CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Control Strategy Applied to Autonomous UAV Operation in Marine Small Cargo Transport

Guilherme Ferreira Carvalho

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 Alcantara Andrade

Rio de Janeiro,
Junho, 2023

Control Strategy Applied to Autonomous UAV Operation in Marine Small Cargo Transport

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.

Guilherme Ferreira Carvalho

Aprovada por:

Milena Faria Pinto
Assinado de forma digital por Milena Faria Pinto
Dados: 2023.06.20 09:04:32 -03'00'

Presidente, Milena Faria Pinto, D.Sc. (orientadora)

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

gov.br Documento assinado digitalmente
FABIO AUGUSTO DE ALCANTARA ANDRADI
Data: 26/06/2023 05:16:07-0300
Verifique em https://validar.iti.gov.br

Fabio Augusto de Alcantara Andrade, D.Sc. (co-orientador)

University of South-Eastern Norway

gov.br Documento assinado digitalmente
ALESSANDRO ROSA LOPES ZACHI
Data: 20/06/2023 19:10:05-0300
Verifique em https://validar.iti.gov.br

Alessandro Rosa Lopes Zachi, D.Sc.

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

gov.br Documento assinado digitalmente
MURILLO FERREIRA DOS SANTOS
Data: 20/06/2023 12:44:29-0300
Verifique em https://validar.iti.gov.br

Murillo Ferreira dos Santos, D.Eng.

Centro Federal de Educação Tecnológica de Minas Gerais

Rio de Janeiro,

Junho, 2023

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C331 Carvalho, Guilherme Ferreira
Control strategy applied to autonomous UAV operation in marine small cargo transport / Guilherme Ferreira Carvalho. — 2023.
x, 66f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2023.
Bibliografia : f. 65-69
Orientadora: Milena Faria Pinto
Coorientador: Fabio A. de A. Andrade

1. Sistema de controle inteligente. 2. Controle automático. 3. Drones. 4. Transporte marítimo. 5. Petróleo e gás. 6. Transporte de carga. I. Pinto, Milena Faria (Orient.). II. Andrade, Fabio A. de A. (Coorient.). III. Título.

CDD 629.8312

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

ACKNOWLEDGMENTS

To my advisor Milena, for her patience

To my friends, for the fun good times

To my family, for their kind care

To my parents, for everything!

Guilherme Ferreira Carvalho

A scientist in his laboratory is not only a technician, but a child placed before phenomenons that impress like fairy tales.

Marie Curie

RESUMO

Control Strategy Applied to Autonomous UAV Operation in Marine Small Cargo Transport

Guilherme Ferreira Carvalho

Orientadores:
Profa. Milena Faria Pinto
Prof. Fabio Augusto de Alcantara Andrade

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 em Engenharia Elétrica.

Os Veículos Aéreos não Tripuláveis (VANTs) têm sido utilizados em diversas aplicações devido à sua flexibilidade para manobrar e realizar tarefas com trajetos curtos e ângulos de rolagem máximos. Um dos principais objetivos da indústria de petróleo e gás é melhorar o processo de exploração sem perder a eficiência e a qualidade da fabricação. Um dos desafios em manter a estabilidade de voo diante de perturbações, como o vento.

Os VANTs geralmente usam um controlador Proporcional, Integral e Derivativo (PID) clássico. O controlador PID é amplamente utilizado na literatura e na indústria devido a sua estrutura simples e menor dependência do modelo. No entanto, esse controlador clássico exige do operador experiência para ajustar os parâmetros ao lidar com não linearidades. Portanto, este trabalho propõe desenvolver e implementar um controlador Sliding Mode Controller (SMC) a bordo de um software PX4 FCU para VANT e uma técnica de visão computacional para aplicar um VANT autônomo em uma tarefa de carga offshore para o setor de Petróleo e Gás.

A avaliação do controlador foi realizada tanto em um ambiente de simulação realista quanto em testes do mundo real. Os resultados mostram a robustez do controle proposto para o transporte de pequenas cargas com VANTs.

Offshore, Transporte de Carga Pequena, SMC, VANT, Controle Automático.

ABSTRACT

Control Strategy Applied to Autonomous UAV Operation in Marine Small Cargo Transport

Guilherme Ferreira Carvalho

Advisors:

Profa. Milena Faria Pinto

Prof. Fabio Augusto de Alcantara Andrade

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.

Unmanned Aerial Vehicles (UAVs) have been used in diverse applications due to their flexibility for maneuvering and performing tasks with short path lengths and maximum roll angles. One of the main goals of the oil and gas industry is to improve the exploration process without losing manufacturing efficiency and quality. One of the challenges is maintaining flight stability when facing disturbances, such as wind.

UAVs usually use a classical Proportional, Integrative, and Derivative (PID) controller. PID controller is widely used in literature and industry due to its simple structure and less dependence on the model. However, this classical controller requires expertise from the operator to adjust the parameters when dealing with nonlinearities. Therefore, this work proposes developing and implementing an SMC controller aboard a PX4 Flight flight control unit (FCU) software for UAV and a computer vision technique to apply an autonomous UAV in an offshore cargo task for the Oil and Gas sector.

The controller was evaluated in both a realistic simulation environment and with real-world tests. The results demonstrated the robustness of the proposed control strategy for marine small cargo transportation using UAVs.

Offshore, Small Cargo Transport, SMC, UAV, Automatic Control.

Contents

I Introduction **11**

I.1 Initial Considerations and Work Motivation 11

I.2 Work Contributions 13

I.3 Related Publications 14

I.4 Organization 15

II Background and Related Works **16**

II.1 Cargo Transport in Marine Offshore Scenario 16

II.2 UAV Modeling 18

II.3 PID strategy 20

II.4 Sliding Mode Controller 22

II.5 Specialized Tools 26

II.5.1 Robotic Operating System 26

II.5.2 PX4 Autopilot Firmware 27

II.5.3 Gazebo Software 29

II.5.4 Control interface with python 31

II.5.5 Matlab 36

III Proposed Methodology **40**

III.1 Overall Idea 41

III.2 First attempts to improve PID for external disturbances 42

III.3 SMC Control Design and Implementation 44

III.4 Autonomous Trajectory 46

IV Results and Discussion **48**

IV.1 Simulation Environment 48

IV.1.1 Simulation results considering the absence of external disturbances 53

IV.1.2 Simulation results considering the presence of external disturbances 55

IV.2 Autonomous Trajectory Results 59

V Conclusions and Future Work **60**

V.1 Final Conclusions 60

V.2 Future Ideas 61

Bibliographic References 62

List of Tables

II.1 Variables Descriptions used in the set of Quadrotor Equations. 19

III.1 Fuzzy rule (CARVALHO et al., 2021). 43

IV.1 PID parameters 49

IV.2 SMC parameters 50

IV.3 Tuning parameters tested in gazebo simulator 52

List of Figures

II.1 Cargo transport scheme. 17
II.2 Quadrotor UAV Architecture. 18
II.3 Simplified cascade PID controller. 20
II.4 Data exchange in ROS. 26
II.5 Example of ROS nodes and topics. 27
II.6 PX4 control arquitecture. 29
II.7 WPG Software User Interface. 35
II.8 Ros topics without fuzzy. 35
II.9 Matlab fuzzy data flow. 38
II.10 Ros topics with fuzzy. 39

