observed to regulate the UAV's position to its desired point. Nevertheless, in what concerns fast regulation, the fuzzy-gain controller performed better for this last test.
Figure IV.1 shows the position and velocity results for the PID control implementation in the tests. In order to compare the improvement achieved by the fuzzy algorithm applied to the PID controller, Figure IV.2 shows the same set of results. The fuzzy gain enhanced the controller performance on the main test, reducing overshoots and undershoots in the output response.

A second simulation trial was conducted with the primary objective of testing the diagonal movement performance of the fuzzy-gain controller. Multiple lengths of diagonal motions were tested, with a quarter, half, and the entire length of the test size, respectively, as can be seen in Figure IV.3.

By evaluating the presented results, it is possible to notice that the combined fuzzy algorithm has improved the UAV general performance. These simulated results indicated that the method's practical implementation is feasible for deployment in real scenarios (real hardware).

As was previously mentioned, this part of the results was published in [27]. A video containing the simulation results of the proposed Cascaded-PID-Fuzzy controller method in a UAV is available at https://youtu.be/9c_JVZtPlU8

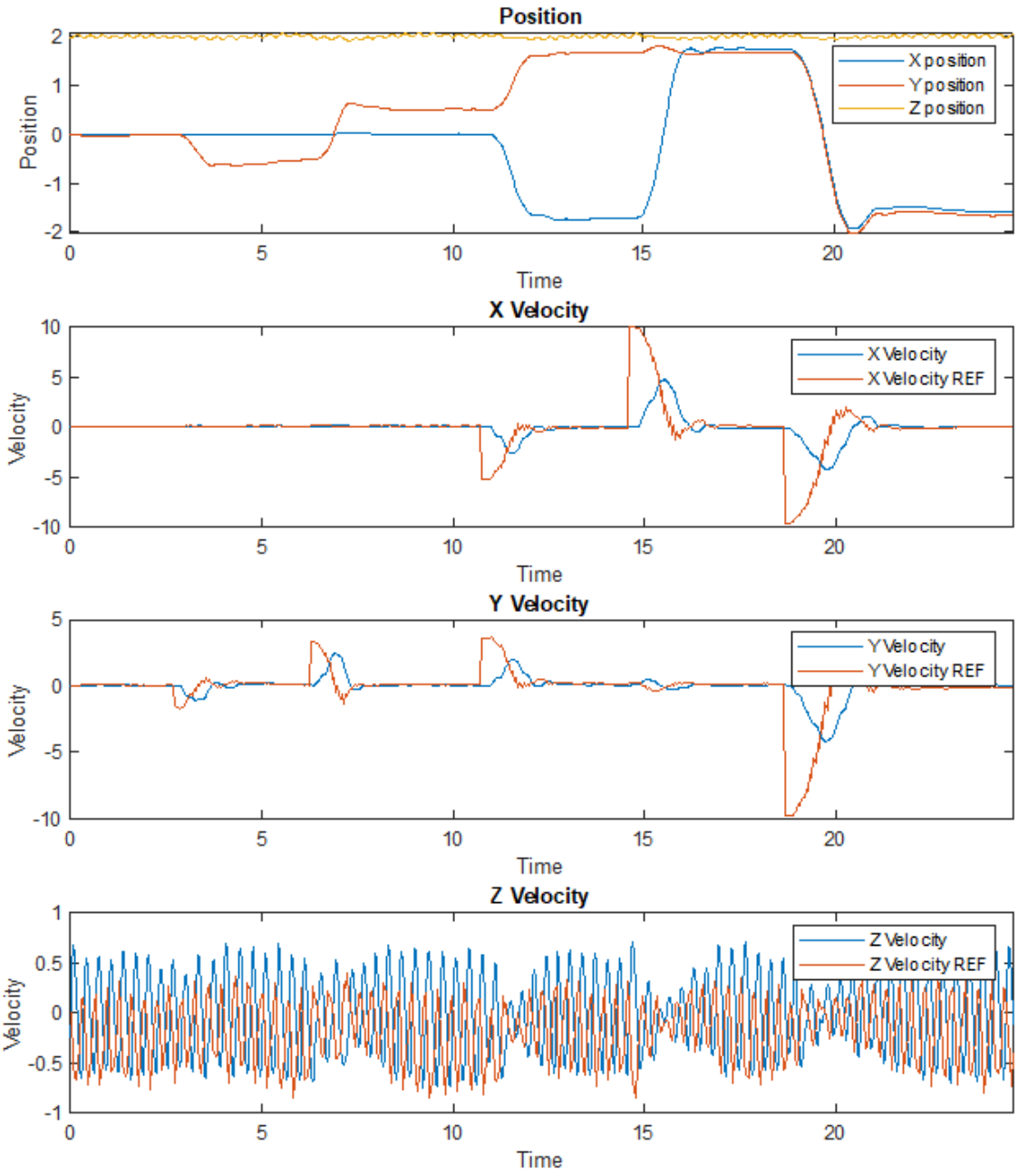


Figure IV.1: Position and velocity for one row of tests by implementing the PID controller. From: [27]

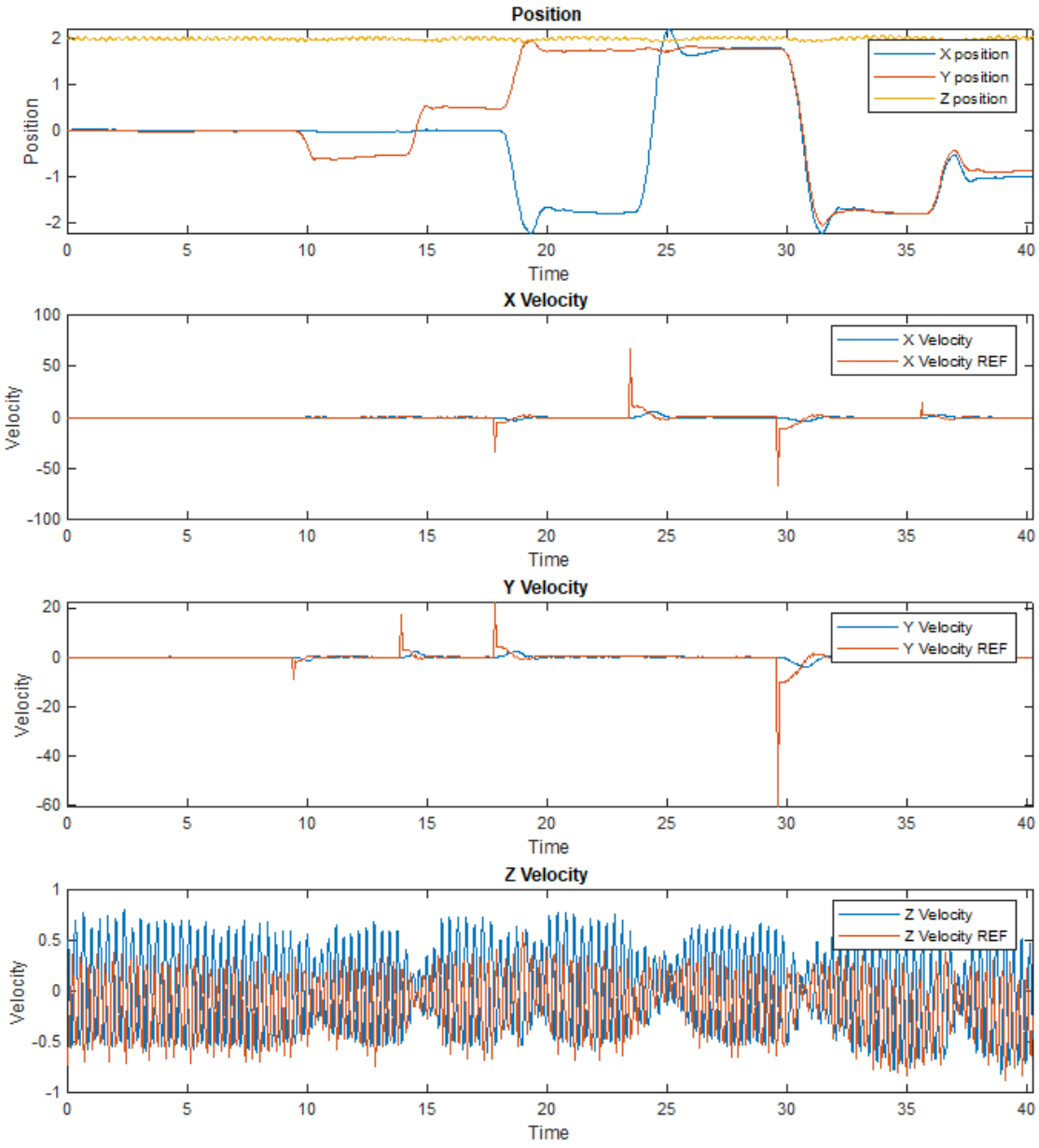


Figure IV.2: Position and velocity for one row of tests by implementing the Fuzzy-gain controller. From: [27]

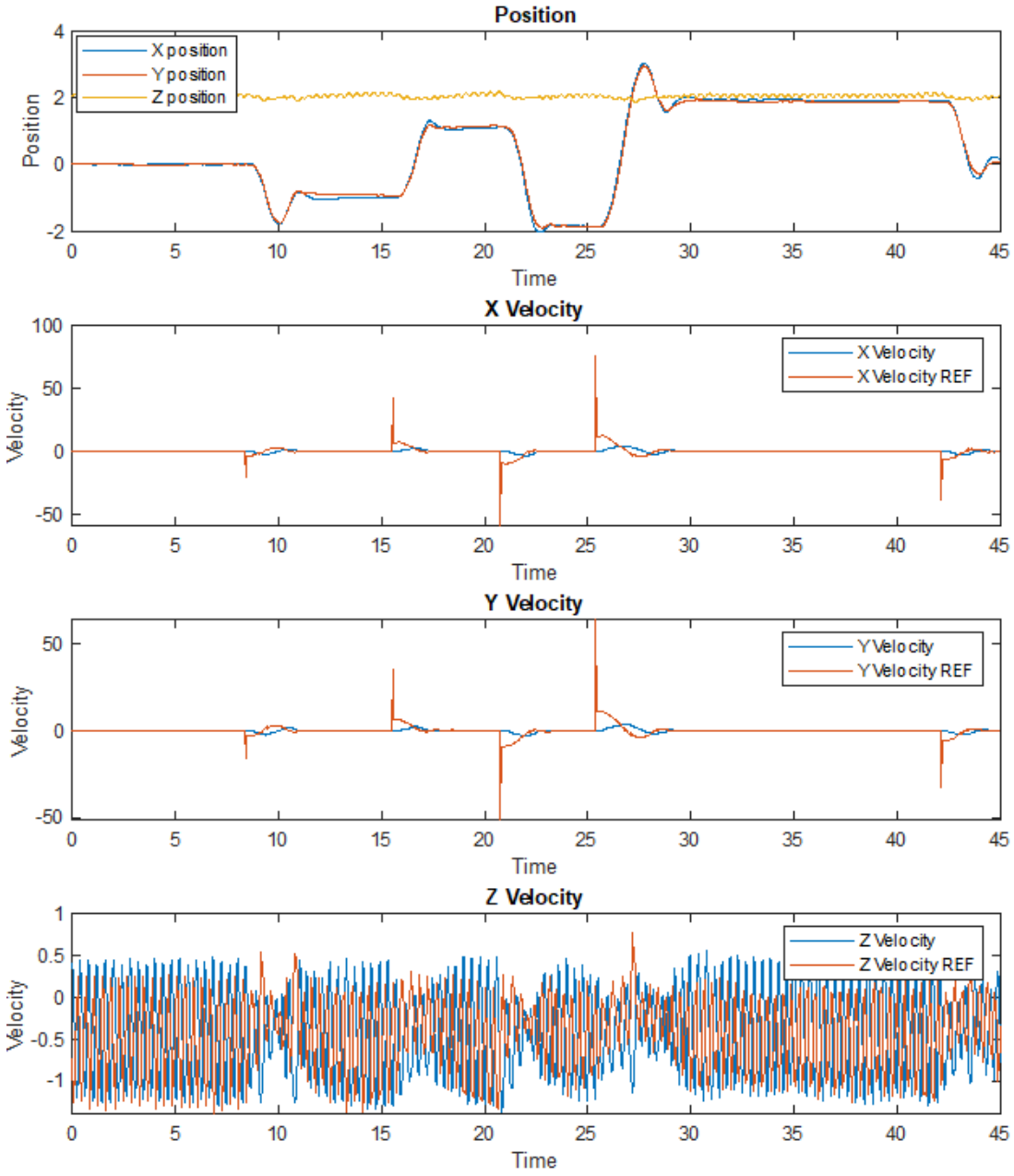


Figure IV.3: Diagonal motion test on Fuzzy-gain controller.
From: [27]

IV.2 Motion Control with Fuzzy Tuning of Cascaded-PID Gains

In this work, three tuning methods for the cascaded-PID were evaluated, as depicted in Figure IV.4. As already pointed out in [89] and [39], the Zhao/Larson tuning method achieves a similar convergence time as the Ziegler-Nichols method but with a lower overshoot and faster accommodation profile. For aerial vehicle applications, having a fast convergence is desirable. However, having a good accommodation property is more important, as the UAV usually needs to read data with onboard sensors.

In the tests, the UAV is commanded to go from one position, 2 meters at East from takeoff position, to another position, less 2 meters. The command to move was sent around 1.6 seconds after starting recording the logs and building the graphs.

The proposed solution suits better for UAVs, as it can achieve faster accommodation with less overshoot. Considering that the acceptable error is 1 centimeter, this solution reaches accommodation in 6.7 seconds, 4.1 seconds less than the Zhao/Larson tuning (with 10.8 seconds), and 13.3 seconds less than the Ziegler-Nichols tuning method (with 20.0 seconds).

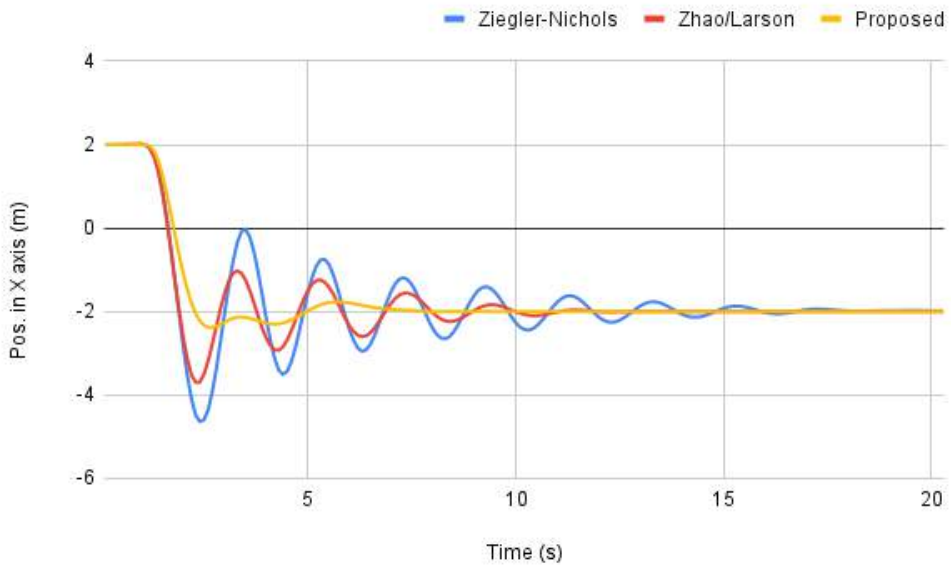


Figure IV.4: The comparison among Cascaded-PID tuning methods. (From: [5])
From: [5]

The method is also compared to ArduPilot's built-in controller in *guided* mode, which allows the user to set target position commands, updates the velocity control at a rate of 50 Hz, and is possible to send a ROS message of the target position through MAVROS. It is also worth mentioning that the ArduPilot controller is very advanced and has been developed and tested for many years, with a big

team of contributors and also developers.

The result can be seen in Figure IV.5, in which it is possible to conclude that the ArduPilot achieves a faster time of accommodation. However, in the absolute error, the proposed method begins 1.5 seconds faster, with an overshoot smaller than 0.5 meters. This is also illustrated in Figure IV.6.

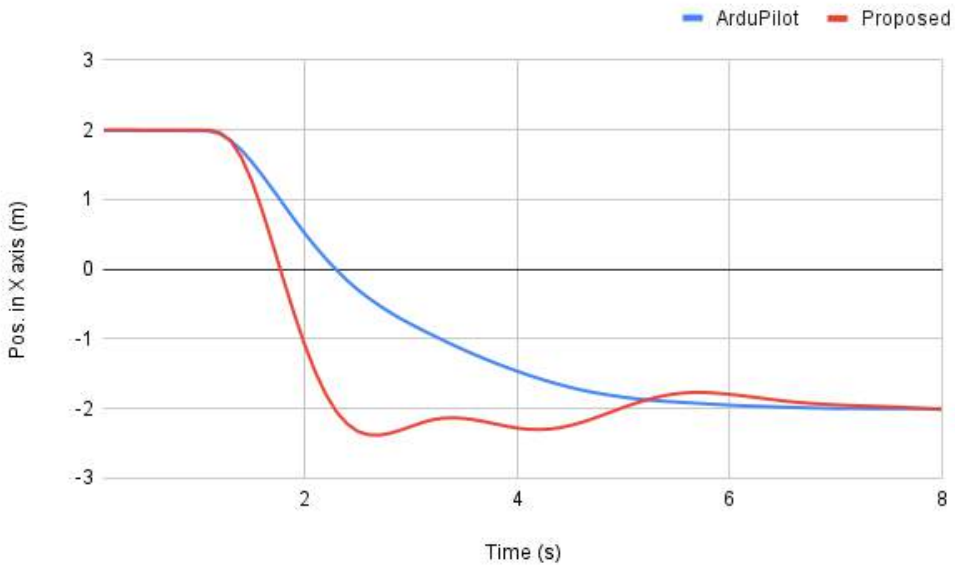


Figure IV.5: ArduPilot and Proposed method compared.
From: [5]

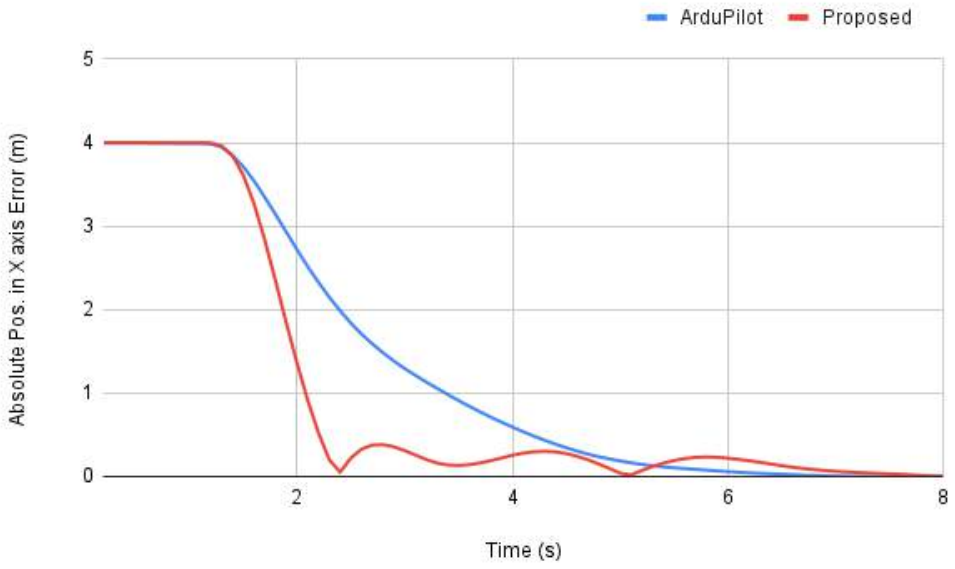


Figure IV.6: Absolute Error comparison between ArduPilot and the proposed method.
From: [5]

Still, compared to the ArduPilot controller, the accumulated error can be analyzed in Figure IV.7.