III.1 Overview flowchart. 41
III.2 Scheme of fuzzy addition to the PID controller (CARVALHO et al., 2021). 42
III.3 SMC implemented in the PX4 Firmware. 45
III.4 Linear trajectory. 46

IV.1 Quadrotor used in the Gazebo software. 48
IV.2 World created in the Gazebo software. 49
IV.3 Tests conducted in three stages: (a) Take off. (b) Approaching vessel. (c) Landing. . . . 50
IV.4 Position error without disturbances for X-axis (a,d,g), Y-axis (b,e,h) and Z-axis (c,f,i) . . 53
IV.5 Derivative error without disturbances for X-axis (a,d,g), Y-axis (b,e,h) and Z-axis (c,f,i) . 54
IV.6 Position tracking comparing SMC and PID controllers without disturbances 55
IV.7 Position tracking comparing SMC and PID controllers with disturbances 56
IV.8 Position error with disturbances for X-axis (a, d, g), Y-axis (b, e, h), and Z-axis (c, f, i). 57
IV.9 Derivative error with disturbances for X-axis (a, d, g), Y-axis (b, e, h), and Z-axis (c, f, i). 58
IV.10 Comparison of the SMC and PID controllers in autonomous trajectory simulation 59

List of Abbreviations

DOF	Degrees Of Freedom . . . 12
EKF	Extended Kalman Filter . . . 47
FCU	Flight Control Unit . . . vi, 13
FPSO	Floating, Production, Storage, And Offloading . . . 11
GPS	Global Positioning System . . . 28
GUI	Graphical User Interface . . . 28
IDE	Integrated Development Environment . . . 28
IMUS	Inertial Measurement Units . . . 28
LACEA	Laboratório De Controle E Automação . . . 13
MAE	Maximum Average Error . . . 58
MAVLINK	Micro Air Vehicle Link . . . 28
MIMO	Multiple-Input Multiple-Output . . . 11
NWU	North, West, Up . . . 21
ORB	Object Request Broker . . . 26
P2P	Peer-to-Peer . . . 26
PID	Proporcional, Integral E Derivativo . . . v, 20
ROVS	Remotely Operated Vehicles . . . 16
RPC	Remote Procedure Call . . . 27
SITL	Software In The Loop . . . 30
SMC	Sliding Mode Controller . . . v, vi, 12, 25
UAV	Unmaned Aerial Vehicle . . . 11, 12, 13, 18, 20, 21, 25
URDF	Unified Robot Description Format . . . 29
VANTS	Veículos Aéreos Não Tripuláveis . . . v
YOLO	You Only Look Once . . . 13

Chapter I Introduction

I.1 Initial Considerations and Work Motivation

Unmaned Aerial Vehicle (UAV) has been used in diverse applications. For instance, they are used in the inspection of large structures (MELO et al., 2020; PINTO et al., 2020b), search and rescue activities (PINTO et al., 2020a), delivery goods (MURRAY; RAJ, 2020; MADRIDANO et al., 2020), security surveillance (PINTO et al., 2017b; PINTO et al., 2018), etc. UAVs present flexibility for maneuvering and performing tasks with short path length and maximum roll angle (MELO et al., 2021). One of the main goals in the oil and gas industry is to improve the exploration process without losing manufacturing efficiency and quality (STAVINOHA et al., 2014). Therefore, applying robots in this sector is a prominent solution to replace tasks that can be unsafe for humans or cause workers depletion (ANDRADE et al., 2022).

A few works have applied UAVs to automate tasks in the Offshore industry, (WEN et al., 2018; RAMOS et al., 2022b). According to (WANASINGHE et al., 2020), UAVs offer a faster, safer, and more cost-efficient way of mass data collection, which can directly impact the reduced costs in a few applications. For instance, in (RAMOS et al., 2022a), the authors applied an autonomous UAV to deliver a message cable from the shuttle tanker to the Floating, Production, Storage, and Offloading (FPSO) unit. In (DAI et al., 2022), the authors investigated the application of UAVs for data offloading for smart containers in offshore maritime communications. The UAV relays on a node between smart containers and an onshore base station in their application. For a complete overview of UAVs remote sensing in the oil and gas industry, the authors refer to (ASADZADEH; OLIVEIRA; FILHO, 2022).

Note that the selection of UAVs depends on different factors, including the coverage area, distance from the base station, flight duration, and the sensors aboard the UAV (ASADZADEH; OLIVEIRA; FILHO, 2022). Cameras by far are the most widely used sensors aboard UAVs (MAES; STEPPE, 2019). By using computer vision techniques along with a self-controlling function to fly in an unstructured environment to perform autonomously, it is possible to use UAV's versatility to improve and substitute the riskier jobs, such as the inspection of power and transmission lines (GUAN et al., 2021), dams (BIUNDINI et al., 2021), or buildings (TAN et al., 2021). Other tasks that autonomous UAVs can perform are the transport of materials or equipment between predefined locations, like offshore landing platforms on ships. In this case, computational vision techniques could be used to identify landing zones (NGUYEN et al., 2018).

For autonomous or semi-autonomous operations, the UAV should also be capable of performing these missions in a stable way. The control design for quadrotor UAVs has two main challenges. The first challenge regards the fact that the quadrotor is a Multiple-Input Multiple-Output (MIMO)

unstable nonlinear system, presenting four actuators and six Degrees of Freedom (DoF). The second issue is that this kind of robot faces different difficulties, such as external and internal disturbances, uncertainty in the model, and parametric perturbations (ALMAKHLES, 2019). In this sense, many attempts from the literature suggested the use of different robust approaches to improve the robustness of the UAV controller in order to compensate for the external disturbances effects (THANH; HONG, 2018; ANDRADE et al., 2022; ELTAG; ASLAMX; ULLAH, 2019; HU; LIU, 2020), such as feedback linearization (LEE; KIM; SASTRY, 2009; MARTINS; CARDEIRA; OLIVEIRA, 2021), dynamic inversion (DAS; SUBBARAO; LEWIS, 2009; ANSARI; BAJODAH, 2017), sliding mode control (XU; OZGUNER, 2006; AHMED; CHEN, 2018), backstepping (DAS; LEWIS; SUBBARAO, 2009; HUO; HUO; KARIMI, 2014), and others.

The external disturbances can cause physical instability of the UAVs platform, which commonly uses a classical PID controller with a simple structure, good stability, and less dependence on the exact system model (ANDRADE et al., 2022). However, the PID process of adjusting the parameters requires expertise, especially when dealing with nonlinearities. For instance, the authors of work (JOYO et al., 2013) proposed using a self-tuned PID control method to handle external disturbances in a quadrotor UAV.