This measures how much the UAV has been away from the desired position over time. It is expected that the accumulative error increases in time, but on these results, its possible to see that the upper curve (representing the ArduPilot controller) has an accumulative error of *ca.* 25 meters greater than the proposed method. The ArduPilot stabilizes the error in almost 6 seconds, while the proposed method stops increasing later. In this experiment, the proposed method shows itself more accurately in the position in x, which should be replicated for the position in y since it uses the same control principle.

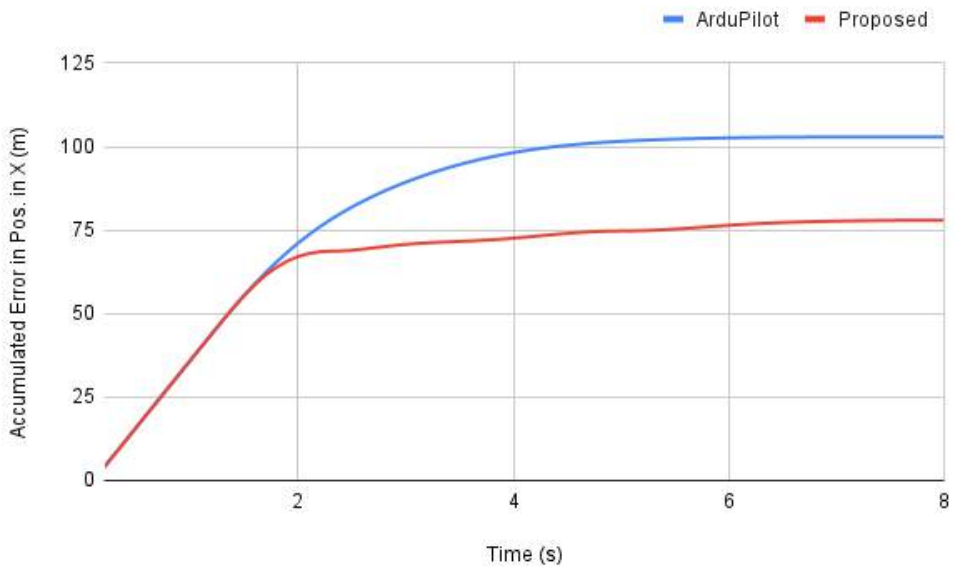
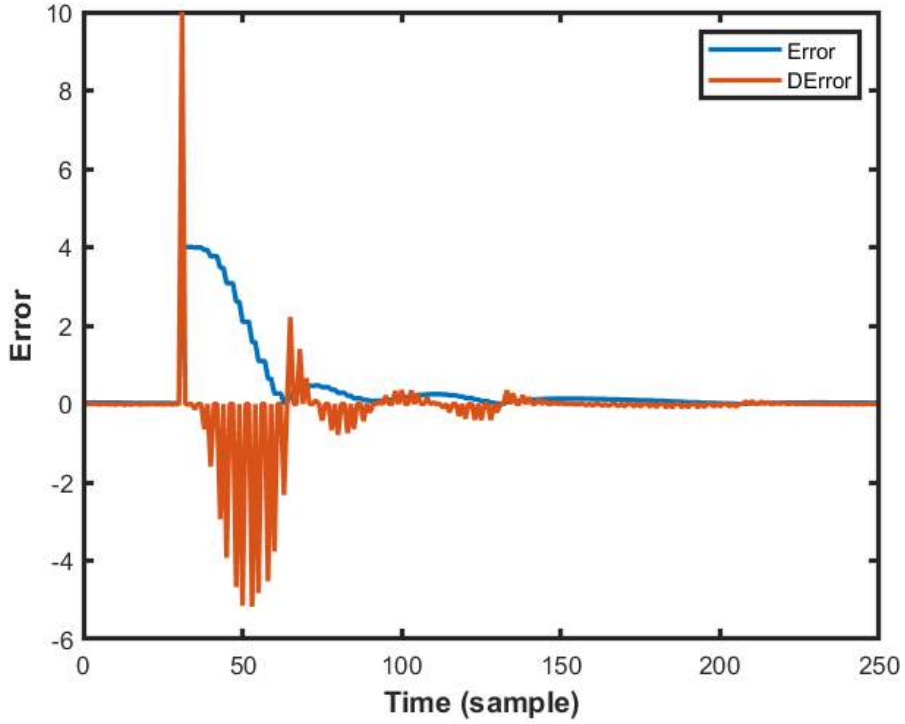


Figure IV.7: Accumulated Error comparison between ArduPilot and the proposed method. From: [5]

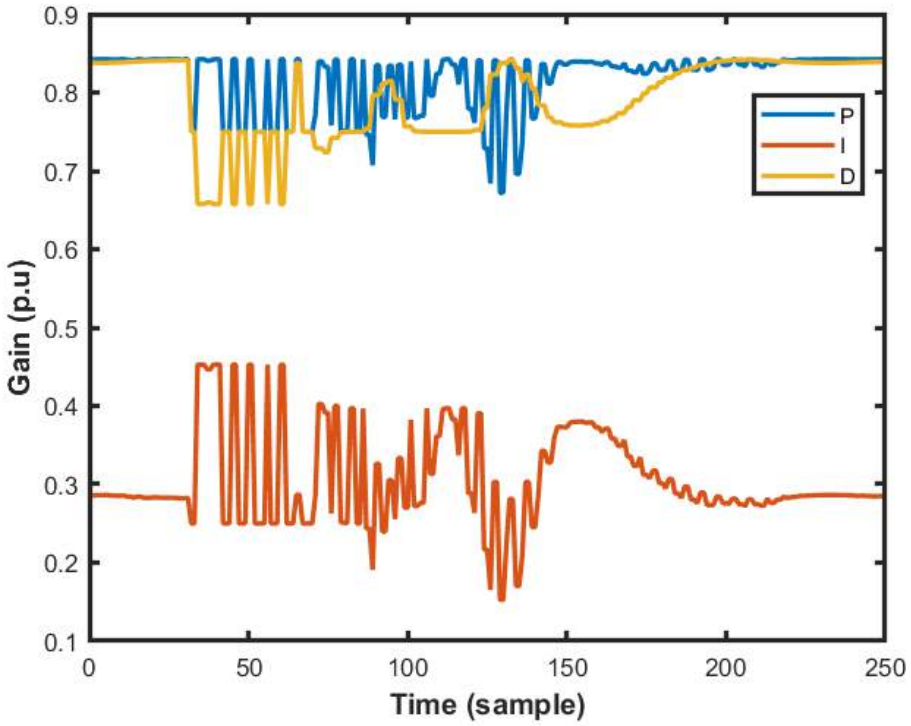
The position error and its derivative in absolute values can be observed along with the PID gain chosen by the fuzzy controller in Figure IV.8. A discontinuity generated by the variable initialization caused a peak of the derivative error in the beginning and thus should not be considered.

The variation of the PID gains demonstrates that the Fuzzy Logic was properly configured as in the beginning when the error exhibits a big value, the derivative gain was lowered, but it approaches every time the error approaches the setpoint. Further, the derivative gain tends to increase if the error persists for a long time, being smoothly reduced in case the UAV stays around the setpoint. The proportional gain follows the error directly.

The oscillatory behaviors of the P, I, and D in Figure IV.8(b) are probably caused by the measurements of Figure IV.9(a), which are also propagated to the velocity command. Even though this oscillatory behavior is undesired, it didn't affect the measured velocity in Figure IV.9(a).



(a)



(b)

Figure IV.8: Measured errors and Fuzzy PID gains. (From: [5])
From: [5]

The measured and commanded velocity can be compared in the mentioned Figure IV.9. Because the commands are sent at a rate 2.5 times greater than the measurements, the sample times are different, but the time range is the same in seconds.

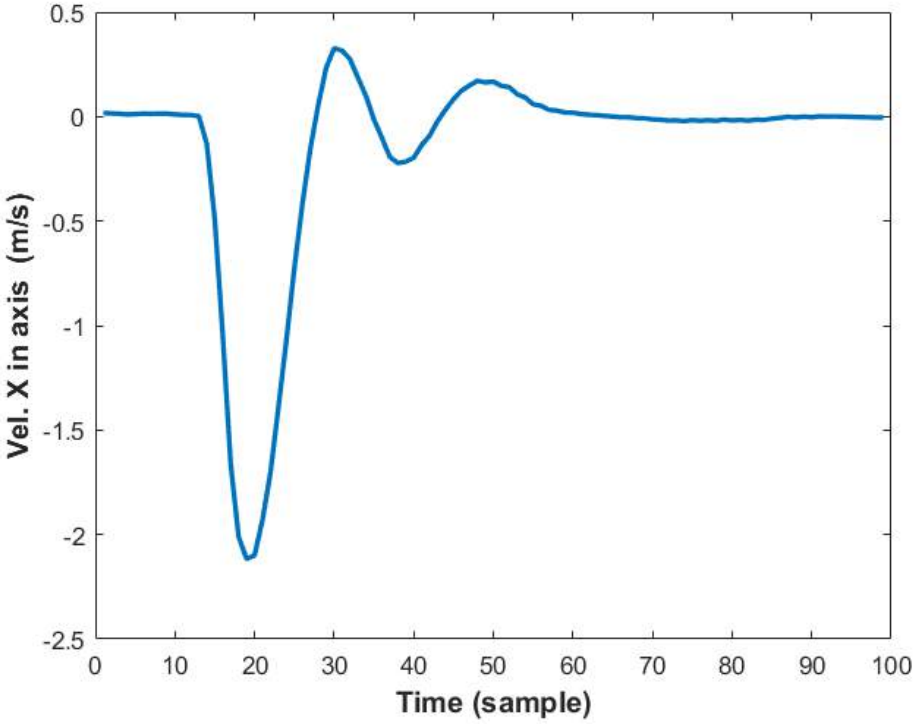
Because the velocity commands are sent before the UAV reaches the desired velocity, the measured velocity of the UAV does not achieve the desired commanded velocity. The UAV also has a limited maximum velocity, set to 3 m/s. The fuzzy-cascaded-PID controller calculates the desired velocity greater than three meters per second, as at the beginning of the test when the error was big, and the UAV was steady. The UAV is thus unable to achieve the desired velocity immediately because of inertia.

From another point of view of the results, using the mean values of the Fuzzy Logic gains, the proposed method is compared to fixed PID gains in Figure IV.10. The variation of the fuzzy gains oscillates less than the mean-gains approach, shows a smaller overshoot on the first attempt to stabilize, and achieves the desired position on X-axis before the mean-values method. In contrast, the mean values of fuzzy gains present reasonable results. In this same test analysis, the proposed method has its accumulated position error on X-axis slightly smaller than the fuzzy gain's mean values, in Figure IV.11. At 7 seconds, it's possible to notice that the accumulated error from the proposed method is greater than the mean-values method. At this point, the controller is trying to stabilize for the last time, with no further oscillations, while the mean-values method oscillates and, thus, achieves the objective position briefly.

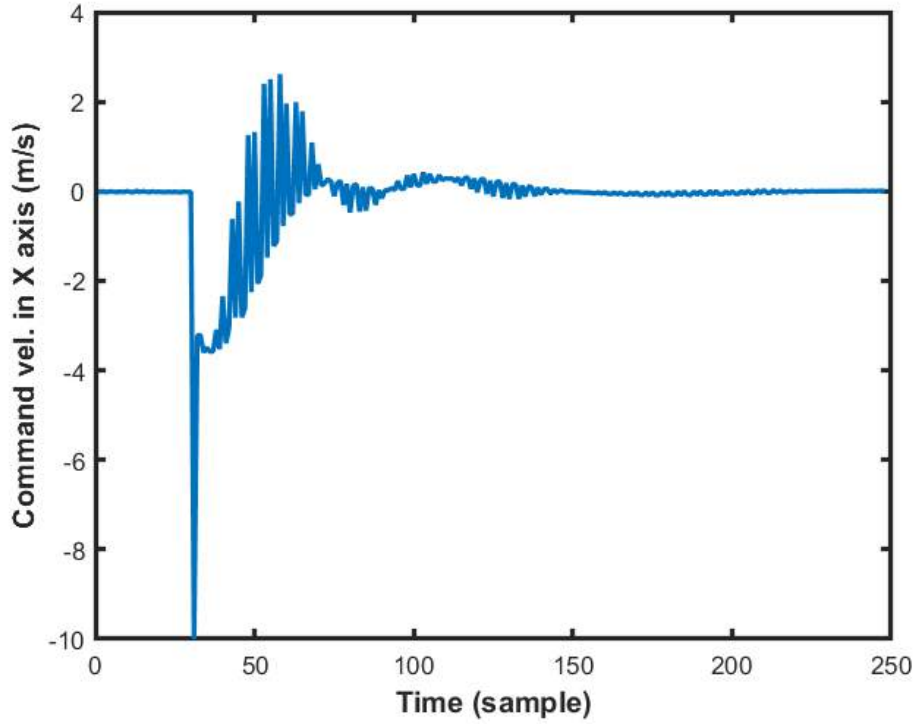
The last simulation was performed when a wind of 15 meters per second was introduced. The results can be seen in Figure IV.12. The proposed method may perform better than the approach using only the mean values. In addition, it achieves nearly the same time of accommodation as the Ardupilot solution, with faster convergence.

In the real implementation, the multi-rotor Bebop 2 by Parrot was used, and it is shown in Figure IV.13. This UAV's system has a compatible network interface for ROS, by furnishing the required sensor information as well as the control interface.

For the Altitude Control Tests in real flight, only the height control is used, while the original Flight Management Unit (FMU) algorithm still handles the position control and stabilization. This ensures safety once the UAV is stable and locked into position. The Parrot Bebop 2 is shown during the flight test in Figure IV.14. The Bebop 2 weighs 500 g, and has 32.8 x 38.2 cm of dimension. According to the vendor, its take-off speed is six m/s, and the maximum speed can reach 57 km/h. The battery gas



(a)



(b)

Figure IV.9: Measured and Commanded velocities.
From: [5]

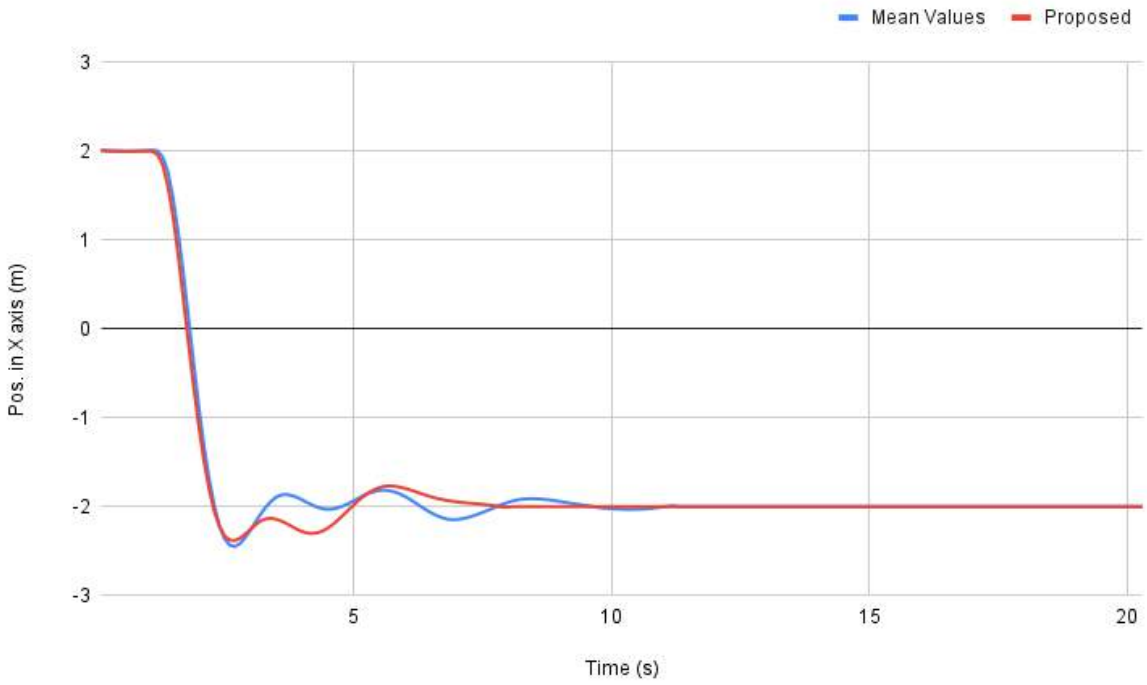


Figure IV.10: Comparison between the proposed method and using the mean values of the fuzzy logic gains.

From: [5]

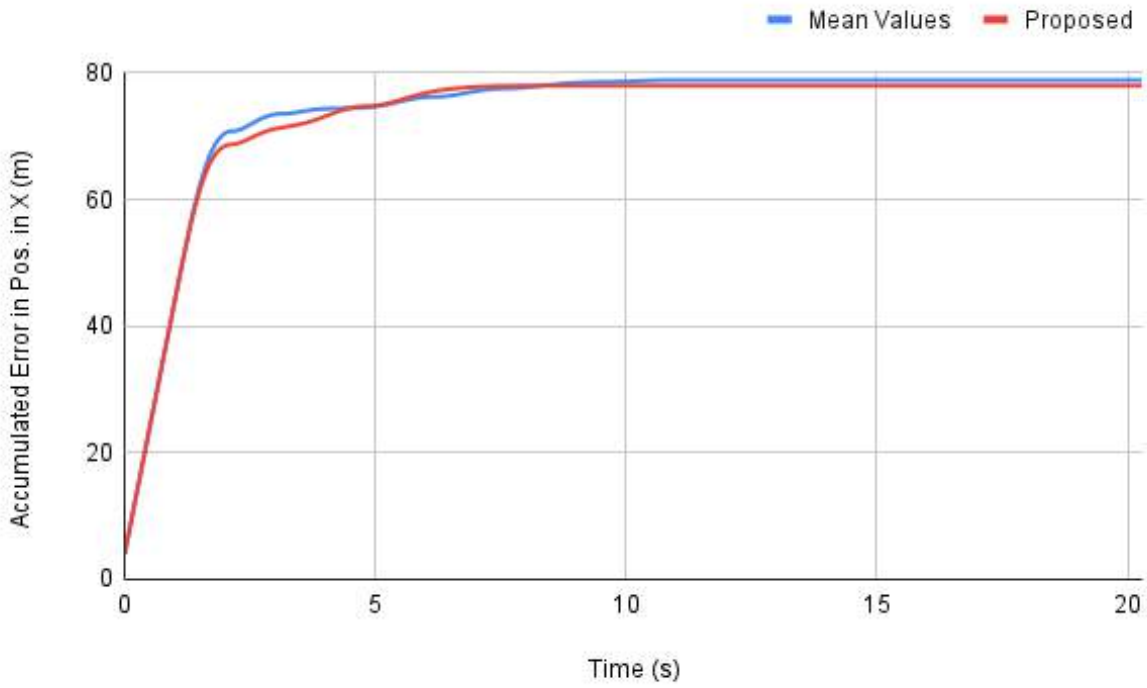


Figure IV.11: Accumulated Error comparison between the proposed method and using the mean values of the fuzzy logic gains.

From: [5]

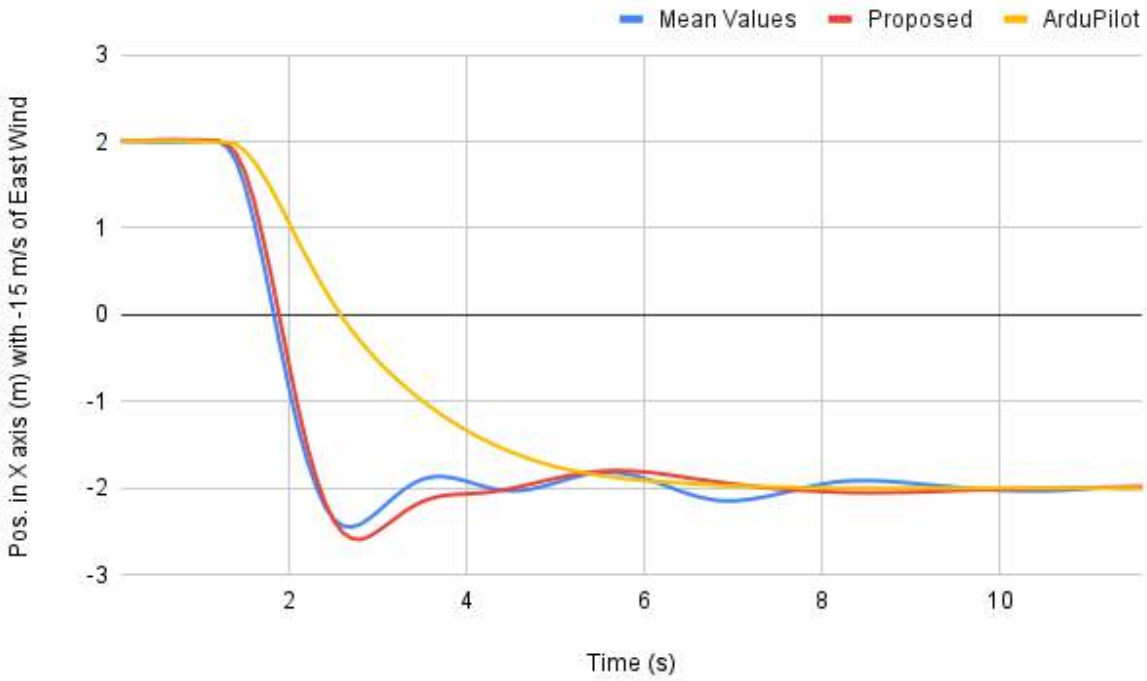


Figure IV.12: Comparison among different methods in the presence of 15 m/s wind to the West.
From: [5]



Figure IV.13: The Parrot Bebop 2.
From: [5]

is 2700 mAh, which supports approximately 25 minutes of flight. It also has an in-built camera, GPS and can resist winds until 59 km/h, giving a good versatility to run any test.

To avoid discontinuous movements, this experiment tests the method using ramp profiles from one altitude setpoint to the next one. The error measurements and the control signal are shown in Figure IV.15. The overall behavior of the control signal is represented by the dotted line, which is applied in discrete time instants.

The control PID gains estimated by the fuzzy algorithm are within the expected patterns and can be seen in Figure IV.16. By analyzing the plot, it is possible to see that the derivative estimation changes were the most aggressive compared to the others.



Figure IV.14: The UAV height control flight test.
From: [5]

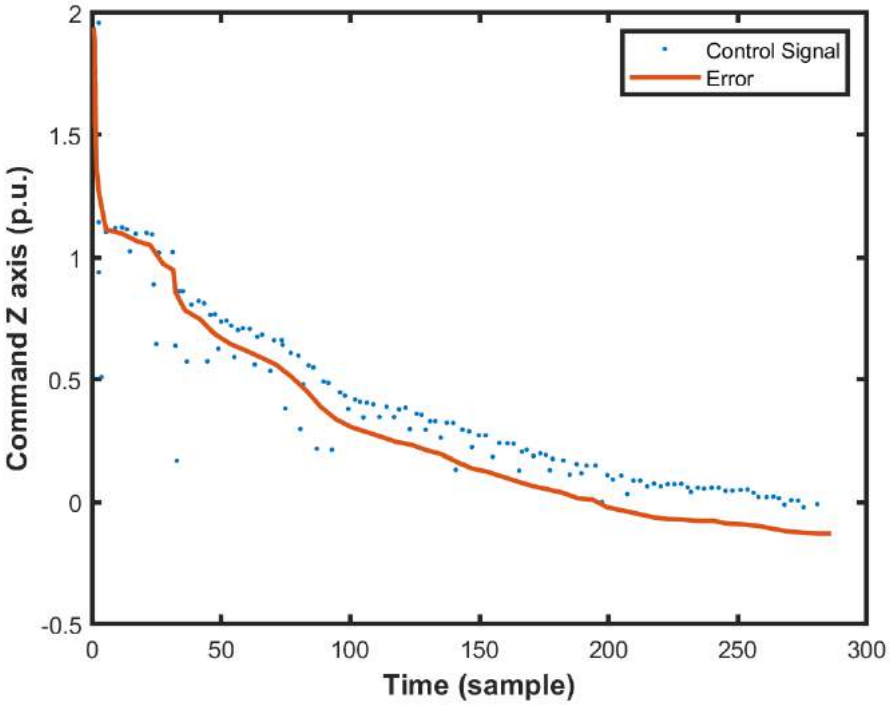


Figure IV.15: Control signal and error during the experimental trial.
From: [5]

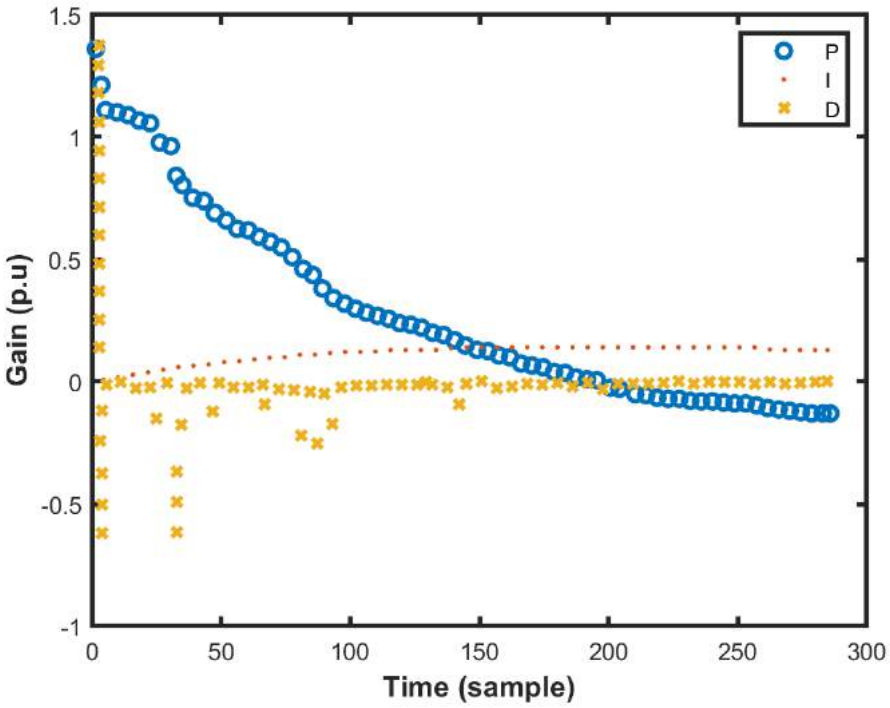


Figure IV.16: Individual control response during the experiment.
From: [5]

Table IV.2: Gain for the altitude control law.

	L1	L2
P	1.7	0.16
I	0	0.009
D	0.6	0.6

Table IV.3: Performance of the altitude controllers.

	L1	L2
Rise Time	0.12	0.2
Setting Time	1.0	1.6
Overshoot	20%	2%

IV.3 Fuzzy Gain-Scheduling PID for Position and Altitude Controllers

Based on the values presented at Table IV.2, it is possible to quantify the performances, as described in Table IV.3. As expected, the control law 1 (L1) had greater overshoot but shorter rise and setting times, if compared to L2, which is the smoother control law.

In Figure IV.17 and Figure IV.18 below, it can observed the qualitative response related to each of the control laws. Faster response means that the UAV responded to an environmental change faster. In order to avoid the overshoot, the fuzzy logic moderates the gains when closer to the target, and, additionally, integral control is only added when the error is low enough to require steady-state compensation.

As soon as the altitude controller is set, the position controller takes place and operates with a similar strategy. The main difference between the controllers is that the position controller uses a Proportional-Derivative (PD) controller, and the first control law is configured to produce a robust response, while the second control law gives a slower response and is used when the altitude error is very high and needs to be prioritized. The gains from control laws 1 and 2 are described in Table IV.4.

The fuzzy scheduler's strategy is meant to be very simple. The proportional control gain is set, so the UAV promptly responds to any change in the error, but it also produces an overshoot response. The fuzzy scheduler is responsible for moderating this gain when the target is closer, thus removing the overshoot behavior. The integral gain is only applied when both error and its derivatives are low,

Table IV.4: Gain for position control laws.

	L1	L2
P	0.24	0.7
I	0	0
D	0.1	0.05

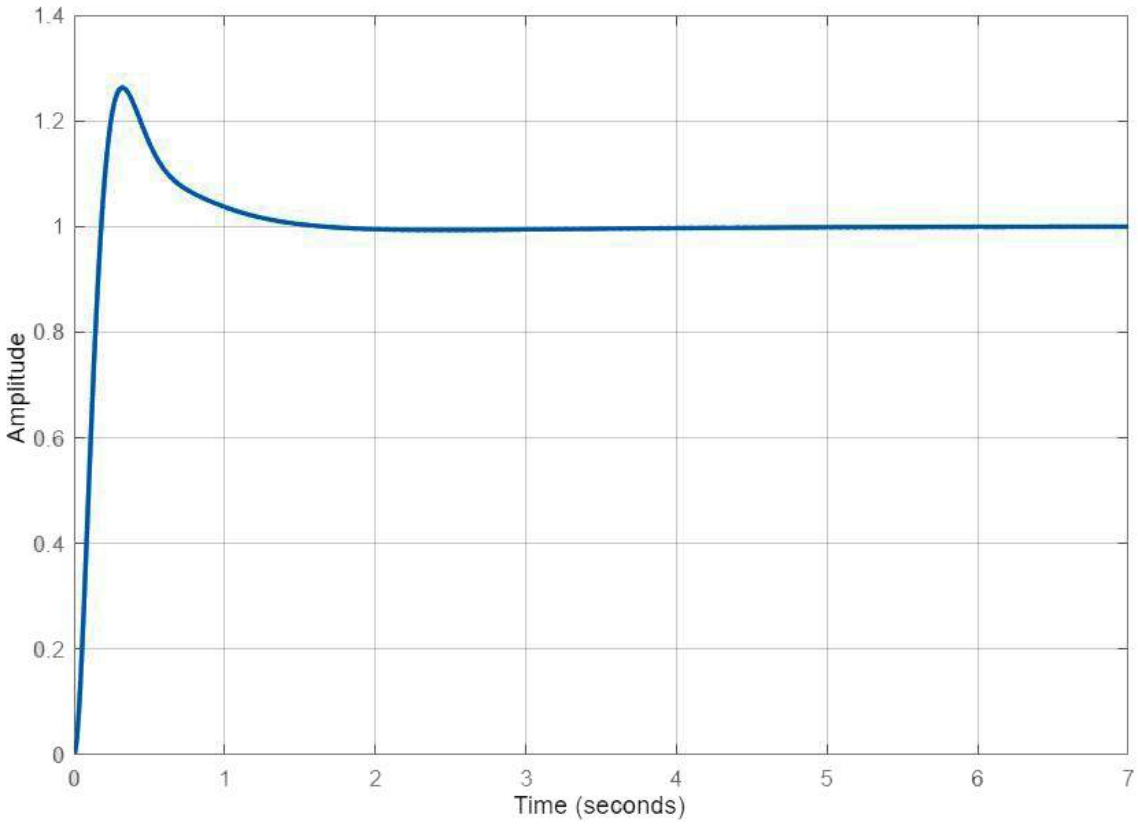


Figure IV.17: Altitude response to first control law.
From: [49]

meaning that the UAV should be closer to the target.

Below, in Figure IV.19, there are the X-error and its time derivative for the inputs of the fuzzy control, and in Figure IV.20, there can be seen the defuzzification outputs. The defuzzified outputs are presented for each control effort of the PID controller, and its values represent the changes in the control law in effect. The shape of the fuzzy sets produces a selective transition between control laws as the error level increases for the inputs and smoothly increases the PD gains to stable values as the error increases for the output. However, it might be remarked that the integrative error is only applied when small errors are present to ensure that steady-state errors are going to be corrected. However, no instability would be produced when combined with anti-windup.

The results show the evaluation of the fuzzy schedule control usage to improve the performance of an altitude controller in a simulated environment. This step increases the controller effort in response to larger errors while still allowing a smooth response. This step also evaluates and schedules the combined position and altitude gain in the simulated environment. This idea reduces the position controller effort when a high altitude error is observed while it is still possible to produce an adequate response. The height control scheduling is also evaluated in a practical scenario as a control experiment to validate

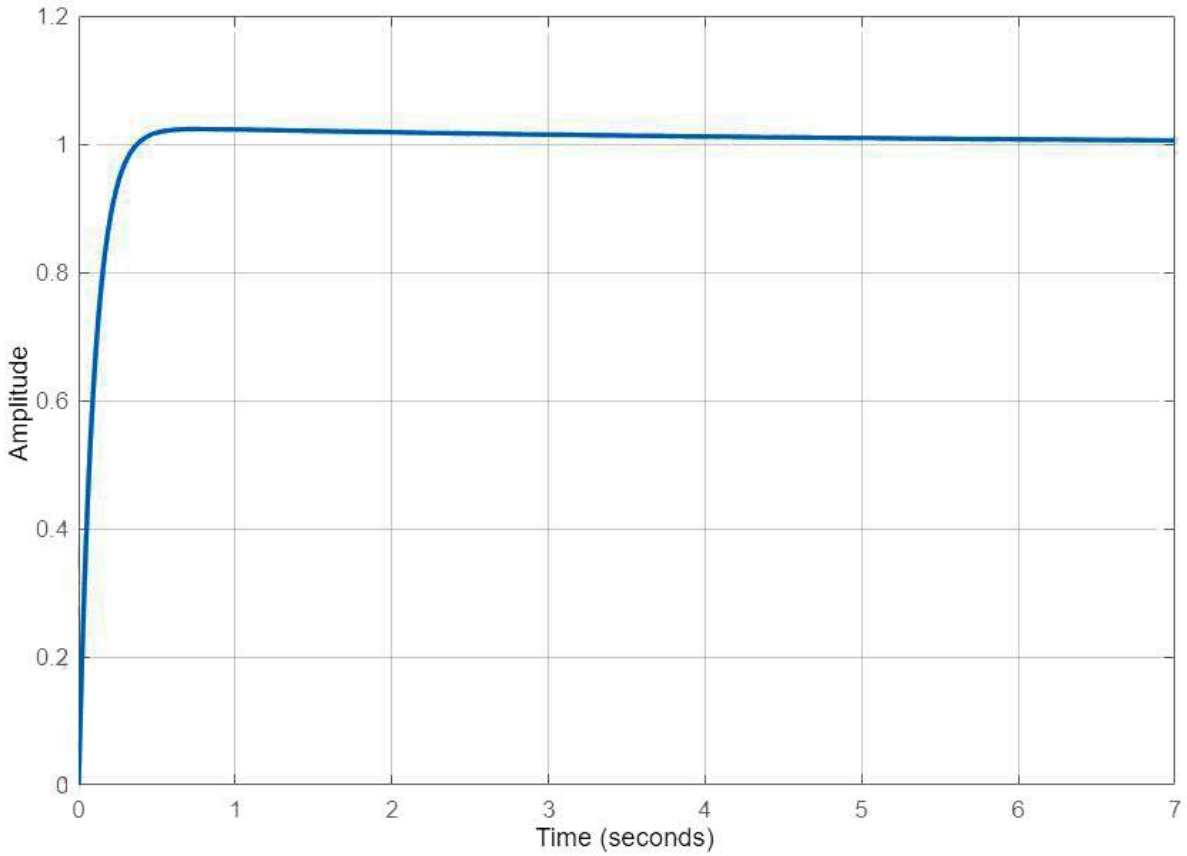


Figure IV.18: Altitude response to second control law.
From: [49]

the first simulation.