In (CARVALHO et al., 2021), the authors proposed using a PID and Fuzzy Logic hybrid controller. The Fuzzy logic was used to schedule the PID controller. Another kind of controller for UAV stabilization was used in (NGUYEN et al., 2021). In their work, the authors proposed a sliding mode control based on neural networks for the attitude and altitude system of a quadcopter under external disturbances. The controller evaluation was performed by numerical simulation.

The SMC arises as a robust control design to deal with uncertain conditions for complex high-order nonlinear systems (BENALLEGUE; MOKHTARI; FRIDMAN, 2008; CAI et al., 2019; MOFID; MOBAYEN, 2018; MOBAYEN, 2014). A main advantage of the SMC technique is its low sensitivity to disturbances and system parameter variations, being widely employed for trajectory tracking control algorithms (RÍOS; GONZÁLEZ-SIERRA; DZUL, 2017).

The authors of (MOFID; MOBAYEN, 2018) proposed using an adaptive sliding mode control strategy to stabilize quadrotor UAV that presents parametric uncertainty.

Another use of the SMC strategy is proposed in (CAI et al., 2019). In their work, the authors proposed the use of a sliding-mode-observer-based equivalent-input-disturbance approach to handle the disturbance suppression of quadrotor UAVs. Both works evaluated their results using numerical simulation analysis.

The controller is the main component in this entire hardware setup known as the brain of this under-actuated quadcopter. The controller's firmware has many parts working in parallel that are responsible for maintaining the UAV stability and desired position. Citing just a few of the parts, there are the

outermost position controller, the speed controller, the angular controller, and the rate controller which are responsible for the main cascade control loop which is best explained in (DEVS, 2020). In addition, we also have the "motor mixer" responsible for distributing the output values to the UAV actuators so that the propellers behave correctly, which is better known as control allocation.

Another important component that is in the middle of the control cascade is the world-to-body conversion module that translates the coordinates passed with the global frame reference to the UAV reference frame. The most difficult part when modifying this controller is understanding how all the modules communicate and where each modification should be done to achieve the desired result or behavior.

The FCU is the piece of hardware that encapsulates all the sensors and processing cores that the control software will use to generate the desired outputs.

I.2 Work Contributions

The main contribution of this work is developing and implementing an SMC controller in a PX4 flight control software for quadrotor UAV for position control. Besides, this work also proposed the use of this SMC-based controller and along with a computer vision technique to apply an autonomous UAV in an offshore cargo task for the Oil and Gas sector. As already mentioned, in (RAMOS et al., 2022b), the previous work published by the research group of the Laboratory Laboratório de Controle e Automação (LACEA) has proposed a framework that fuses flight data information from UAV's sensors and an object detection algorithm based on You Only Look Once (YOLO)v3 to deploy the vehicle from an oil rig and safely reach a shuttle tanker vessel nearby. The context of such a proposal is to assist the mooring process in offloading operations. A thin messenger line must be carried from the vessel to the production facility so it can be tied.

Besides being complete, the work fails to analyze stability and environmental effects on the UAV's flight. It proposes a solution for this issue, focusing on developing and simulating the hybrid approach. The present work takes a step ahead, covering control solutions to deal with wind and flight instabilities during this operation using the SMC controller. The work also considers a classical PID controller for comparison purposes. The main contributions can be summarized as follows:

(I) Implementation of a SMC controller in a PX4 FCU software for quadrotor UAV for position control;

(II) Development of an SMC-based and Computer Vision technique that enables the UAV to fly autonomously between vessels to deliver materials;

(III) Experimental tests are carried out in a realistic simulation environment using the Robot Operating System (ROS)/Gazebo platforms to verify the performances of the proposed methodology;

I.3 Related Publications

During the master's dissertation, the student has helped with the following related published papers:

Published Conference Paper:

- Guilherme Carvalho, Ihannah Guedes, Milena Pinto, Alessandro Zachi, Luciana Almeida, Fabio Andrade, Aurelio G Melo. "Hybrid PID-Fuzzy controller for autonomous UAV stabilization". 14th IEEE International Conference on Industry Applications (INDUSCON). Available at: (CARVALHO et al., 2021).

Published Journal 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: (MELO et al., 2022).

- 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: (ANDRADE et al., 2022).

Submitted Journal Paper:

- Guilherme Carvalho, Fabio Andrade, Gabryel S. Ramos, Alessandro Zachi, & Milena F. Pinto. Sliding Mode Controller Applied to Autonomous UAV Operation in Marine Small Cargo Transport. Submitted to IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS.

I.4 Organization

The rest of this dissertation is organized as follows. Chapter II gives the necessary background for cargo transport in Oil and Gas industry. Also, this section presents related works in employing SMC controllers for quadrotor UAV controlling.

Chapter III shows the mathematical foundations for the control strategy and its implementation in a real flight control unit.

The results obtained through the methods discussed are shown in Chapter IV. In addition, the necessary discussions will be held about the results to demonstrate the efficiency of the obtained control strategy for small cargo transport.

The concluding remarks of this research work and the various possibilities for continuing this work are given in Chapter V.

Chapter II Background and Related Works

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

II.1 Cargo Transport in Marine Offshore Scenario

UAVs are suitable for military missions, agriculture, surveillance, mapping, aerial photography, resource exploration, traffic monitoring, and anti-terrorism investigation, among others (AHMED et al., 2015; PINTO et al., 2020a; PINTO et al., 2020b). When dealing with more general classes of tasks and environments, one major challenge is to maintain flight stability in the face of the vehicle nonlinear dynamics (LIU et al., 2020). In general, The prototypes manufactured in series for performing fixed payloads tasks are designed to keep stability at safety margins.

Currently, the use of UAVs in general transport applications is consolidated and well documented, as established in (GUPTA et al., 2021). The operations may vary from parcel delivery to human transportation (KELLERMANN; BIEHLE; FISCHER, 2020), enhancing the logistic chain efficiency, reducing costs compared to conventional delivery options, and reducing environmental damage by eliminating the emission of polluting gases. The study of UAV controllers and aircraft designs reduced the manufacturing costs of these vehicles and advanced the technology for more audacious applications, such as the rapid transportation of human organs in medical emergencies (SCALEA et al., 2021).

The naval industry has used robotic frameworks in several applications since the 70's decade, mainly in the form of Remotely Operated Vehicles (ROVs) for underwater inspections (PETILLOT et al., 2019). Recently, the versatility of UAVs and aerial robotics systems caught the attention of the naval industry for small parcels transportation (STAVINOHA et al., 2014), naval operations assistance, such as in ship maneuver on ports (ENGINEERS, 2018) and offloading hose connection (RAMOS et al., 2022b), an inspection of naval structures (DURDEVIC et al., 2019) and marine search and rescue (QINGQING et al., 2020). One application in particular that is drawing attention in this sector is cargo transport. In fact, traditional marine logistics is restricted to transshipment by boats, vessels, and helicopters.

Usually, the aerial alternative is used for small cargo. However, it presents some restrictions for transportation and navigation (HLAS; STRAUB, 2016). For instance, there is a restriction to transporting some packages, such as flammable petrochemical samples, cutting or piercing tools, explosive components (i.e., batteries), etc. Using boats for transport demands the manipulation of cargo by cranes (even when it is lightweight cargo) or launching the cargo in water (for example, in offloading operation mooring and connection stages). In this sense, UAV represents an attractive alternative for small cargo transport, as it could easily replace the conventional means for small cargo

transport between near naval structures, eliminating traditional restrictions at the same time it improves efficiency (FAUST et al., 2017; LIERET et al., 2019).

In the context of this research work, an autonomous UAV is applied to an offshore cargo task for the Oil and Gas sector. An SMC controller is implemented in the PX4 flight control software for the quadrotor in order to perform position control. To test the controller's performance in the simulation was added a navigation process based on the article (RAMOS et al., 2022b) that used a hybrid approach based on deep learning and sensor fusion for the path planning of autonomous UAVs. Figure II.1 presents the operation in marine small cargo transport performed by a UAV between a shuttle tanker and an FPSO. This is a complex and difficult marine operation due to the windy disturbance and difficulty of reaching a safe place to land or drop the cargo.

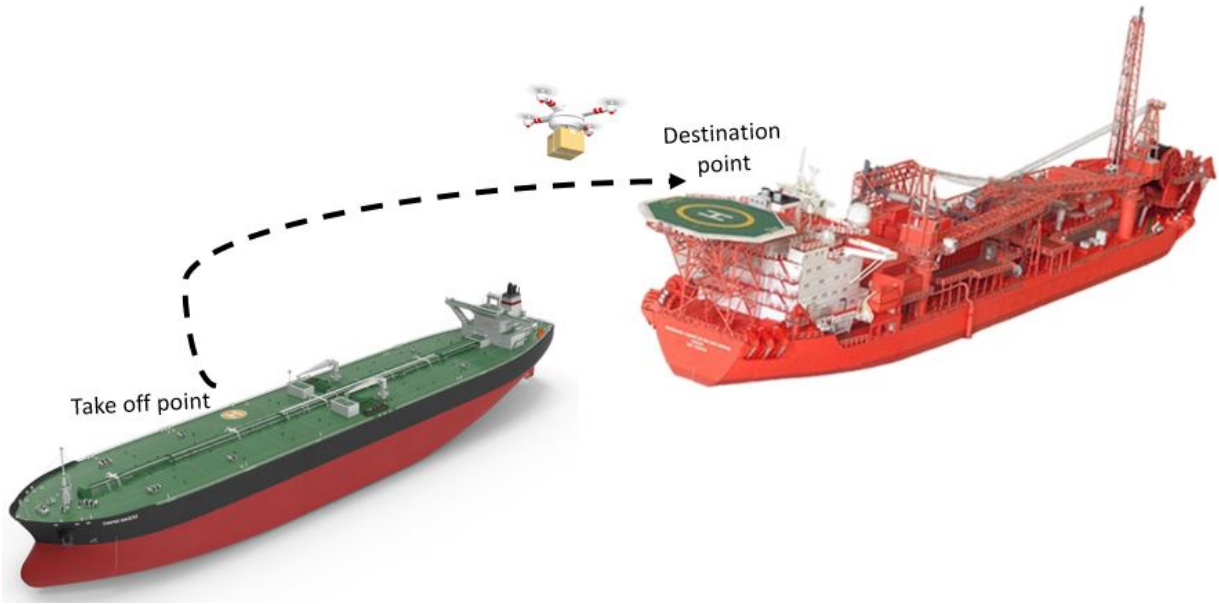


Figure II.1: Cargo transport scheme.

II.2 UAV Modeling

Parts of this section derived from the paper entitled "Hybrid PID-Fuzzy Controller for Autonomous UAV Stabilization", published in INDUSCON 2021 (CARVALHO et al., 2021). In the last years, many tasks were 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 (AHMED et al., 2015; PINTO et al., 2020a; PINTO et al., 2020b).

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. It is important to consider the quadrotor as a nonlinear unstable system that comprises four actuators whose thrust is treated as an input to the entire system when deriving the dynamic model. Besides, this work considers a quadrotor body frame as a rigid and symmetrical structure.

Figure II.2 presents a widespread quadrotor control architecture. This architecture is better described in the article (DEVS, 2020). The approach as described before was to modify the controller logic on the "Angular Rate Control" block in order to implement an SMC-based controller in place of the current PID controller. Table II.1 presents the variables descriptions used in the set of equations for modeling the quadrotor.

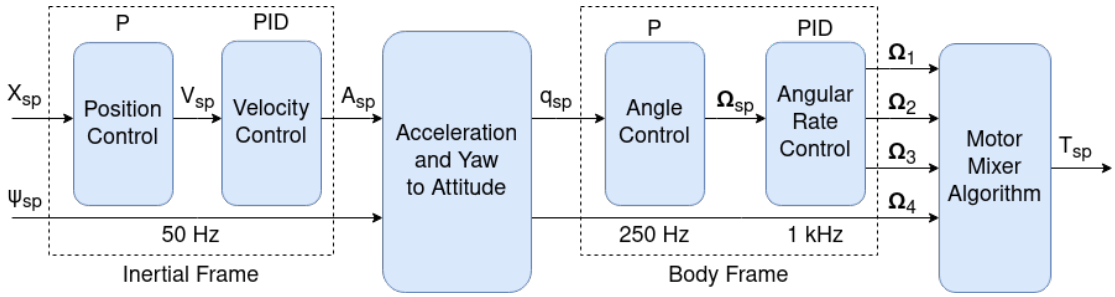


Figure II.2: Quadrotor UAV Architecture.

Table II.1: Variables Descriptions used in the set of Quadrotor Equations.

Variable	Description
ϕ, θ and ψ	Euler angles *roll*, *pitch* and *yaw*
Ixx, Iyy, $Izz(kgm^2)$	Moment of inertia
$m(kg)$	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
Ω_r	Engines residual angular speed
$\xi_{(\phi,\theta,\psi,h)}$	Euler angles disturbances
U_1	Thrust equivalence force
U_2	Rotational force on roll axis
U_3	Rotational force on pitch axis
U_4	Rotational force on yaw axis

The expression in Equation II.1 represents the mathematical model of the quadrotor with respect to the body frame's axes. 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.