In the first test, a set of mixed inputs is chosen to evaluate the controller's performance when answering to increase and decrease errors with different slopes. This performance test summarizes the performances of the controller for the original and scheduling controller with and without noises. The altitude control result can be seen in Figure IV.21. Both the classical PID controller and the controller with the fuzzy scheduler are tested, for comparison purposes, with and without the presence of 10% of measurement white noise.

In Figure IV.22, there is a section of the test in detail. It is possible to observe the performance of each controller more precisely compared to the reference scenario. In Figure IV.23 and Figure IV.24 the performance along the X and Y axis can be also seen. These figures illustrate well the importance of adaptive controllers for UAVs. For example, in Figure IV.24, the fuzzy PID presented a slightly smaller error than the other control strategies. Another characteristic that must be mentioned is that the output obtained during stabilization with the fuzzy scheduler PID controller oscillated less than the other controllers, achieving the positioning objective in a more well-behaved profile.

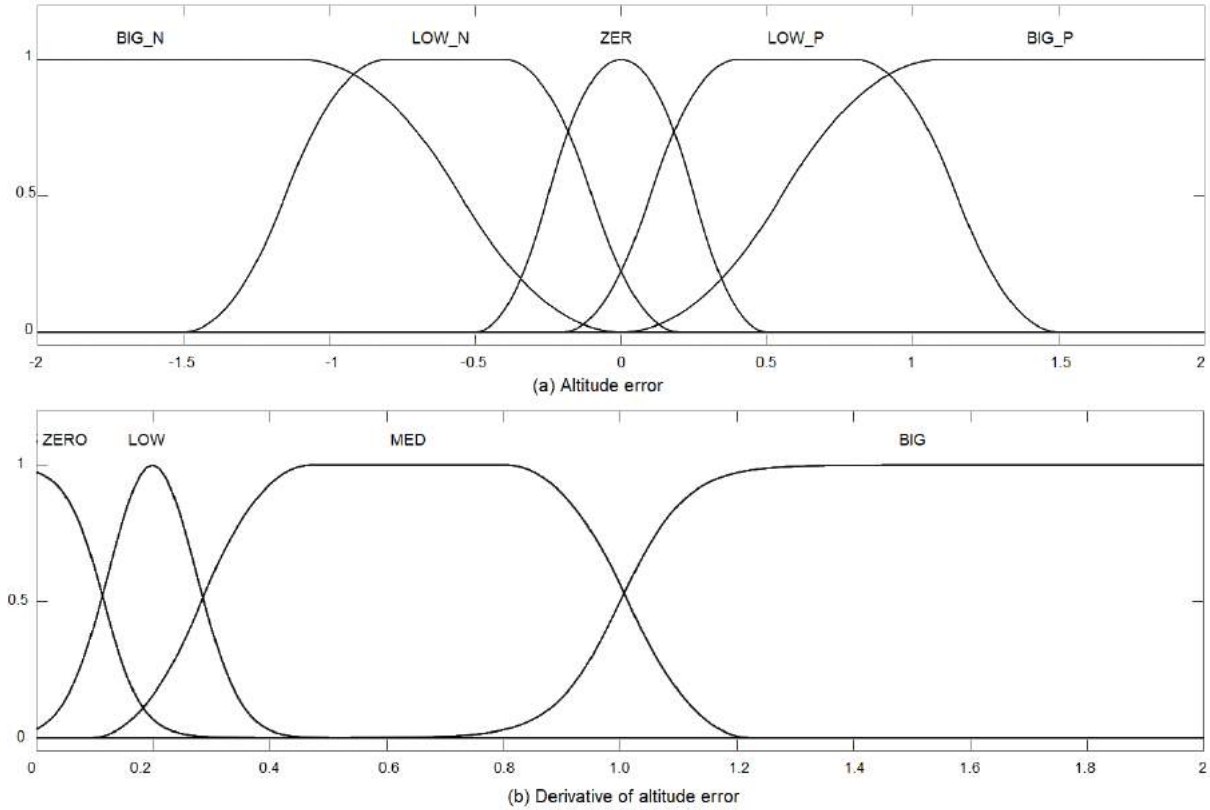


Figure IV.19: Input Fuzzification.
From: [49]

In Figure IV.25 a 3D profile is depicted with the results of the simulation tests, for comparison among other control approaches. It's possible to notice that the fuzzy scheduler controller has smaller error amplitude.

In a second simulation test series, the same test as before is applied, but considering the previous fuzzy controller alone and using both altitude and position controllers. For this test, the position control error became more relevant, as can be seen in Figure IV.26. As the position controller prioritized altitude, a minor overall error is expected. Four different controllers are analyzed, which are PID with noise combined fuzzy without noise, PID without noise, and combined fuzzy with noise.

In Figure IV.27 is depicted in details the height response at the end of a step, which represents quantitatively that the error is 2% lower. It is possible to see that the position controller is only affected when the altitude error is higher.

In this research, the performance test of the fuzzy gains-scheduling for the PID controllers is also tested in a real-world scenario, using the quadrotor of Figure IV.28. This UAV is embedded with a

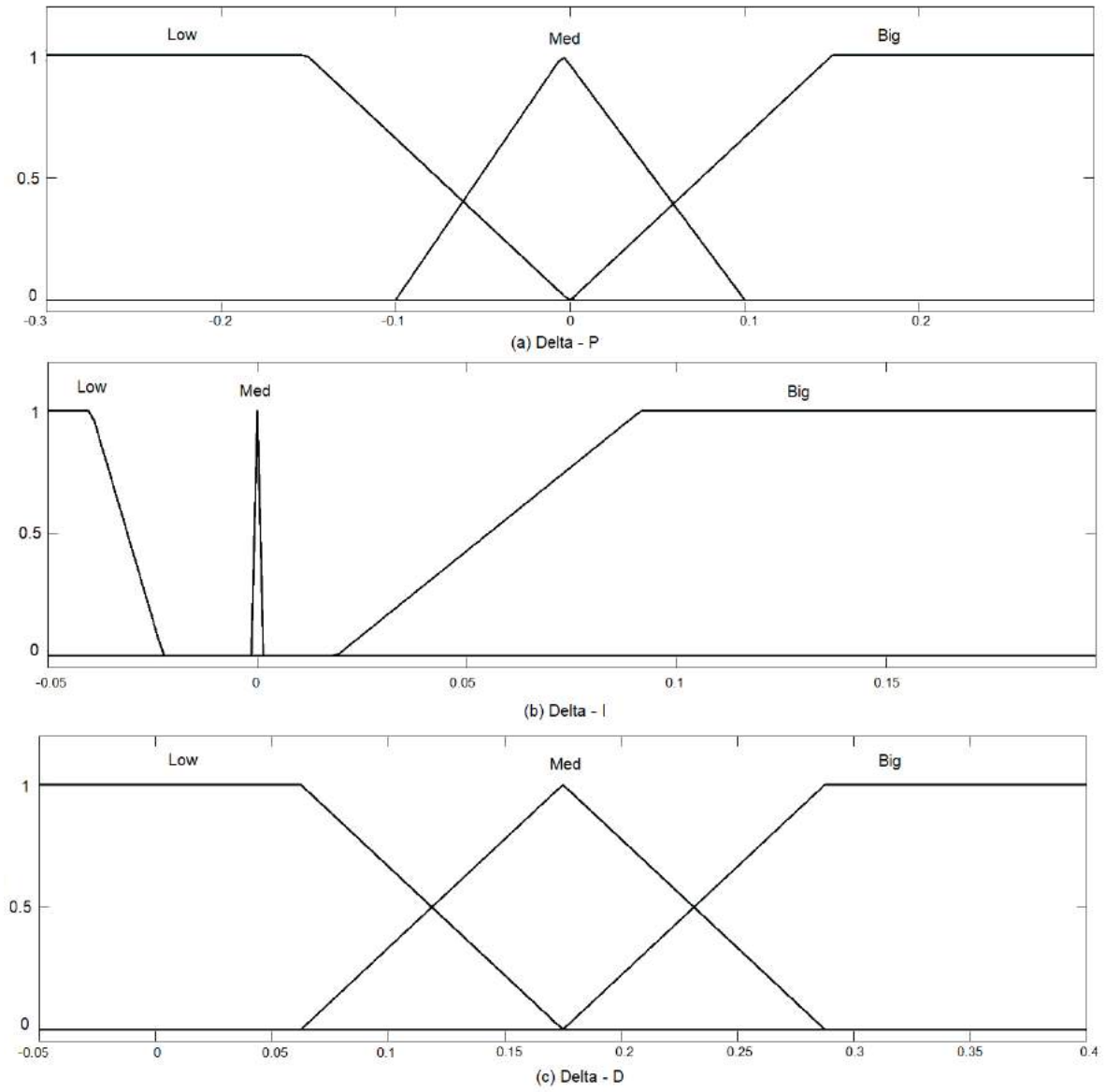


Figure IV.20: Output DeFuzzification.
From: [49]

Pixhawk 2.4.8 32-bit FCU with Ardupilot firmware, a NEO-8M GPS, and a companion computer. The FCU is compatible with the network interface for ROS and the MavLink communication protocol for telemetry and offboard controller. The MAVROS interface allows real-time information access. The experiment evaluates both the altitude and position controllers and the original gains are obtained from the standard PID gains.

In the first experiment, the UAV is submitted to a fast altitude change, and the position response is measured to simulate a step response. The process is repeated once the proportional gain is changed to modify the aircraft's responsiveness. After having the proportional gain defined and fixed, the derivative and integrative gains are adjusted. It is important to remember that these changes are part of a practical strategy to avoid any possible modeling issue in the PID adjustment. The fuzzy-PID and the PID gains

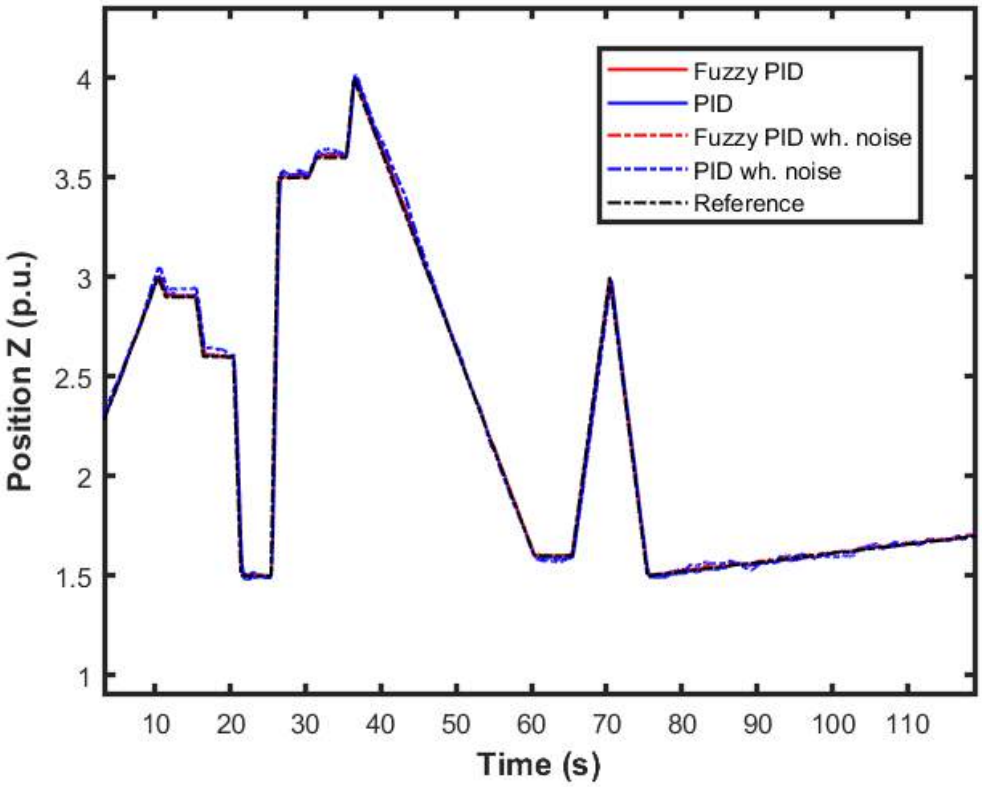


Figure IV.21: Comparison of the proposed method and PID controller for altitude control. From: [49]

are defined by the same strategy used in the simulations, using the values from the previous phase. The UAV received some inputs to simulate step response and evaluate the methods, as Figure IV.29 presents.

In this practical experiment, the behavior observed is that the fuzzy-scheduler scheme produced a higher noise than the original controller. This can be related to the increased gains used and the scheduling, but it could also be due to the sensors' amplified noise in the controller. On the other hand, the height tracking is more perceptible, being even closer to its reference. The experiment also permitted to use of the data to estimate the controller performance parameters, as seen in IV.5:

Table IV.5: Controller performance.

Original	
Param.	Value
Rise Time	0.1 s
Settling Time	1.1 s
Overshoot	13%

Fuzzy	
Param.	Value
Rise Time	0.1 s
Settling Time	0.2 s
Overshoot	1%

This work used a fuzzy scheme to adjust the position and the altitude controllers following any environmental changes. Each of the fuzzy controllers measures the variable error to determine the current

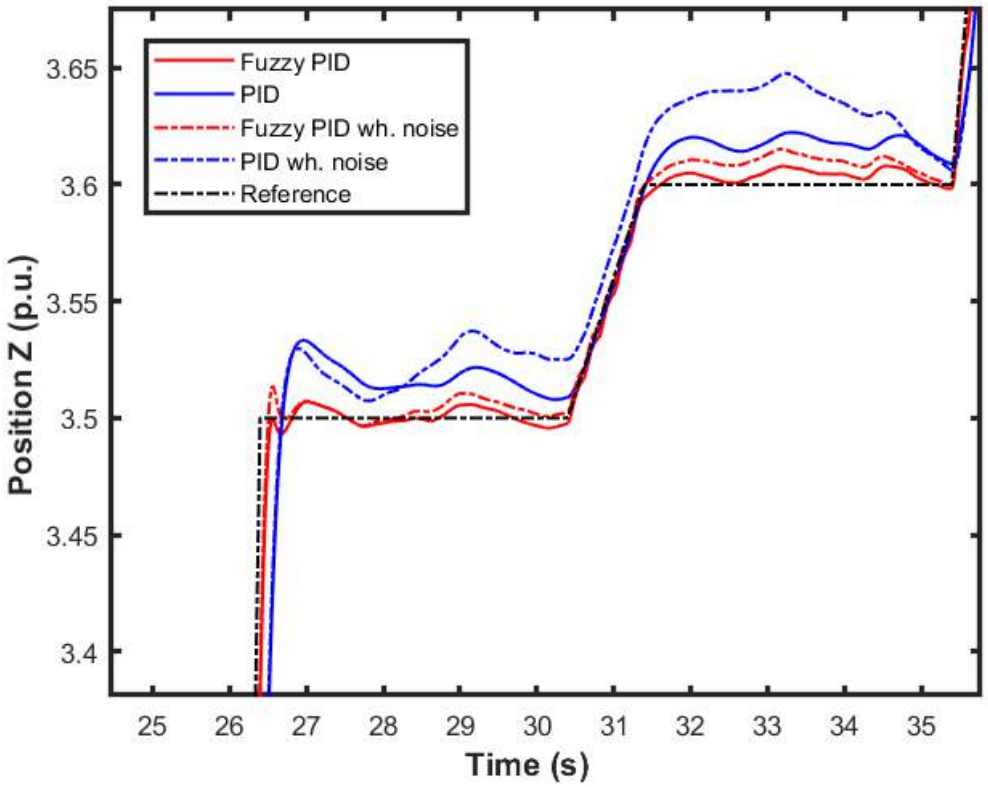


Figure IV.22: Detail of comparison of the proposed method and PID controller for altitude control. From: [49]

scheduling approach. The first scheduler is meant to control the altitude and is based on the output error and its time derivative. The second scheduler is the difference between most of the researched works. It uses position and altitude errors to define how the position controller should approach. The main PID gains were chosen through frequency response, although other methods such as Ziegler-Nichols, for instance, could be applied.

For the second experiment, it is made a control test on critical conditions. The UAV is sent to a given height, and then, to simulate a sudden increase in power requirements from the motors, more height is added. With this test, the fuzzy-scheduler can show the power of prioritizing altitude control. First, the UAV takes off and finds a stable position control while hovering at 1 meter, then a 5-meters-step with a load attached to a 4-meter cable is added. This test increases the UAV weight to 95% of its designed load. Figure IV.30 depicts the experimental results without the fuzzy-scheduler. It can be observed that when the load increases, both position and height control do not operate properly.