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \ddot{z} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{zz}}\dot{\theta}\dot{\psi} + \theta\frac{J_r}{I_{xx}}\Omega_r + \frac{l}{I_{xx}}U_2 \\ \frac{I_{zz}-I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} + \phi\frac{J_r}{I_{yy}}\Omega_r + \frac{l}{I_{yy}}U_3 \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{l}{I_{zz}}U_4 \\ g - \frac{\cos(\phi)\cos(\theta)U_1}{m} \\ \frac{U_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \frac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) + \sin(\phi)\cos(\psi)) \end{bmatrix} = \begin{bmatrix} \xi_\phi(t) \\ \xi_\theta(t) \\ \xi_\psi(t) \\ \xi_h(t) \\ 0 \\ 0 \end{bmatrix} \tag{II.1}$$

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

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

Several control platforms were developed to increase the UAVs' degree of autonomy to allow them to perform tasks simultaneously in a semi-autonomous or autonomous way (Selecký et al., 2015; PINTO et al., 2017a). 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. However, PID controller can not deal with systems that are not easy to be modeled because of their nonlinearities, as presented in (DOITSIDIS et al., 2004; DONG; HE, 2019; WANG et al., 2018; NAJM; IBRAHEEM, 2019).

II.3 PID strategy

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

The diagram of Figure II.3 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, proving that the roll and pitch angles converge to their (generated) reference values. Then, Equation II.2 is used to compute the rotors' velocity.

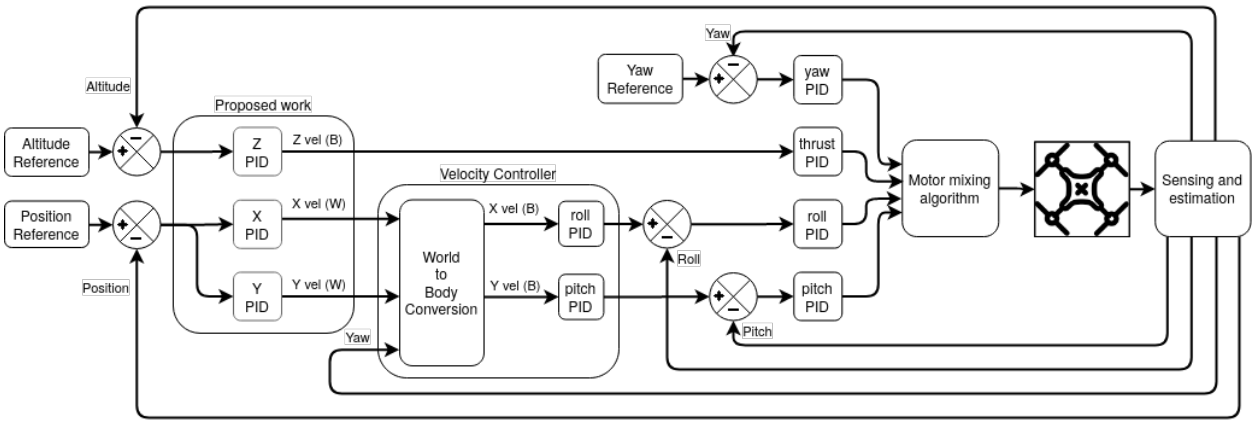


Figure II.3: Simplified cascade PID controller.

Source: published in (CARVALHO et al., 2021)

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.3.

$$\begin{bmatrix} X_{world} \\ Y_{world} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} X_{body} \\ Y_{body} \end{bmatrix} \tag{II.3}$$

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

$$\begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{body} \\ Y_{body} \\ Z_{body} \end{bmatrix} \tag{II.4}$$

The matrix of Equation II.4 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 roll and pitch rotations representation, there are the matrices $R_{X,\phi}$ and $R_{Y,\theta}$, respectively. Equation II.6 shows a complete transformation of the three roll, pitch, and yaw angles. Such matrix product is not commutative, which means that the product order matters and this transformation is done by parts, from right to left, and it is denoted by $R_{\psi,\theta,\phi} = R_{Z,\psi} \cdot R_{Y,\theta} \cdot R_{X,\phi}$.

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

As each column of $R_{\psi,\theta,\phi}$ represents where the base vectors would be located after a rotation, A way of finding the direction in which the UAV is positioned about the world frame is through the Equation II.7 whose parameters are the angles of ϕ, θ and ψ.

This equation is simply the first two lines of the third column of the matrix $R_{\psi,\theta,\phi}$ because the third column of $R_{\psi,\theta,\phi}$ shows us where [0, 0, 1] would be after the coordinate transformation. As this uses North, West, Up (NWU) coordinates for the equations (X is the front/north and Y is the left/west), it computes the direction in which the drone is tilted at any moment.

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

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 $[u_x, u_y]$ within the world frame.

In recent years, many researchers have worked on UAV control strategies. For example, the authors of Joyo et al. (JOYO et al., 2013) have used a self-tuned PID control method to deal with external disturbances in a quadrotor UAV. In Tanveer et al. (TANVEER et al., 2014), 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 (WANG; ZHANG, 2017) is a fault-tolerant strategy considering system uncertainties and actuator failures.

In (Yuan et al., 2019), the authors proposed a flight control for a quadrotor UAV for hovering with a slung load attached. 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 (Liu et al., 2020), (ELTAG; ASLAMX; ULLAH, 2019) and (HU; LIU, 2020).

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, the process of adjusting its parameters deserves attention from the designer, mainly when nonlinearities of the model are present.

The SMC is an alternative to the classical PID controller applied by several hardware manufacturers and researchers. Unlike the PID controller, the SMC controller has the advantage of a faster response against external disturbances to the system and can withstand more significant errors in the structural modeling of the system. Therefore, this controller is considered more robust and relatively simple to implement (CHEN et al., 2016; ABRO et al., 2021; MOFID; MOBAYEN, 2018). Therefore, this research will use a Sliding Mode Control strategy for the proposed application.

II.4 Sliding Mode Controller

The sliding surface of the SMC controller defines a barrier in the state space in which the change of the control structure applied to the model will be made in order to guarantee a unique behaviour for the system. The biggest problem with this controller change is that it can cause a chattering effect when it is too abrupt.

The chattering effect is when the controller quickly changes its operation mode between very different operating states, causing unnecessary energy expenditure and wearing out the actuators. In addition, the controller and actuator cannot physically respond at the same speed as there are response delays between the actuator and controller (ALATTAS et al., 2022), what generates even more instability.

Regarding the SMC control structure, the relative degree of the constraint function $s(x)$ and the order of an SMC is the dimension of the state constraints vector. The relative degree refers to the shape of the sliding surface in relation to the state space. For example, a relative degree of 1 means that the state space on which the sliding surface depends is composed of only one state of the system. The sliding surface is like a point since order 2 depends on two states and would be a line. At the same time, the 3-state sliding surface would be a plane and so on. The SMC control strategy has many more advantages than other control designs. However, it faces the phenomenon of chatter. This problem occurs due to the high number of oscillations that appear in the four rotors of the four-engine UAV (LEE; JANG; KIM, 2020).