The same test is made in Figure IV.31 but with the fuzzy-scheduler active and running. This process was repeated five times to quantify the average response observed. The position and height are relatively stable, maintained by the UAV to its designed height with some overshoot. This overshoot

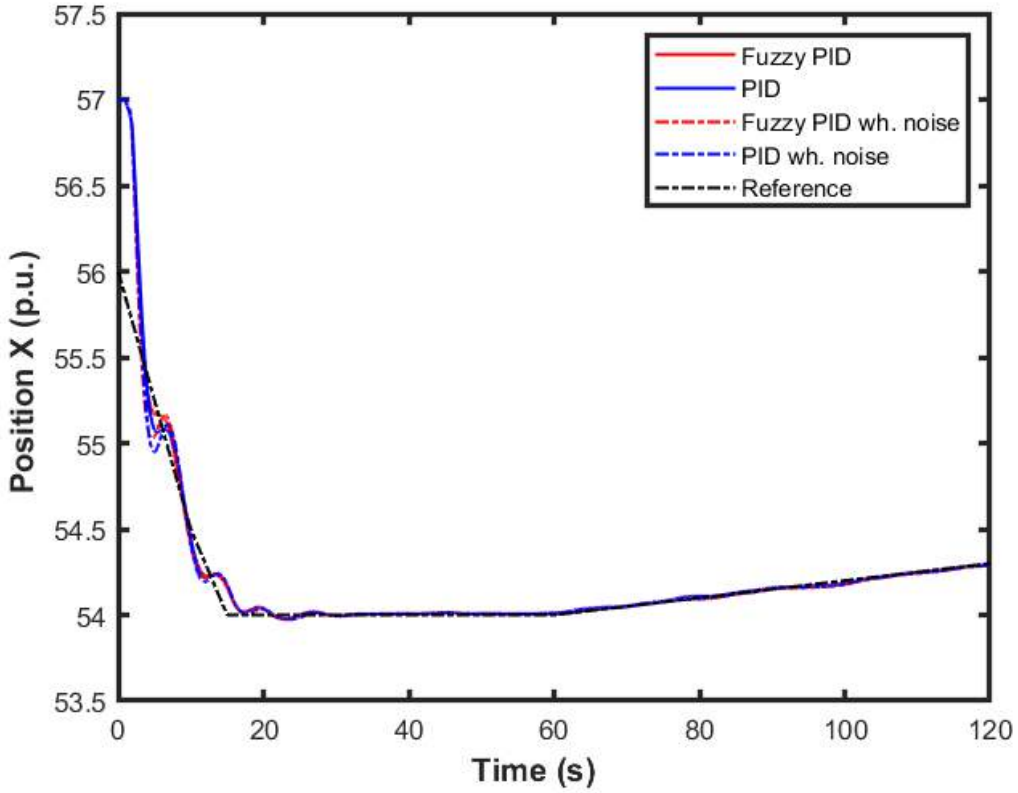


Figure IV.23: Comparison of the proposed method among others for position control in the X-axis. From: [49]

can be addressed to the high disturbance caused by the load when there is a control interaction of the position and height.

Some notes that might be made about the experiment are that the UAV couldn't maintain its altitude due to the motors' thrust condition, and the position control also loses its reference at some point in the experiment. This can be better seen in Figure IV.32, where the position response is isolated, as the view is seen from above in the same test in IV.31. The figure's position is concentrated in the given area related to the GPS errors and wind disturbance. The position reference is then lost, and the UAV deviates from its location.

The quantification of the performance in this condition is made by gathering the average from 500 samples of the height error. The fuzzy scheduler has an average error of 0.765 meters, while the average without it was 2.038 meters. As the ground truth was not captured externally, the experiment did not obtain the absolute improvement value.

The results for the trajectory tracking problem show that the PID-fuzzy-scheduler strategy, compared to the classical PID results, brings a good performance, even when considering the presence of

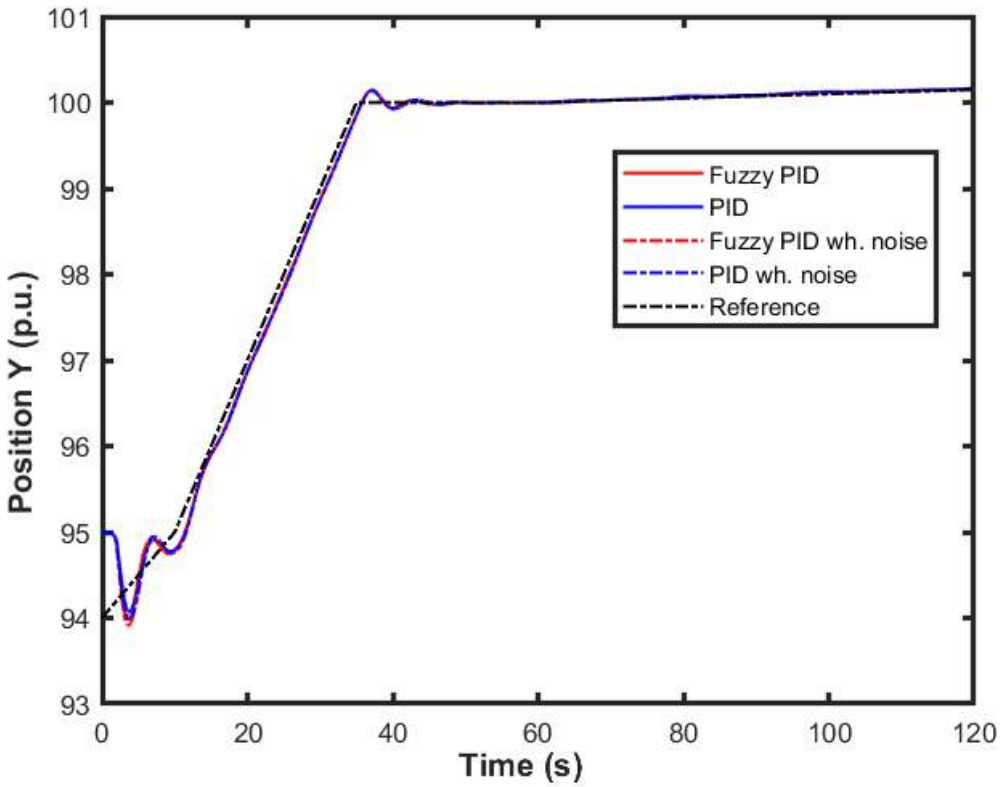


Figure IV.24: Comparison of the proposed method among others for position control in the Y-axis. From: [49]

measurement noise.

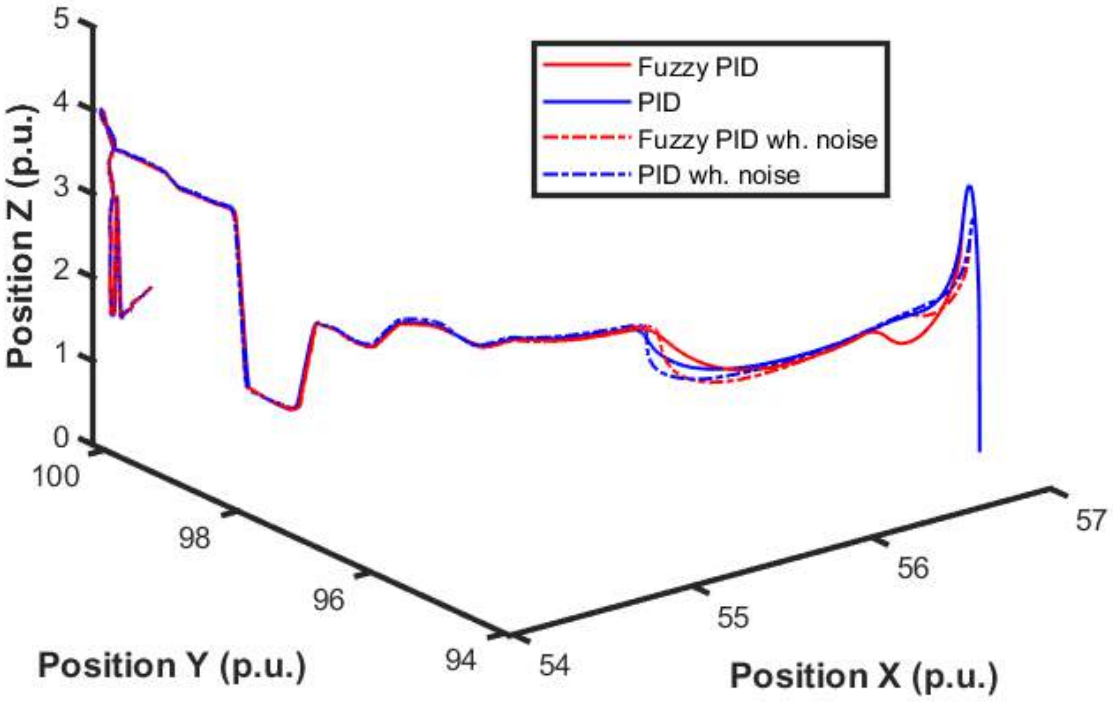


Figure IV.25: 3D visualization for the comparison of the proposed method among other approaches for position control.

From: [49]

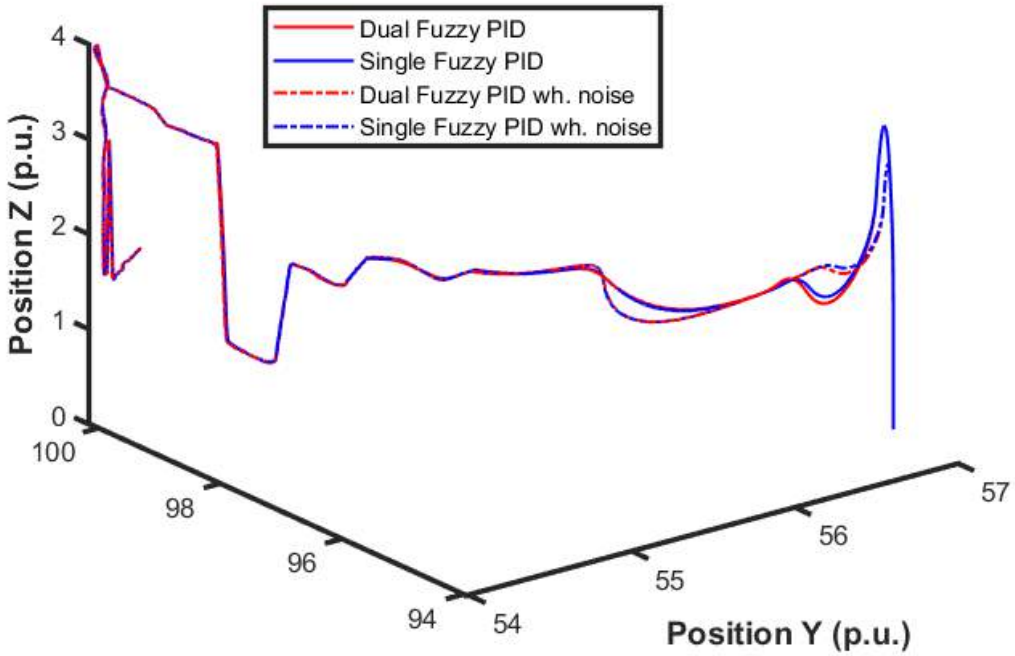


Figure IV.26: UAV 3D path during second simulation tests.
From: [49]

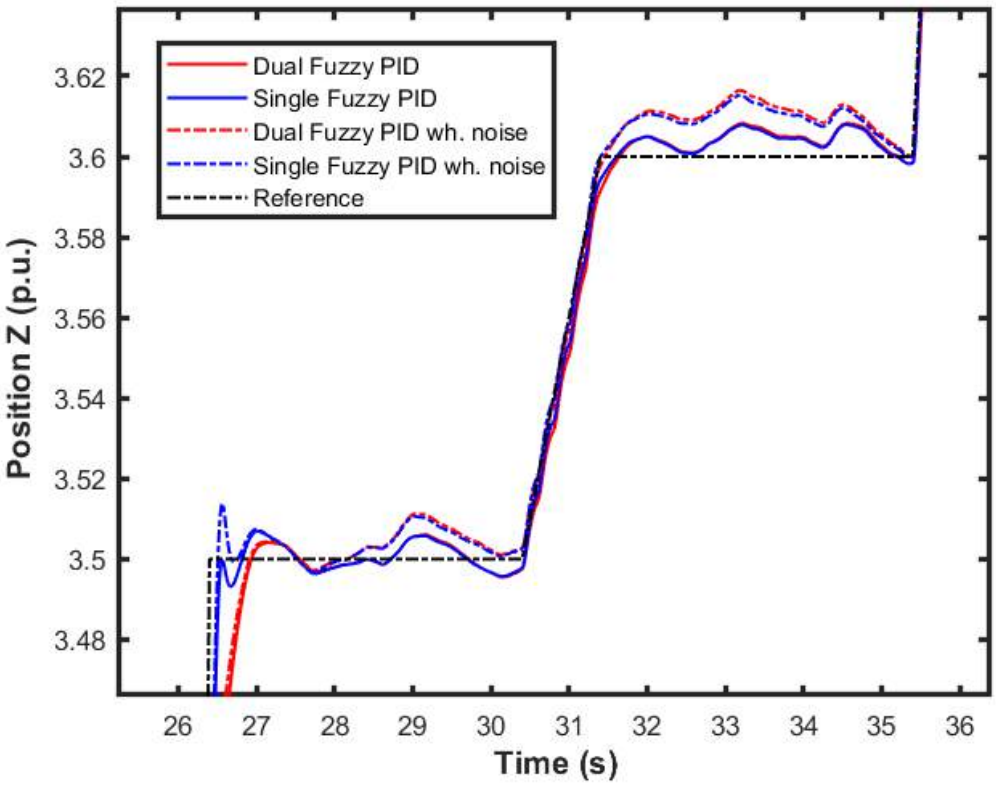


Figure IV.27: UAV height response details.
From: [49]



Figure IV.28: Quadrotor used in the experimental tests.
From: [49]

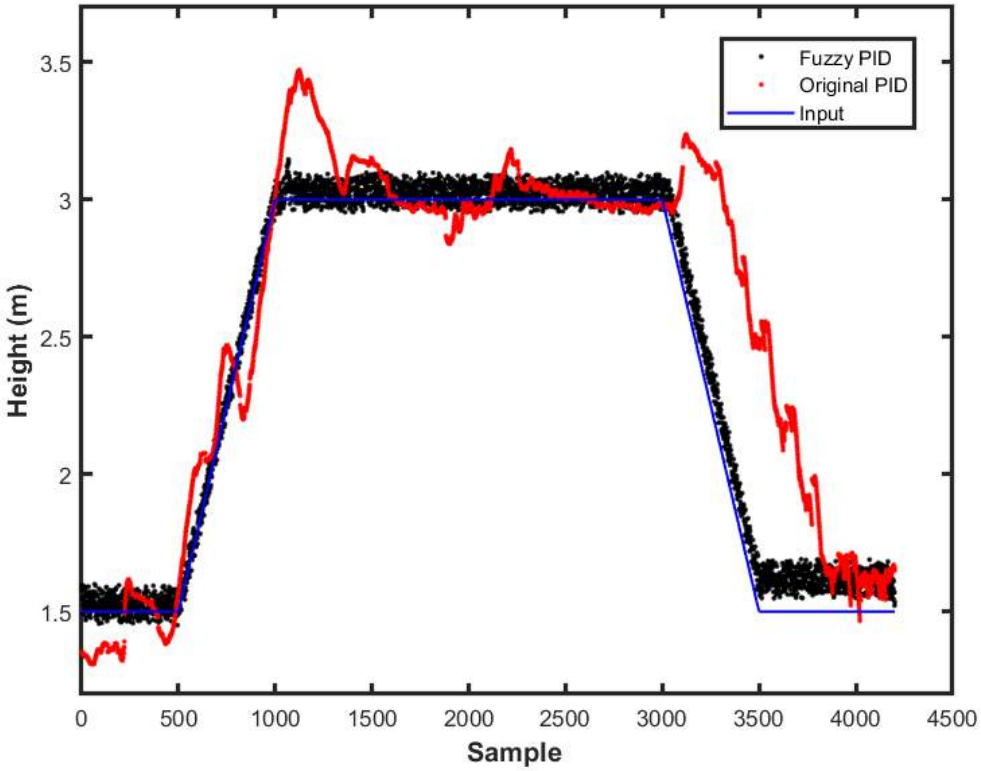


Figure IV.29: UAV height response for the practical experiment.
From: [49]

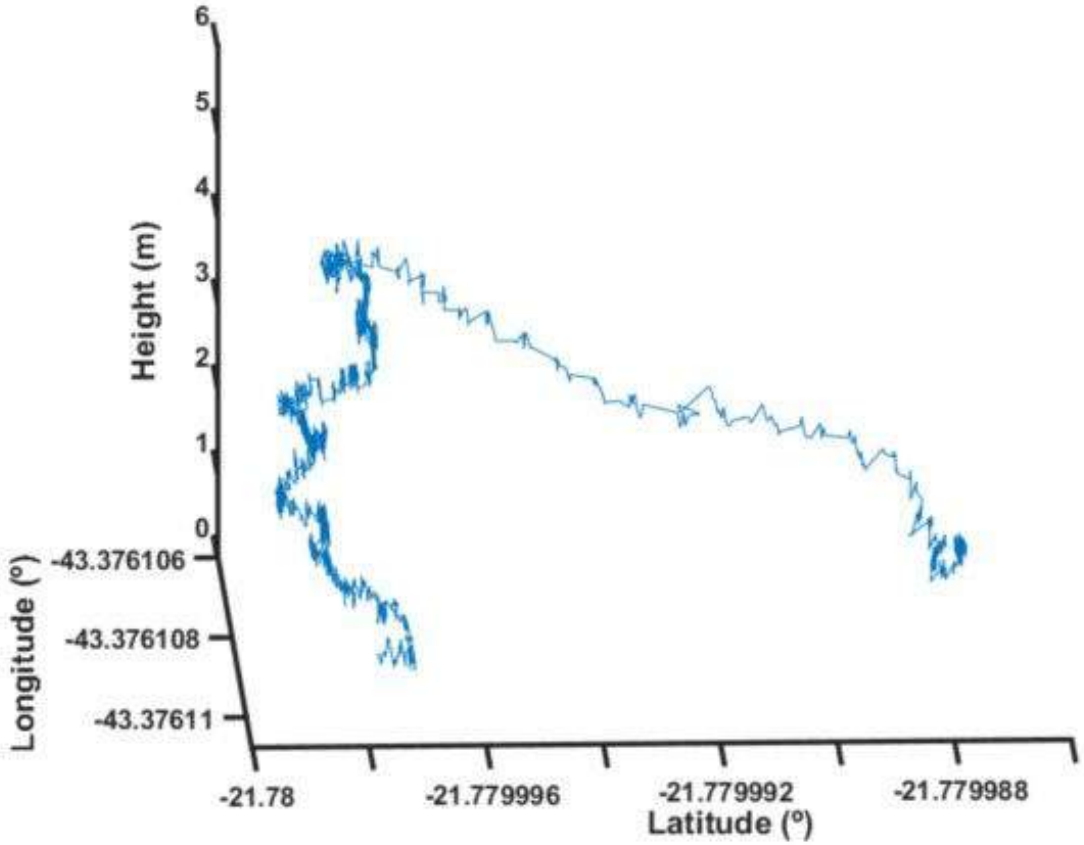


Figure IV.30: Height response without fuzzy scheduling.
From: [49]