The SMC is an attractive controller since it can deal with nonlinear systems with parametric uncertainties subjected to external disturbances. Several hardware manufacturers and researchers have used it in applications of dynamical systems with hard nonlinearities. In such applications, unlike the PID controller, the SMC controller has the advantage of faster response and can withstand significant errors in system modeling. This robustness property makes the technique very attractive in many control applications (CHEN et al., 2016; ABRO et al., 2021; MOFID; MOBAYEN, 2018).

In SMC design formalism, the commonly chosen sliding surface $s(X)$ is a function of the state variables $X = [x, \dot{x}, \cdots, x^{(n-1)}]^T$, namely:

$$s(X) = x^{(n)} + \lambda_{n-1}x^{(n-1)} + \cdots + \lambda_1\dot{x} + \lambda_0 x = 0 \tag{II.8}$$

The surface in Equation (II.8) defines a region in state space where the control structure changes will be applied. Once the state trajectories reach this surface, it remains invariant, that is, $s(X) = 0$, and then the closed-loop system exhibits a desirable behavior which is defined by the coefficients λ_i $(i = 1, \cdots, n-1)$ in (II.8). In the following, is briefly shown that such behavior can be performed via a switched-type control law in the form

$$u := -\rho\ sign[s(X)] \tag{II.9}$$

with

$$\rho > 0 \in \Re\,, \qquad sign[s(X)] = \begin{cases} -1\,, & s(X) < 0\,, \\ 0\,, & s(X) = 0\,, \\ 1\,, & s(X) > 0\,. \end{cases} \tag{II.10}$$

Consider, for example, a class of second-order systems in the format

$$\ddot{x} = u + g(t) \tag{II.11}$$

in which $x \in \Re$ is the system output, $u \in \Re$ is the system control input, and $g(t) \in \Re$ is a function that represents some disturbance term. Here, the main assumptions are:

(A1) A real upper bound constant $\rho > 0$ is known for $|g(t)|$, i. e., $\rho > |g(t)|$.

(A2) The state variables $\dot{x}$ and x are available.

If the control input is chosen as

$$u = -\lambda \dot{x} \; - \; \rho \; sign(s) \, , \quad \lambda > 0 \tag{II.12}$$

$$s = \dot{x} + \lambda x \tag{II.13}$$

then, the closed-loop system becomes

$$\ddot{x} + \lambda \dot{x} \; = \; \dot{s} \; = \; -\rho \; sign(s) + g(t) \tag{II.14}$$

For analyzing the stability of the previous closed-loop system, is chosen the following Lyapunov function candidate

$$V(s) = \frac{1}{2} s^2 \tag{II.15}$$

The time derivative of (II.15) over the trajectories of (II.14), is given by

$$\dot{V}(s) = s \; \dot{s} = -\rho |s| \; + \; sg(t) = -|s| \; [\; \rho + g(t) \; sign(s) \;] \tag{II.16}$$

Since the magnitude of the term '$g(t) \; sign(s)$' can not overcome the magnitude of ρ, as assumed by Assumption $(A1)$, then $\dot{V} < 0$. The last condition guarantees that $s \to 0$ for some finite time $t = t_f$. Thus, after the system trajectories (II.11) reach the sliding surface $s = 0$ (II.13) for $t \geq t_f$, one has that

$$\dot{x} + \lambda x = 0 \tag{II.17}$$

which does imply that $x \to 0$ exponentially fast.

As the aforementioned structural change occurs via a switched-type control law, the abrupt changes that can cause unwanted oscillations are known as *chattering* (SLOTINE, 1991). This technique's known disadvantage can become accentuated if there are response delays between the actuator and the controller (ALATTAS et al., 2022). A well-known fact in the literature is that the chattering phenomenon can be avoided by adopting a smooth control law instead of the switching one in Equation (II.12). However, with such smoothing, the closed-loop system loses the sliding property. In this case, the trajectories of the system do not converge to the surface $s = 0$ but lie in a residual region around it that is denoted by *boundary layer* (SLOTINE, 1991). For instance, if a smooth control law

$$u = -\lambda \dot{x} \; - \; \rho \; tanh(s) \, , \tag{II.18}$$

is used instead of its switched-type version in Eq. (II.12), then the Lyapunov function derivative of Eq.

(II.16) would have the following format

$$\dot{V}(s) = s\ \dot{s} = -|s|\ \ [\ \rho|tanh(s)| + g(t)\ sign(s)\ ] \tag{II.19}$$

One can note that by adopting such a smoothing approach, the magnitude of the term '$\rho|tanh(s)|$' decreases with s and can not always overcome the magnitude of the disturbance term'$g(t)\ sign(s)$' as before. In this case, $\dot{V}(s) < 0$ can be guaranteed only outside a certain region around the origin defined by

$$|\rho\ tanh(s)| > |g(t) \tag{II.20}$$

which, as mentioned, is well-known as the *boundary layer* region (SLOTINE, 1991). In the following, the SMC is applied in the control of a UAV for the Cargo Transporting problem.

II.5 Specialized Tools

II.5.1 Robotic Operating System

This system is a framework developed by Stanford University. This framework was created to standardize research and distribute tools through a robust and reliable development system. ROS is a collection of software frameworks for developing robots, enforcing the functionality of an operating system. Simply put, ROS has a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robotic behavior on various robotic platforms. ROS resembles the *framework* of Object Request Broker (ORB), which is responsible for registering the nodes, managing the communication between them, and ensuring that messages are sent and received correctly. Thus, in ROS, there is a node called the master, responsible for assigning names and records to the nodes and the necessary connections for communications, like registering the advertising of a new topic and allowing other nodes to subscribe to specific topics. With this, the nodes can locate each other to carry out their tasks, communicating in a Peer-to-Peer (P2P) way. Note that the exchange of messages between nodes occurs between communication channels called topics. This communication channel is created to convey a specific message type, facilitating communication organization. This process is best described in Figure II.4.

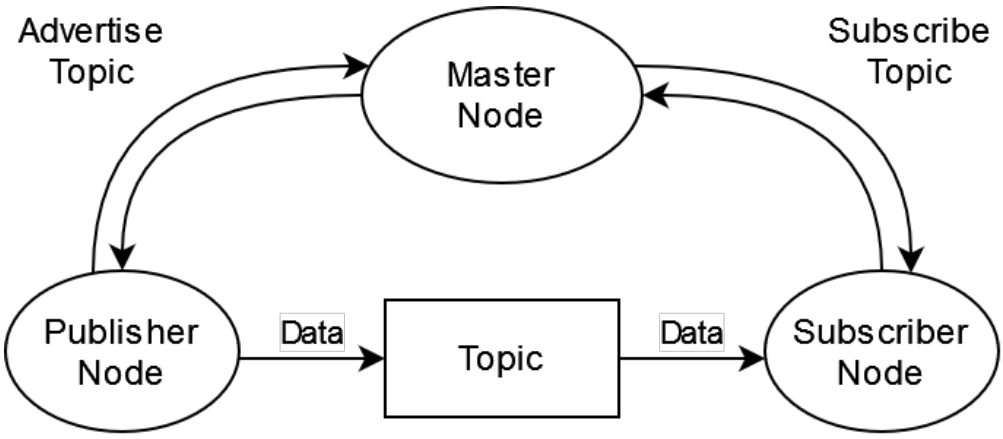


Figure II.4: Data exchange in ROS.