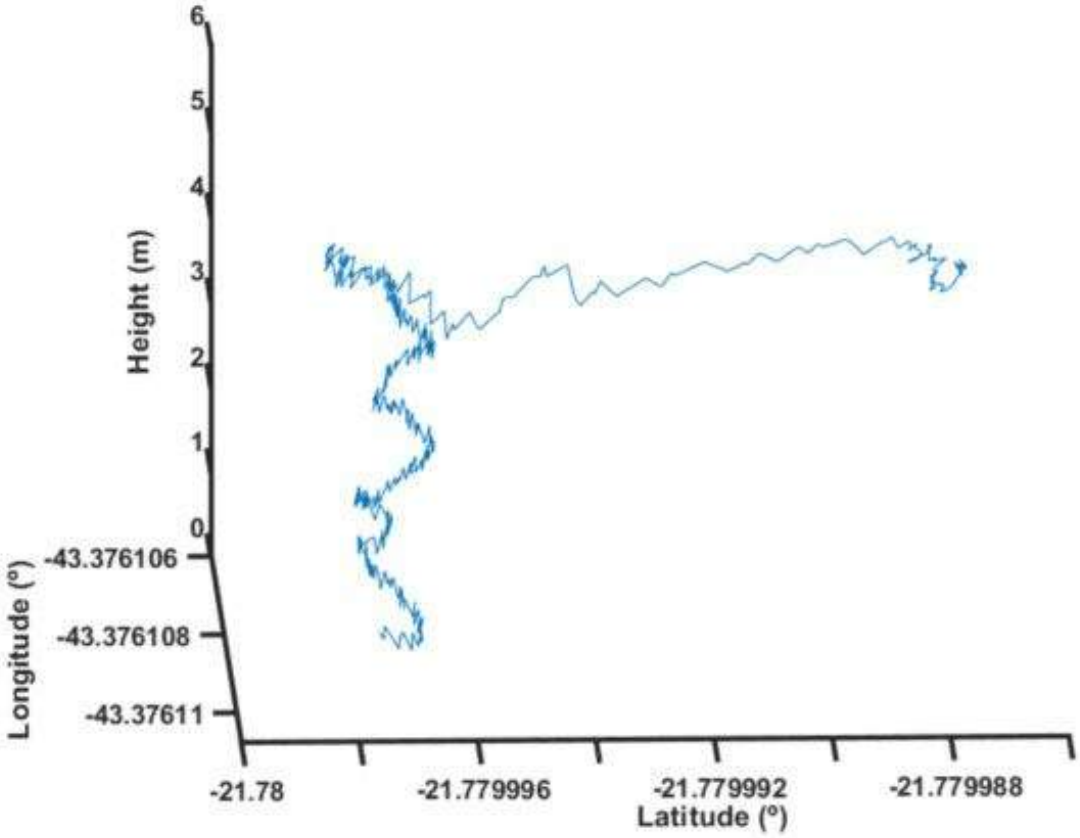


Figure IV.31: Height response with fuzzy scheduling.
From: [49]

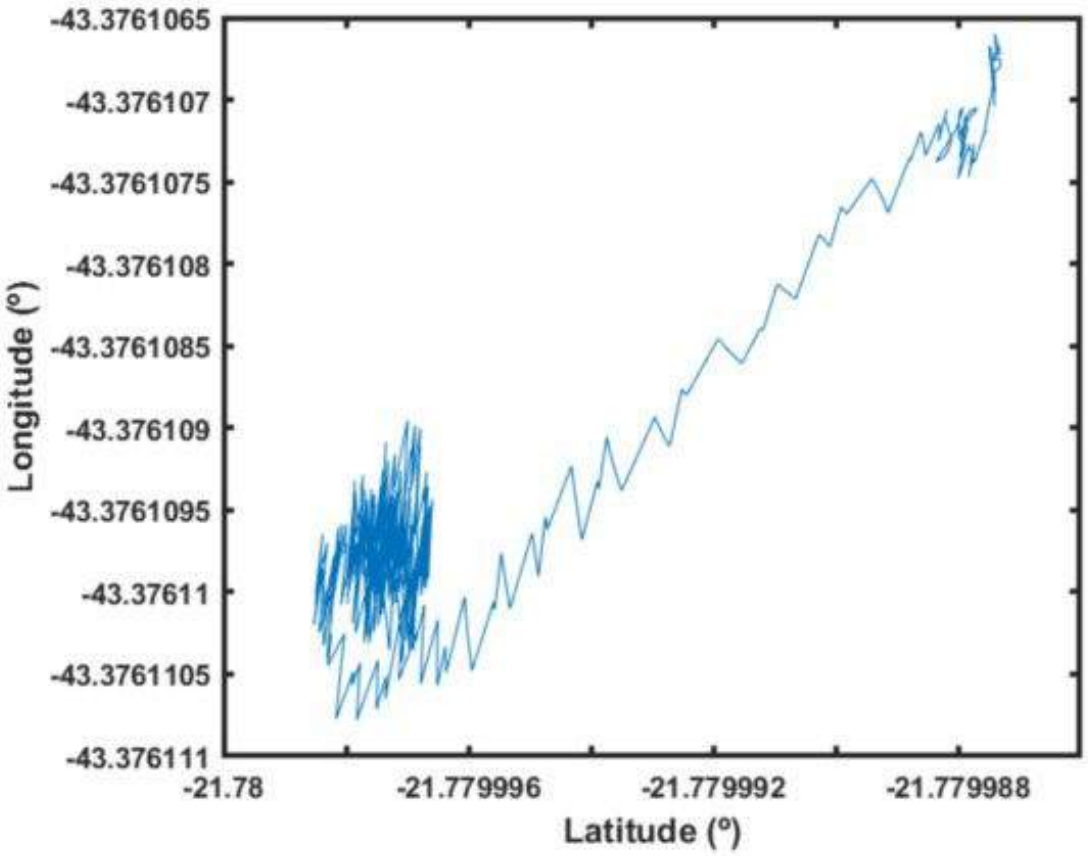


Figure IV.32: Position response of the test with the fuzzy-scheduler.
From: [49]

IV.4 Optical Flow simulation

After the development of [27], the implementation of the Optical Flow using Lucas-Kanade, combined with the EKF took place and generated the results that will be further described.

In order to evaluate the performance of the implemented technologies for this project, a world was developed using Gazebo software to simulate a solar farm. This simple scheme contains 14 solar panels set all oriented to the same direction, in theory, the direction in which the harnessing of sunlight would be optimal.

The world generation is used as the default parameter for physics calculations, the "type quick". The *Type* means the solver parameters used to solve the Ordinary Differential Equation (ODE)s within the model. The *Quick* type uses an iterative Projected Gauss-Seidel (PGS) method [25] with successive over-relaxation and depends on the number of iterations to reach an accurate solution, but gives the model a faster solution. In this case, the number of iterations was 50, and the quick model also sets other parameters such as gravity and magnetic field, for instance. Aesthetic parameters such as light and grass are also defined, and the solar panels and the solar panel's patterns are also defined.

In this scenario, the UAV must trace a trajectory that allows it to analyze all solar panels pragmatically. For this, the way-point method was used to trace the initial trajectory. To correctly observe the panels, the UAV must follow the zigzag path, with the field of view angled downwards. In this simulation, the UAV travels at 5 m/s, at 5 meters, as depicted below.

The project also has the Cascaded-PID-Fuzzy controller, as well as a script with defined way-points for scanning the desired area, in further works being auto-corrected using the optical flow with the aid of Lucas-Kanade, which can be monitored through the test environment in Gazebo, or using QGround Control. Note that other sensors of the UAV can be used to aid the optical flow by using EKF.

A work that is possible to cite that used EKF to fuse data from the optical flow and other sensors of the UAV is [65]. In their work, the authors proposed a flight hybrid methodology based on deep learning and sensor fusion to support the autonomous flight of a UAV. Their technique focused on take-off from an oil rig and precisely reaching a specific point in a vessel without using expensive positioning devices, such as GPS. As mentioned before, these devices are susceptible to Global Navigation Satellite System (GNSS) dropouts.

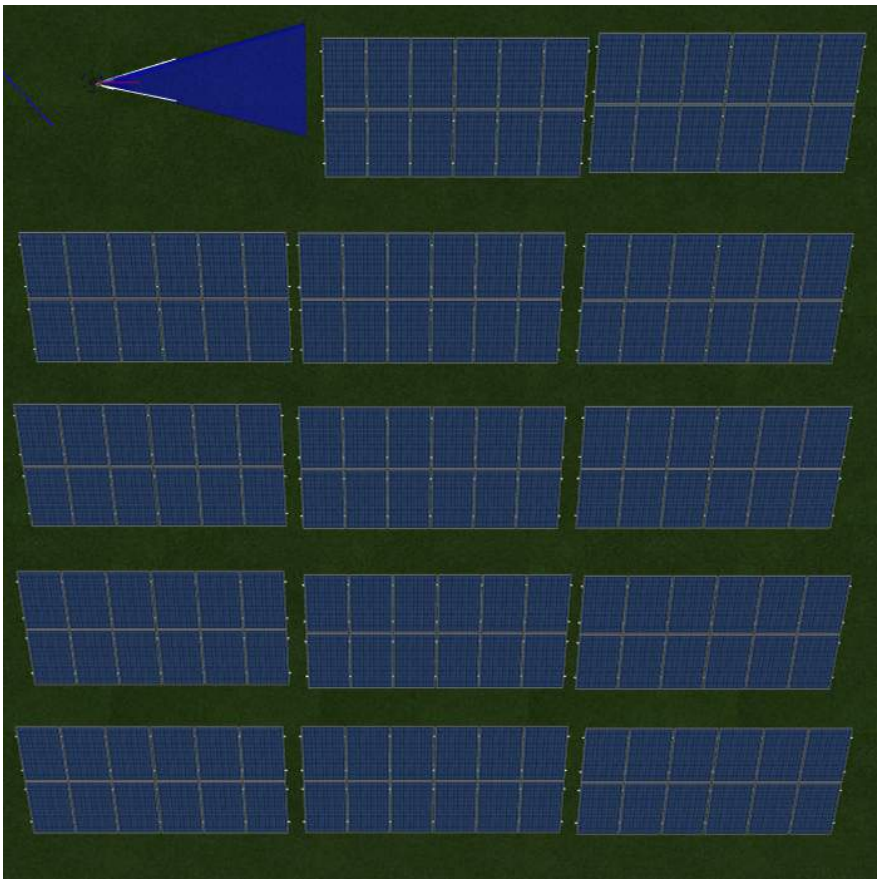


Figure IV.33: Simulation environment in Gazebo.
From: Author

First, to design and test the proposed Cascaded-PID-Fuzzy controller, it was used a numerical software simulator (i.e., Matlab). The Simulink configuration is shown in Figure IV.35. Note that Simulink support ROS communication, which allows different components of a robot system to exchange information in the form of messages. In this case, a message is sent by publishing it into a particular topic (i.e., position). Other components receive the message by subscribing to that topic.

Further it will present the results that were reached so far testing this kind of visual-based navigation. First, it was chosen a video containing aerial imagery to test the UAV basic motion derived from the optical flow algorithm once the optical flows can be divided into rotation-induced flows, translation-induced flows, and general flows generated by both rotations and translations. In this sense, Figure IV.36 presents the application of the Lucas-Kanade technique over sample images. Note in the figure that the optical flow present's the change in the direction of the image similar to a key point.

Using this information from two subsequent images, one can estimate the movement relative between them. This step is shown on Figure IV.37. Note in the figure that a vectorial field representing the changes along the image is generated. This vector has to be interpreted to determine the camera

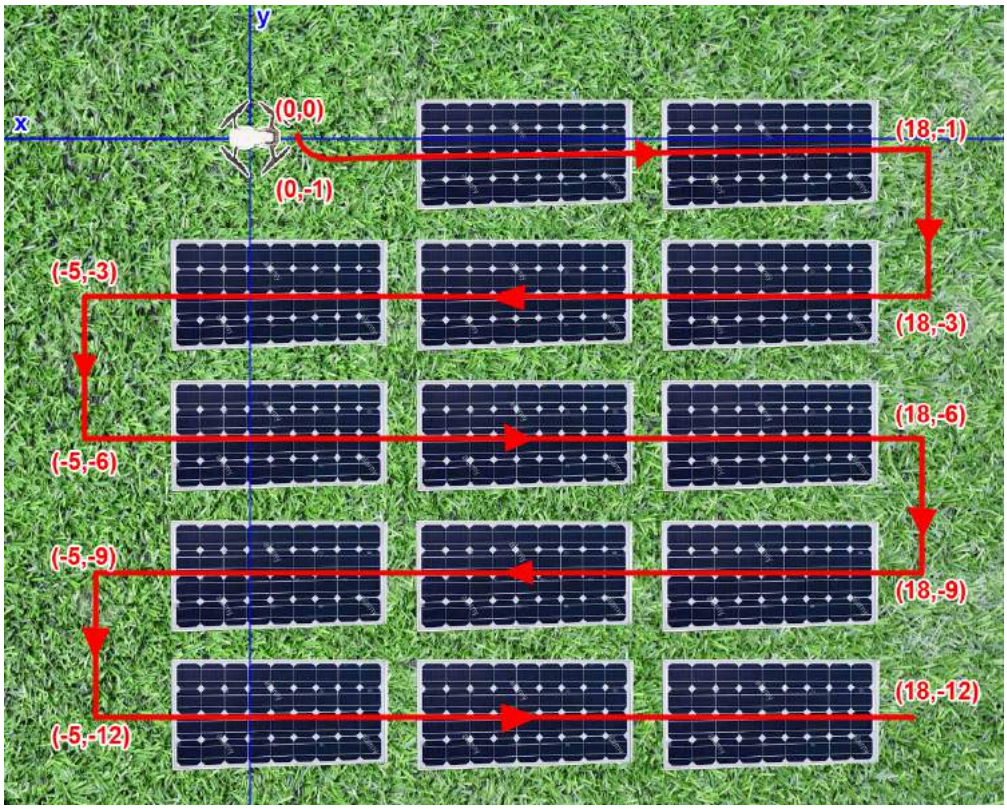


Figure IV.34: Trajectory scheme.
From: Author

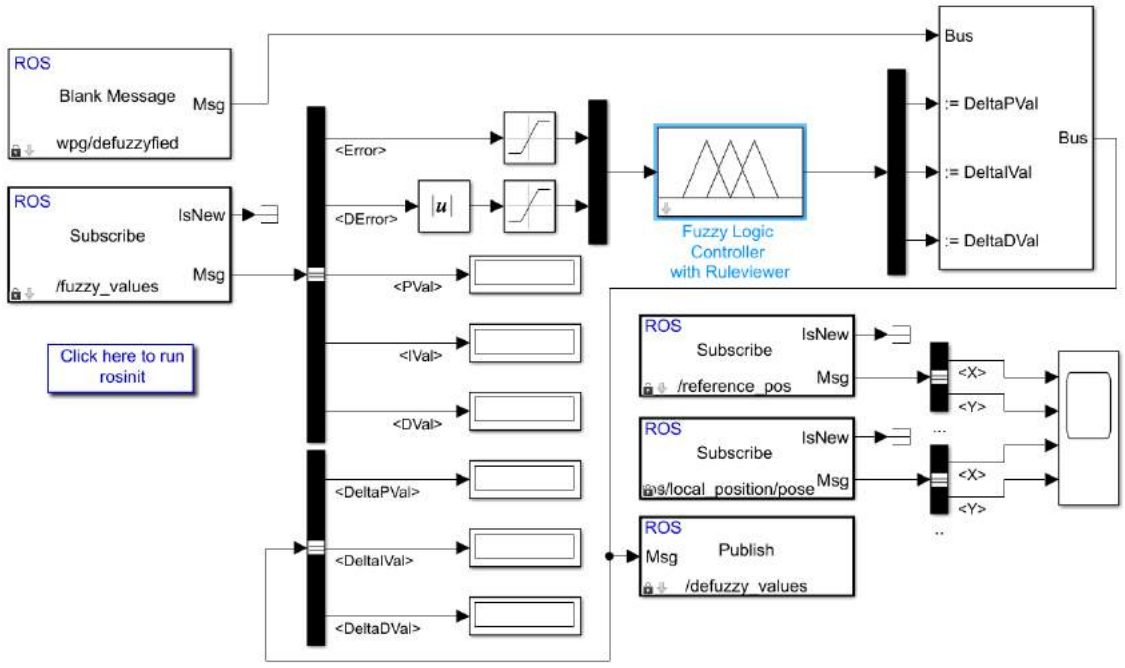


Figure IV.35: Simulink scheme.
From: Author

movement from its origin.

By using this same methodology, two sample cases were built. In Figure IV.38, one can see the

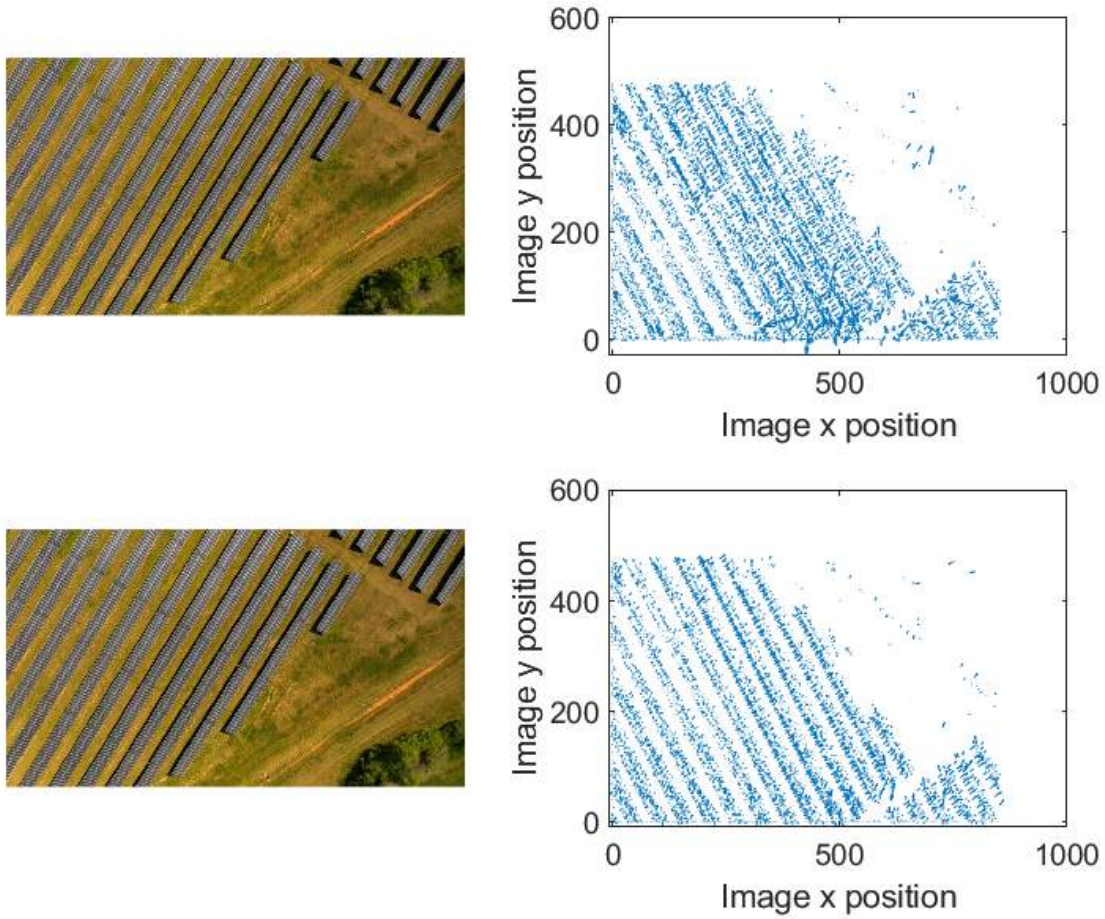


Figure IV.36: Lukas-kanade application over a sample images.
From: Author

optical field generated for a linear movement.

For the rotation case, it is also possible to see an example in Figure IV.39. Note in this figure that the average vector movement will be closer to zero. However, the rotation information can be observed without any issue.

These results indicate that tracking drone position using images is feasible. More experiments are required to produce an adequate analysis of the errors in each condition, i.e., in different camera resolutions. The script used to develop the Optical Flow tests is in Appendix V, and it generates the Optical Flow results based on a mp4 video, which contains the flight of a camera over a solar farm. Two different videos were used to test the script, as they should have the required motion to produce the needed analysis.

Tracking the camera position depends on many factors, such as the camera orientation related to the ground, the UAV orientation, and the distance from the target. Further experimentation is also

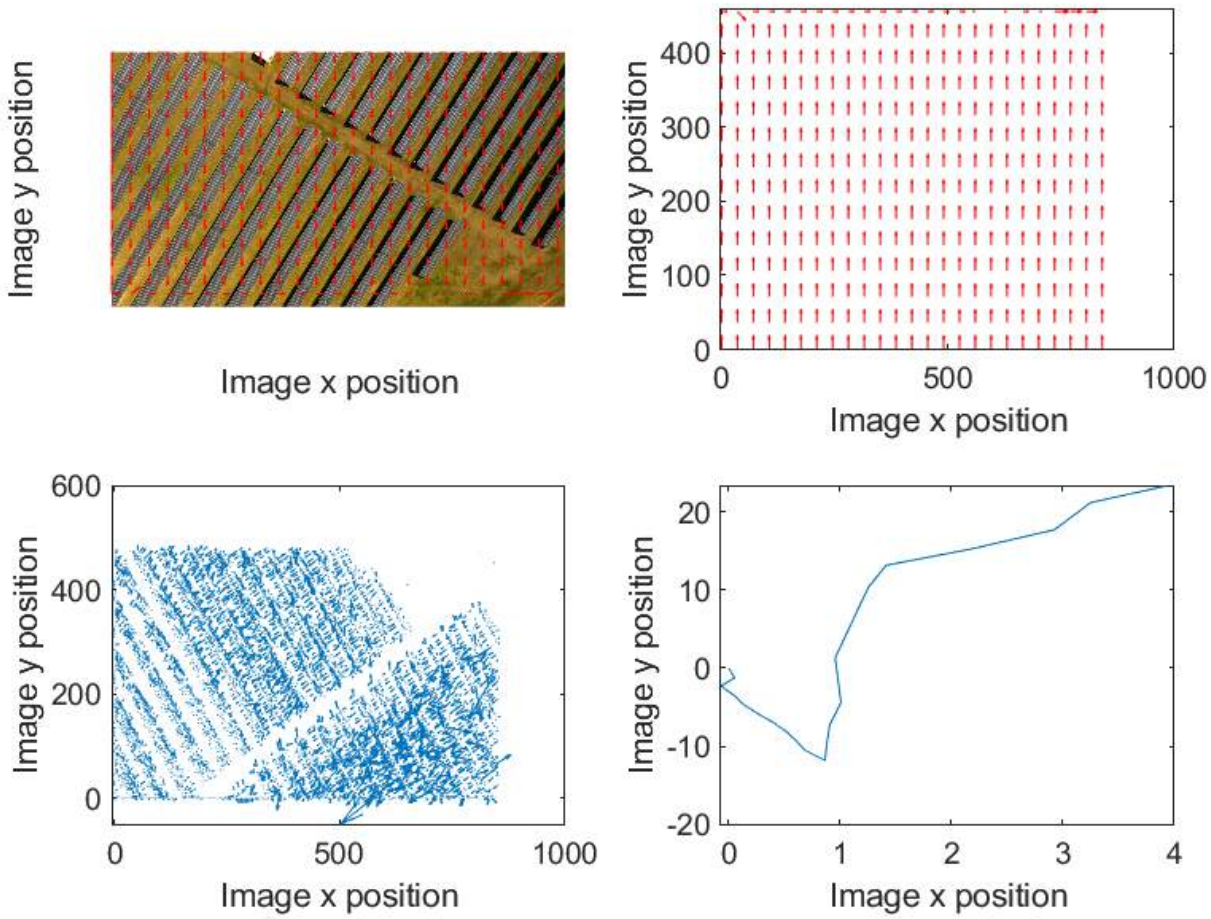


Figure IV.37: Movement estimation from optical flow.
From: Author

required to parameterize filters and transformations capable of processing this information.

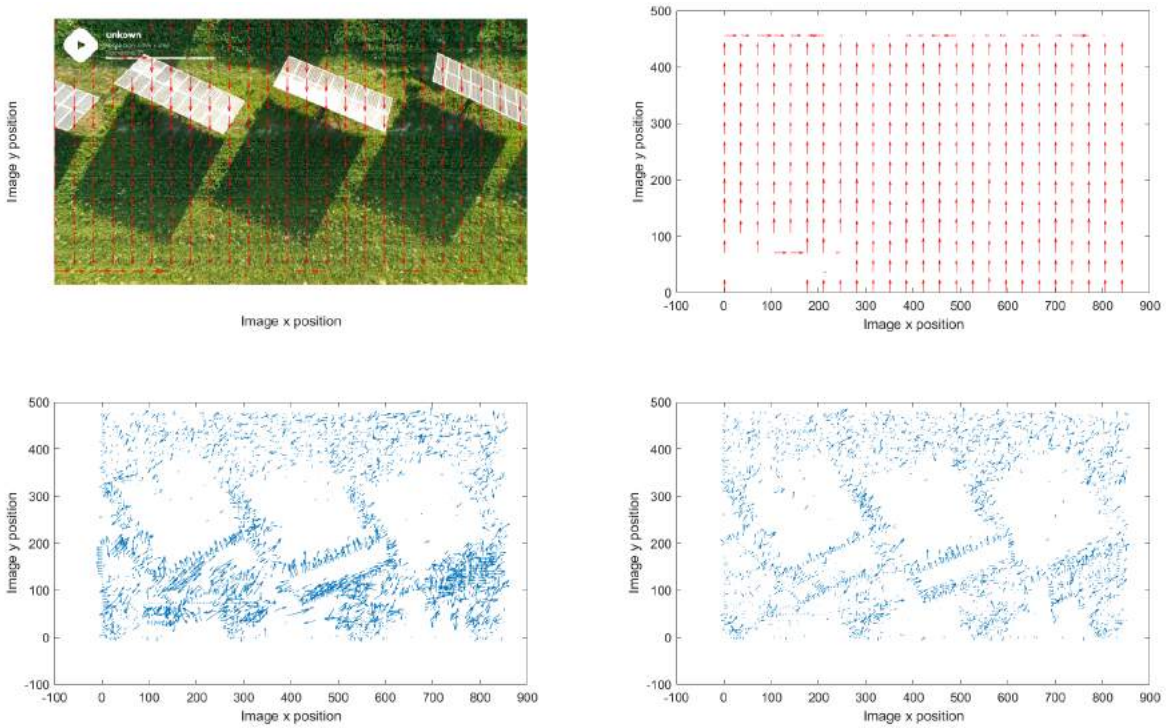


Figure IV.38: Example result for linear movement.
From: Author

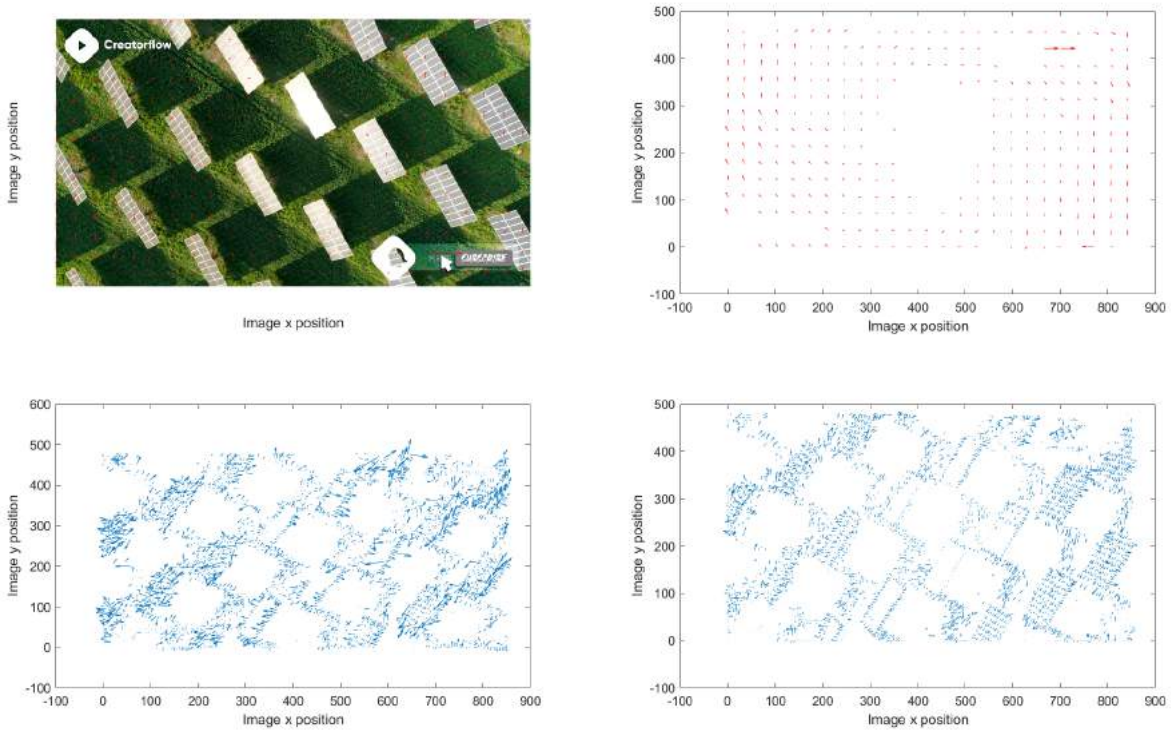


Figure IV.39: Example result for rotation.
From: Author

Chapter V Conclusions and Future Work

Nowadays, UAVs have been widely applied in diverse applications because of their flexibility of maneuvers, reducing risks to human life, ease to control, and cost-effectiveness. A drawback is the necessity of a platform with a high degree of stability to prevent the UAV from falling and to maintain the desired orientation and path during the flight. For this, a simple structure and less dependence on the model is the PID controller. However, its design and adjustment can be difficult, and in this sense, this research proposes a set of PID-Fuzzy algorithms. Although the fuzzy gains are not based on the previous state values, such modification would be easily achieved once the ranges' reach and gain limits were redefined. The PID controller is a simple control scheme widely applied in the literature. However, as a linear control strategy, it is observed that its performance suffers degradation when it is applied in linearized systems at different operating points.

In this specific scenario, the fuzzy rules and membership functions play a crucial role in defining whether PID gains should prioritize stability. For instance, enhancing derivative gain in a constant position (zero error) scenario or prioritizing high gain PI for faster response when position error and its derivative are large. The results showed that the proposed approach was successfully applied to the trajectory tracking of a quadrotor UAV. It was also possible to evaluate that fuzzy-gain enhanced the main test's controlling performance, reducing overshoots and undershoots.

The results showed that the proposed approach in [27] was successfully applied to the trajectory tracking of a quadrotor UAV. It was also possible to evaluate that fuzzy-gain enhanced the main test's performances and also reduced overshoots and undershoots in the response. Regarding the results of [5], some oscillations were noted on the commanded PID gains, which should indicate that this is due to different rates among the different systems (namely ROS, FMU, MATLAB/Simulink). This affected the commanded velocity but did not affect the actually measured velocity.

In the researched literature, most of the Fuzzy Logic tuning solutions use rules defining the PID gains (K_P, K_D, and K_I) based on the output error and its time derivative. In work [5], however, the absolute value of the output error is used to generate the Fuzzy rules. With this, the deviation between the desired and current UAV position always assumes positive or null values, which facilitates

the definition of "low", "medium," and "high" levels for the fuzzy algorithm. It is common to find the same definitions of such levels for both the positive and negative excursions of the error time derivative, which is the same strategy adopted in fuzzy rules. Based on the observations made on the results obtained, it emphasized that the definition of different rules for the negative and positive excursions of the error derivative seems to influence the performance of the closed loop and that, therefore, it should be an issue to be investigated in future work.

The sudden change in operating points is usually due to rapid changes in motion. In such a scenario, the gain schedule technique is an attractive solution to adjust this type of controller during changes in plant operating regions. In this sense, this work developed a fuzzy gain-scheduling PID controller for a quadrotor vehicle in order to control its position and altitude. The tests were performed both in a simulation environment and in a real quadrotor system. The results for the trajectory tracking problem showed that the proposed control and tuning strategy returned a good performance compared to the classical PID results, even considering the measurement noise.

In terms of evaluation, this research opens the possibility of several future works. Many development ideas were already proposed by the authors in the articles, such as including the fuzzy iterative learning algorithm applications to select the controller parameters, testing the control system when the UAV is carrying different payloads, testing energy usage, inserting the proposed control algorithm along with an optical flow algorithm to improve the position attitude estimation and allow the use in GPS-denied environments and performing autonomous flight.

The optical flow tests and consulted literature suggest that implementing the proposed controller with optical flow imaging is not also feasible but should reduce the position errors significantly, especially at greater paths. Another aspect presented by the implementation of such a controller should be the automatizing of tasks and enhancing the reliability of the captured images and flight control. The altitude controller combined with the fuzzy-scheduler also can be implemented along with these optical flow tests in further research, and the energy spent with and without the scheduler can be measured.

The MATLAB/Simulink was used for the proof of concept version of this solution, and the Fuzzy Logic algorithm could be embedded in the ROS C++ or Python scripts for the development in future works. By doing so, achieving a faster and more synchronized system is expected, as the information processing would have the delays avoided by embedding in the FCU of the UAV. As the fuzzy logic has multiple parameters and instructions that directly impact the processing speed, there should be several possible improvements in terms of implementation. For instance, optimal algorithms could be used to

improve the processing speed.For further works, it would also be interesting to develop an autonomous inspection task, not only by combining the UAV flight stabilization control and path definition but also by training a neural network to identify any maintenance points or attention.

By the presented aspects, one can relate the achieved results to a better stability in UAV flight control and, thus, to a more stable and better image generation for inspections in general. For the purpose of inspection of solar panels, this would reduce the workload from the inspector and also provide a better view of the panels, enhancing the capacity to inspect. For this purpose, it creates better opportunities to perform such maintenance step more often with better results, followed by energy and money saving.

References

[1] Pia Addabbo et al. "UAV system for photovoltaic plant inspection". In: *IEEE Aerospace and Electronic Systems Magazine* 33.8 (2018), pp. 58–67.

[2] Mohammadreza Aghaei et al. "Unmanned aerial vehicles in photovoltaic systems monitoring applications". In: *European photovoltaic solar energy 29th conference and exhibition*. 2014, pp. 2734–2739.

[3] S Faiz Ahmed et al. "Attitude stabilization of Quad-rotor (UAV) system using Fuzzy PID controller (an experimental test)". In: *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*. IEEE. 2015, pp. 99–104.

[4] Kelson Rômulo Teixeira Aires and Adelardo Adelino Dantas de Medeiros. "A Plane Segmentation System Based on Affine Homography and Optical Flow". In: *2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE. 2010, pp. 346–352.

[5] Fabio AA Andrade et al. "Unmanned Aerial Vehicles Motion Control with Fuzzy Tuning of Cascaded-PID Gains". In: *Machines* 10.1 (2021), p. 12.

[6] Simon Baker and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework". In: *International journal of computer vision* 56.3 (2004), pp. 221–255.