In summary, the main concepts regarding ROS are nodes, topics, and services, being:

- **Nodes:** These are processes that perform some computational calculation or operation. They are combined in a graph, being able to communicate through topics. Their main advantage is that they can work together to perform complex tasks based on simpler objectives, such as having a node responsible for processing data from a LiDAR sensor. At the same time, another is responsible for building a map of the location. From this data, in addition to a third party planning the route to reach a given objective and a last one taking care of sending commands

to the actuators to carry out the locomotion, activating the H bridge responsible for feeding the motors of the robot wheels.

- **Topics:** Nodes use information buses to exchange messages with each other. As they use the publisher/subscriber protocol, the production of their information is decoupled from their consumption. This is very important because, in general, the nodes are not interested in who is communicating. After all, for them, it is much more relevant than just the type of information they receive; therefore, they "subscribe" to the related topics to the data they are interested in. We create this data, generate data, and "publish" it in the corresponding topics. The same topic can have several editors and subscribers.

- **Services:** The publish/subscribe model, or *publisher/subscriber*, is a very flexible communication paradigm. However, due to its "many-to-many" model, it does not meet the demands of Remote Procedure Call (RPC) *request/reply interactions* that are usually necessary for a distributed system. To solve this problem, services were created. They define a pair of request/response messages to send an appropriate response message when they receive a request from some node. In the same way, a call to some remote procedure would work.

As an example, Figure II.5 has the nodes *sensor* and *actuator* communicating through the topics *temp* and *accel*. In this case, the node *sensor* publishes in the topics *temp* and *accel*, while the node *actuator* subscribes to them, thus reading the information contained in them. We can view nodes and topics this way through a ROS module called *rqt_graph*.

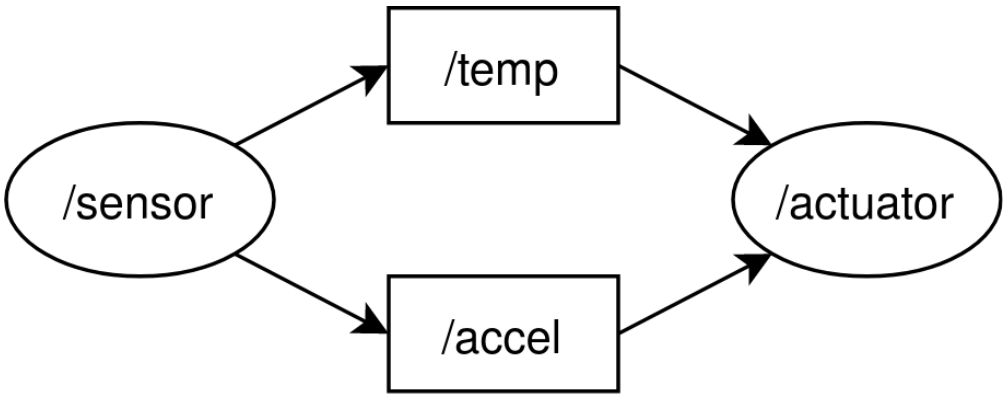


Figure II.5: Example of ROS nodes and topics.

II.5.2 PX4 Autopilot Firmware

PX4 Autopilot is a highly versatile and customizable open-source platform for developing autonomous drones and other unmanned vehicles. It was designed to focus on flexibility, modularity, and extensibility, allowing developers to tailor the software to their specific needs and requirements. PX4 Autopilot is

built on top of the NuttX real-time operating system, which provides a stable and reliable foundation for the software.

One of the key advantages of PX4 Autopilot is its support for a wide range of hardware platforms, sensors, and peripherals. The software can be run on various microcontroller and microprocessor platforms, including ARM Cortex-M, ARM Cortex-A, and x86 architectures. PX4 Autopilot also supports a variety of sensors, including Global Positioning System (GPS), Inertial Measurement Units (IMUs), magnetometers, barometers, and more. This makes it possible to develop and deploy autonomous drones and other unmanned vehicles for a variety of applications and use cases.

Another advantage of the PX4 Autopilot is its support for multiple flight modes. The software supports manual, stabilized, and autonomous flight modes and other specialized modes for specific use cases. This allows developers to create sophisticated control systems that can adapt to different operating environments and flight conditions.

PX4 Autopilot also provides a rich set of tools for debugging and analyzing performance. Developers can use the software's built-in logging and telemetry features to collect data on the behavior of the autopilot and its various components and then analyze this data to identify areas for improvement. The software also includes remote debugging and profiling support, allowing developers to connect to the autopilot over a network and analyze its performance in real-time.

Developing new software for PX4 Autopilot is a straightforward process. The platform is built using the Micro Air Vehicle Link (MAVLink) messaging protocol, which allows developers to communicate with the autopilot and send commands and messages to its various components. Developers can use MAVLink to create their custom messages and commands, which can be used to control the autopilot and interact with its multiple sensors and other components.

To develop new software for PX4 Autopilot, developers typically start by setting up a development environment on their computers. This involves installing the necessary software libraries, compilers, and tools needed to compile and upload new software to the autopilot. Once the development environment is set up, developers can use a text editor or an Integrated Development Environment (IDE) to write their code.

The next step is to compile the code and upload it to autopilot. This is typically done using a command-line interface or a Graphical User Interface (GUI) provided by the development environment. Once the code is uploaded, it can be executed on autopilot, and its performance can be tested in various ways.

One way to test the performance of new software on PX4 Autopilot is to run simulations using Gazebo. Developers can create a virtual environment in the Gazebo that simulates the behavior of the autopilot and its various components and then test their new software in this environment. This allows developers to test their software in a safe and controlled environment before deploying it on a real drone.

Another way to test new software on PX4 Autopilot is to perform flight tests using a real drone. Developers can connect to the autopilot using a ground station, such as QGroundControl, and send commands and messages to the autopilot to control its behavior. This allows developers to test their software in real-world conditions and observe its performance under different scenarios.

PX4 Autopilot is a powerful and flexible platform for developing autonomous drones and other unmanned vehicles. Its modular architecture, support for multiple flight modes, and rich set of tools for debugging and analyzing performance make it an ideal platform for flight control software.

Its default control architecture for quadcopter UAVs is of a four steps cascade control PID, with the steps being position, velocity, angle, and angular rate. Note that between the velocity and the angular controllers, there is a world-to-body converter in order to transpose the correct coordinates from the world frame to the drone frame (DEVS, 2020).