[7] Serkan Biçici and Mustafa Zeybek. "An approach for the automated extraction of road surface distress from a UAV-derived point cloud". In: *Automation in Construction* 122 (2021), p. 103475.

[8] Iago Z Biundini et al. "A Framework for Coverage Path Planning Optimization Based on Point Cloud for Structural Inspection". In: *Sensors* 21.2 (2021), p. 570.

[9] Kleber Macedo Cabral et al. "Design of model predictive control via learning automata for a single UAV load transportation". In: *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE. 2017, pp. 1–7.

[10] Michael Calonder et al. "Brief: Binary robust independent elementary features". In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[11] Marko Car et al. "Autonomous Wind-Turbine Blade Inspection Using LiDAR-Equipped Unmanned Aerial Vehicle". In: *IEEE Access* 8 (2020), pp. 131380–131387.

[12] Elisa Casella et al. "Drones as tools for monitoring beach topography changes in the Ligurian Sea (NW Mediterranean)". In: *Geo-Marine Letters* 36.2 (2016), pp. 151–163.

[13] Haiyang Chao, Yu Gu, and Marcello Napolitano. "A survey of optical flow techniques for UAV navigation applications". In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2013, pp. 710–716.

[14] Haiyang Chao et al. "A comparative study of optical flow and traditional sensors in UAV navigation". In: *2013 American Control Conference*. IEEE. 2013, pp. 3858–3863.

[15] Changhyun Choi and Henrik I Christensen. "Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 4048–4055.

[16] Carlos AM Correia et al. "Comprehensive Direct Georeferencing of Aerial Images for Unmanned Aerial Systems Applications". In: *Sensors* 22.2 (2022), p. 604.

[17] Mohammed Dhriyyef et al. "Management strategy of power exchange in a building between grid, photovoltaic and batteries". In: *International Conference on Electronic Engineering and Renewable Energy*. Springer. 2020, pp. 831–841.

[18] Weidong Ding et al. "Adding optical flow into the GPS/INS integration for UAV navigation". In: *Proc. of International Global Navigation Satellite Systems Society Symposium*. Citeseer. 2009, pp. 1–13.

[19] Lefteris Doitsidis et al. "A framework for fuzzy logic based UAV navigation and control". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 4. IEEE. 2004, pp. 4041–4046.

[20] Jian Dong and Bin He. "Novel fuzzy PID-type iterative learning control for quadrotor UAV". In: *Sensors* 19.1 (2018), p. 24.

[21] Jian Dong and Bin He. "Novel fuzzy PID-type iterative learning control for quadrotor UAV". In: *Sensors* 19.1 (2019), p. 24.

[22] Khaled Eltag, Muhammad Shamrooz Aslamx, and Rizwan Ullah. "Dynamic stability enhancement using fuzzy PID control technology for power system". In: *International Journal of Control, Automation and Systems* 17.1 (2019), pp. 234–242.

[23] Milena Faria Pinto et al. "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle". In: *XXXX*. June 2017, pp. 1324–1329. DOI: 10.1109/ISIE.2017.8001437.

[24] Tiago M Fernández-Caramés et al. "Towards an autonomous industry 4.0 warehouse: A UAV and blockchain-based system for inventory and traceability applications in big data-driven supply chain management". In: *Sensors* 19.10 (2019), p. 2394.

[25] GazeboSim. *Physics Parameters*. http://gazebosim.org/tutorials?tut=physics_params&cat=physics#Solverparameters. 2014.

[26] S Griffith et al. "Maximizing miniature aerial vehiclesObstacle and terrain avoidance for MAVs". In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 34–43.

[27] Ihannah Guedes et al. "Hybrid PID-Fuzzy controller for autonomous UAV stabilization". In: *14th IEEE International Conference on Industry Applications INDUSCON 2021*. IEEE. 2021, pp. 1296–1302.

[28] Kelvin Hamilton and Jonathan Evans. "Subsea pilotless inspection using an autonomous underwater vehicle (SPINAV): concepts and results". In: *Europe Oceans 2005*. Vol. 2. IEEE. 2005, pp. 775–781.

[29] Bruno Herissé et al. "Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow". In: *IEEE Transactions on robotics* 28.1 (2011), pp. 77–89.

[30] Berthold KP Horn and Brian G Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[31] Xusheng Hu and Juan Liu. "Research on UAV Balance Control Based on Expert-fuzzy Adaptive PID". In: *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. IEEE. 2020, pp. 787–789.

[32] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. "Liteflownet: A lightweight convolutional neural network for optical flow estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8981–8989.

[33] Sergey Jatsun et al. "Control fligth of a UAV type tricopter with fuzzy logic controller". In: *2017 Dynamics of Systems, Mechanisms and Machines (Dynamics)*. IEEE. 2017, pp. 1–5.

[34] M Kamran Joyo et al. "Altitude and horizontal motion control of quadrotor UAV in the presence of air turbulence". In: *2013 IEEE Conference on Systems, Process & Control (ICSPC)*. IEEE. 2013, pp. 16–20.

[35] Niklas Karlsson et al. "The vSLAM algorithm for robust localization and mapping". In: *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE. 2005, pp. 24–29.

[36] Tyler Kersnovski, Felipe Gonzalez, and Kye Morton. "A UAV system for autonomous target detection and gas sensing". In: *2017 IEEE aerospace conference*. IEEE. 2017, pp. 1–12.

[37] Weiren Kong et al. "UAV Autonomous Aerial Combat Maneuver Strategy Generation with Observation Error Based on State-Adversarial Deep Deterministic Policy Gradient and Inverse Reinforcement Learning". In: *Electronics* 9.7 (2020), p. 1121.

[38] David Lara Alabazares et al. "Quadrotor UAV attitude stabilization using fuzzy robust control". In: *Transactions of the Institute of Measurement and Control* 43.12 (2021), pp. 2599–2614.

[39] Kurt Larson. "Fuzzy Logic Tuning of a Proportional-Integral-Derivative Controller". PhD thesis. California State Polytechnic University, Pomona, 2016.

[40] H. Le Nhu Ngoc Thanh and S. K. Hong. "Quadcopter Robust Adaptive Second Order Sliding Mode Control Based on PID Sliding Surface". In: *IEEE Access* 6 (2018), pp. 66850–66860.

[41] Xiaoxia Li et al. "Visible defects detection based on UAV-based inspection in large-scale photovoltaic systems". In: *IET Renewable Power Generation* 11.10 (2017), pp. 1234–1244.

[42] D. Liu et al. "Robust Fault-Tolerant Formation Control for Tail-Sitters in Aggressive Flight Mode Transitions". In: *IEEE Transactions on Industrial Informatics* 16.1 (2020), pp. 299–308.

[43] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[44] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: Vancouver, British Columbia. 1981.

[45] Yang Lyu et al. "Vision Based Autonomous UAV Plane Estimation And Following for Building Inspection". In: *arXiv preprint arXiv:2102.01423* (2021).

[46] Ángel Madridano, Abdulla Al-Kaff, David Martn, et al. "3D trajectory planning method for UAVs swarm in building emergencies". In: *Sensors* 20.3 (2020), p. 642.

[47] Marco Mammarella et al. "Comparing optical flow algorithms using 6-dof motion of real-world rigid objects". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1752–1762.

[48] Michael Manz et al. "Monocular model-based 3D vehicle tracking for autonomous vehicles in unstructured environment". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2465–2471.

[49] Aurelio G Melo et al. "Fuzzy Gain-Scheduling PID for UAV Position and Altitude Controllers". In: *Sensors* 22.6 (2022), p. 2173.

[50] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[51] Chase C Murray and Ritwik Raj. "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones". In: *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 368–398.

[52] Aws Abdulsalam Najm and Ibraheem Kasim Ibraheem. "Nonlinear PID controller design for a 6-DOF UAV quadrotor system". In: *Engineering Science and Technology, an International Journal* 22.4 (2019), pp. 1087–1097.

[53] OHTA Naoya. "Optical flow detection by color images". In: *NEC Research and Development* 97 (1990), pp. 78–84.

[54] Janosch Nikolic et al. "A UAV system for inspection of industrial facilities". In: *2013 IEEE Aerospace Conference*. IEEE. 2013, pp. 1–8.

[55] Naoya Ohnishi and Atsushi Imiya. "Dominant plane detection from optical flow for robot navigation". In: *Pattern Recognition Letters* 27.9 (2006), pp. 1009–1021.

[56] Open Robotics. *Gazebo SIM*. https://gazebosim.org/home. Online; accessed in March 2021. 2021.

[57] Open Robotics. *ROS Robots*. https://robots.ros.org/. Online; accessed in March 2021. 2021.

[58] Open Robotics. *Wiki ROS*. http://wiki.ros.org/. Online; accessed in March 2021. 2018.

[59] Manuel Patchou, Benjamin Sliwa, and Christian Wietfeld. "Flying Robots for Safe and Efficient Parcel Delivery Within the COVID-19 Pandemic". In: *2021 IEEE International Systems Conference (SysCon)*. IEEE. 2021, pp. 1–7.

[60] Milena F Pinto et al. "A framework for analyzing fog-cloud computing cooperation applied to information processing of UAVs". In: *Wireless Communications and Mobile Computing* 2019 (2019).

[61] Milena F Pinto et al. "A Robotic Cognitive Architecture for Slope and Dam Inspections". In: *Sensors* 20.16 (2020), p. 4579.

[62] Milena F Pinto et al. "ARCog: An Aerial Robotics Cognitive Architecture". In: *Robotica* (2020), pp. 1–20.

[63] Milena F Pinto et al. "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle". In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. IEEE. 2017, pp. 1324–1329.

[64] Paolo Bellezza Quater et al. "Light Unmanned Aerial Vehicles (UAVs) for cooperative inspection of PV plants". In: *IEEE Journal of Photovoltaics* 4.4 (2014), pp. 1107–1113.

[65] Gabryel Silva Ramos et al. "Technical and Economic Feasibility Study for Implementing a Novel Mooring-Assisting Methodology in Offloading Operations Using Autonomous Unmanned Aerial Vehicles". In: *SPE Production Operations* (June 2021), pp. 1–16. ISSN: 1930-1863. DOI: 10.2118/205524-PA. URL: https://doi.org/10.2118/205524-PA.

[66] Raúl Rojas. "Lucas-kanade in a nutshell". In: *Freie Universit at Berlinn, Dept. of Computer Science, Tech. Rep* (2010).

[67] Hugo Romero, Sergio Salazar, and Rogelio Lozano. "Real-time stabilization of an eight-rotor UAV using optical flow". In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 809–817.

[68] Robert Ross and Rahinul Hoque. "Augmenting GPS with geolocated fiducials to improve accuracy for mobile robot applications". In: *Applied Sciences* 10.1 (2020), p. 146.

[69] Maurizio Rossi and Davide Brunelli. "Autonomous gas detection and mapping with unmanned aerial vehicles". In: *IEEE Transactions on Instrumentation and Measurement* 65.4 (2015), pp. 765–775.

[70] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[71] Indra Sakti. "Methodology of fuzzy logic with mamdani fuzzy models applied to the microcontroller". In: *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*. IEEE. 2014, pp. 93–98.

[72] Theerasak Sangyam et al. "Path tracking of UAV using self-tuning PID controller based on fuzzy logic". In: *Proceedings of SICE annual conference 2010*. IEEE. 2010, pp. 1265–1269.

[73] M. Selecký et al. "Framework for incremental development of complex unmanned aircraft systems". In: *2015 Integrated Communication, Navigation and Surveillance Conference (ICNS)*. 2015, J3-1-J3–9.

[74] Ganeshsree Selvachandran et al. "A new design of Mamdani complex fuzzy inference system for multi-attribute decision making problems". In: *IEEE Transactions on Fuzzy Systems* (2019).

[75] Shital Shah et al. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: *Field and Service Robotics*. 2017. eprint: arXiv:1705.05065. URL: https://arxiv.org/abs/1705.05065.

[76] Zhexiong Shang and Zhigang Shen. "Vision Model-Based Real-Time Localization of Unmanned Aerial Vehicle for Autonomous Structure Inspection under GPS-Denied Environment". In: *Computing in Civil Engineering 2019: Smart Cities, Sustainability, and Resilience*. American Society of Civil Engineers Reston, VA, 2019, pp. 292–298.

[77] Nusrat Sharmin and Remus Brad. "Optimal filter estimation for Lucas-Kanade optical flow". In: *Sensors* 12.9 (2012), pp. 12694–12709.

[78] Mandyam V Srinivasan. "An image-interpolation technique for the computation of optic flow and egomotion". In: *Biological cybernetics* 71.5 (1994), pp. 401–415.

[79] Changhao Sun et al. "An industrial quadrotor UAV control method based on Fuzzy adaptive linear active disturbance rejection control". In: *Electronics* 10.4 (2021), p. 376.

[80] M Hassan Tanveer et al. "NMPC-PID based control structure design for avoiding uncertainties in attitude and altitude tracking control of quad-rotor (UAV)". In: *2014 IEEE 10th International Colloquium on Signal Processing and its Applications*. IEEE. 2014, pp. 117–122.

[81] Céline Teuliere, Eric Marchand, and Laurent Eck. "3-D model-based tracking for UAV indoor localization". In: *IEEE Transactions on cybernetics* 45.5 (2014), pp. 869–879.

[82] The Guardian. *UK supermarkets predict shortage of lettuce, broccoli and citrus fruit*. https://www.theguardian.com/business/2020/dec/21/sainsburys-port-restrictions-missing-products-uk. Online; accessed in March 2021. 2020.

[83] Austin Thomas and Pavan Racherla. "Constructing statutory energy goal compliant wind and solar PV infrastructure pathways". In: *Renewable Energy* 161 (2020), pp. 1–19.

[84] Jhon Jairo Vega Daz et al. "Solar panel detection within complex backgrounds using thermal images acquired by UAVs". In: *Sensors* 20.21 (2020), p. 6219.

[85] Ban Wang and Youmin Zhang. "Adaptive Sliding Mode Fault-Tolerant Control for an Unmanned Aerial Vehicle". In: *Unmanned Systems* 05.04 (2017), pp. 209–221. DOI: 10.1142/S2301385017400052. eprint: https://doi.org/10.1142/S2301385017400052. URL: https://doi.org/10.1142/S2301385017400052.

[86] Yixuan Wang et al. "Optimization of air–fuel ratio control of fuel-powered UAV engine using adaptive fuzzy-PID". In: *Journal of the Franklin Institute* 355.17 (2018), pp. 8554–8575.

[87] Hao Ying et al. "Comparison of necessary conditions for typical Takagi-Sugeno and Mamdani fuzzy systems as universal approximators". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 29.5 (1999), pp. 508–514.

[88] Xiaozhuoer Yuan et al. "Robust H_∞ Control for Hovering of a Quadrotor UAV with Slung Load". In: *2019 12th Asian Control Conference (ASCC)*. IEEE. 2019, pp. 114–119.

[89] Zhen-Yu Zhao, Masayoshi Tomizuka, and Satoru Isaka. "Fuzzy gain scheduling of PID controllers". In: *IEEE transactions on systems, man, and cybernetics* 23.5 (1993), pp. 1392–1398.

[90] Yuanfan Zheng and Qihao Weng. "Modeling the effect of green roof systems and photovoltaic panels for building energy savings to mitigate climate change". In: *Remote Sensing* 12.15 (2020), p. 2402.

[91] John G Ziegler, Nathaniel B Nichols, et al. "Optimum settings for automatic controllers". In: *trans. ASME* 64.11 (1942).

[92] J-C Zufferey and Dario Floreano. "Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 2594–2599.

Appendix: Optical Flow Script

```
Optical flow - Matlab version

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Optical Flow      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hbm = vision.BlockMatcher('ReferenceFrameSource',...
        'Input port','BlockSize',[35 35]);
hbm.OutputValue = 'Horizontal and vertical components in complex form';
halphablend = vision.AlphaBlender;

opticFlow = opticalFlowLK('NoiseThreshold',0.009);



vidReader = VideoReader('insp2.mp4','CurrentTime',11);




h = figure;
movegui(h);
hViewPanel = uipanel(h,'Position',[0 0 1 1],'Title','Plot of Optical Flow Vectors');
hPlot = axes(hViewPanel);

i = 1;

vectDataMovMod = [0 0 0 ]
vectDataMovOr = [0 0 0 ]
posX = [0 0 0]
posY = [0 0 0]

while hasFrame(vidReader)

    frame = readFrame(vidReader);
    frame = readFrame(vidReader);
    frame = readFrame(vidReader);
    frame = readFrame(vidReader);
    frame = readFrame(vidReader);

    frame = readFrame(vidReader);
```

```
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);

frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);

frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);
frame = readFrame(vidReader);


frame1 = readFrame(vidReader);
frameGray1 = im2gray(frame1);

frame2 = readFrame(vidReader);
frameGray2 = im2gray(frame2);


motion = hbm(frameGray1,frameGray2);
[X,Y] = meshgrid(1:35:size(frame1,2),1:35:size(frame1,1));
subplot(2,2,1)
imshow(frame1)
hold on
quiver(X(:),Y(:),real(motion(:)),imag(motion(:)),.6,'r')

xlabel('Image x position')
ylabel('Image y position')

subplot(2,2,2)
quiver(X(:),Y(:),real(motion(:)),imag(motion(:)),.6,'r')
xlabel('Image x position')
ylabel('Image y position')
```

```
    %optflow
    subplot(2,2,3)
    flow = estimateFlow(opticFlow,frameGray1);
    flow2 = estimateFlow(opticFlow,frameGray2);
    imshow(frame1)
    plot(flow,'DecimationFactor',[5 5],'ScaleFactor',10);

    xlabel('Image x position')
    ylabel('Image y position')

    subplot(2,2,4)
    plot(flow2,'DecimationFactor',[5 5],'ScaleFactor',10);

    plot(posX,posY)

    xlabel('Image x position')
    ylabel('Image y position')

    hold off

    name = ['Barchart' num2str(i) '.png']

    saveas(gcf,name)

    vectDataMovMod(i) = mean(real(motion(:)))
    vectDataMovOr(i) = mean(imag(motion(:)))


    i  =i + 1;

    posX(i) = posX(i-1) + mean(real(motion(:)))
    posY(i) = posY(i-1) + mean(imag(motion(:)))




    pause(10^-3)
end
```