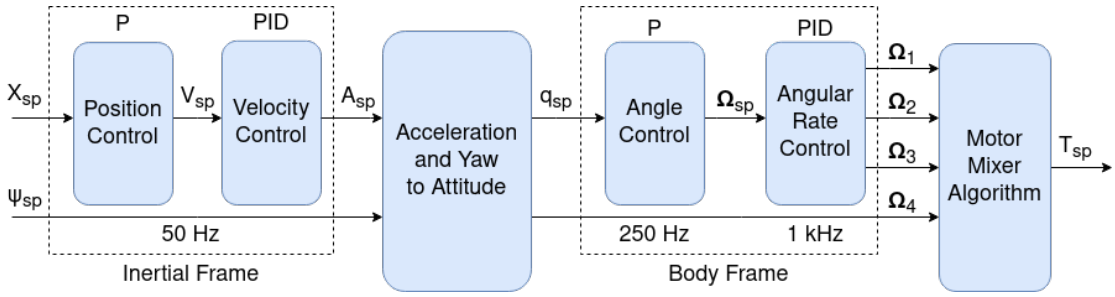


Figure II.6: PX4 control arquitecture.

Please, about the sensors and actuators, consult the following guides about Sensors and Actuators.

II.5.3 Gazebo Software

Gazebo is a free, open-source multi-robot simulator used for complex simulating systems, such as UAVs, to test their functionality in a safe and controlled environment. The software is widely used in the field of robotics and UAVs to test and validate algorithms, sensors, and control systems before actual deployment.

Gazebo allows users to create and customize virtual environments with sensors, actuators, and other components necessary to simulate UAV flight tests. These virtual environments can be modified to simulate different weather conditions, terrains, and obstacles, allowing users to test how their UAVs will perform in various situations.

To simulate UAV flight tests in Gazebo Using SITL, users first create a virtual model of their UAV using a robot description language, such as Unified Robot Description Format (URDF). The model should include all the sensors, actuators, and other components necessary to replicate the real-world UAV.

Next, users can design a virtual environment for the UAV to operate in. This environment can be

customized to replicate real-world scenarios such as landing and takeoff, mission planning, and obstacle avoidance. Users can also introduce wind, rain, and other weather conditions to test their UAV's performance under different conditions.

Finally, users can run simulations in Gazebo to test their UAVs' functionality and performance. Users can monitor the UAV's behavior and performance during the simulation using various visualizations and metrics. These metrics can help users identify potential issues or improvement areas before actual deployment.

Gazebo is an essential tool for simulating UAV flight tests in a safe and controlled environment. Its flexibility and ease of use make it a popular choice among researchers and engineers looking to develop and validate UAVs and their control systems.

Software in the loop

In order to do that, Gazebo uses a concept called Software in the Loop (SITL), a simulation technique used to test and validate the software component of a robotic system before deploying it on the actual hardware. In the context of simulating robotic hardware on the Gazebo, SITL involves running the robot's software on a virtual machine that mimics the behavior of the actual hardware.

The SITL approach allows developers to test the software component of the robot in a safe, controlled environment without risking damage to the physical robot or its surroundings. It also provides a cost-effective way to develop and test complex robotic systems.

To implement SITL on Gazebo, the developer creates a virtual environment that emulates the robot's hardware, including sensors and actuators. The software is then executed on this virtual environment, which produces simulated sensor data that the software component processes in real time. The software's output is then fed back to the virtual environment, which simulates the robot's response to the software's commands.

The SITL process can be summarized as follows:

- Create a virtual environment that emulates the robot's hardware.
- Run the robot's software on the virtual environment.
- Simulate the robot's sensors and actuators to produce sensor data and execute commands.
- The software processes the simulated sensor data and produces commands.
- The virtual environment simulates the robot's response to the software's commands.
- The developer can observe the robot's behaviour in the virtual environment and analyze the performance of the software component.

Overall, SITL is a powerful technique for testing and also validating a robotic system's software component. It can help developers identify and fix issues early in the development process.

MAVROS

MAVROS is a middleware framework that provides communication and control interfaces between a computer running the ROS and a drone or other robotic system that is compatible with the MAVLink protocol. MAVLink is a lightweight messaging protocol that is commonly used in the field of UAVs to enable communication between a ground control station and a UAV's flight controller.

Gazebo is a popular open-source robotics simulator that allows users to create and test robotic models in a virtual environment. By integrating MAVROS with Gazebo, developers can simulate the behavior of a drone or other robotic system within a virtual environment, while still being able to interact with the system as if it were physically present.

MAVROS provides a ROS interface to communicate with the drone or robotic system, and Gazebo provides a simulation environment where the system can be tested and validated. This integration allows developers to test and debug their algorithms and control strategies in a simulated environment before deploying them on a physical system.

To use MAVROS with Gazebo, a user would typically create a model of the drone or robotic system in the Gazebo, configure MAVROS to communicate with the simulated system, and then develop their control algorithms using ROS nodes. Once the simulation is running, the user can monitor the behavior of the system and collect data to evaluate the performance of their control algorithms.

In summary, the MAVROS protocol can be used together with Gazebo to simulate the behaviour of a drone or other robotic system within a virtual environment, allowing developers to test and debug their algorithms and control strategies before deploying them on a physical system.

II.5.4 Control interface with python

For the easy testing of parameters in the model, an interface for easy testing was needed. For that, It was built a graphical control interface using MAVROS, *rospy*, and *PySimpleGUI27* libraries. The source code for this program with more information about its usage can be found at https://github.com/piradata/wpg/tree/master#readme.

```
from mavros.utils import *
from mavros_msgs.msg import *
from mavros_msgs.srv import *
from mavros import setpoint as SP
import PySimpleGUI27 as sg
```

More about the MAVROS library can be found at the following links https://masoudir.github.io/mavros_tutorial/ and http://wiki.ros.org/rospy/Overview. These links explain how to install and use the MAVROS and rospy libs respectively. Basically, this program implements two classes, *SetpointPosition* and *SetpointVelocity*, that are responsible to publish set-points for position and velocity respectively on the PX4 control flow using the MAVLink protocol.

In order to do that, we initialize a $waypoint_controller$ node in the default namespace.

```
rospy.init_node('waypoint_controller')
mavros.set_namespace()  # initialize mavros module with default namespace
rate = rospy.Rate(20)
```

And then, it is necessary to wait while the UAV is in the process of arming.

```
rospy.loginfo("arming")
while not state.armed:
    modes.setArm(True)
    rate.sleep()
```

With the rosnode initialized and the drone armed, it is possible to start the UAV control. As previously stated, the UAV can be controlled with both the $SetpointPosition$ and $SetpointVelocity$ classes. The classes have the following methods:

- **init**

  Basically, the variables are initialized to store the current desired drone position on the x,y,z space and the desired yaw angle. The roll and pitch desired positions are assumed to be zero in order to maintain a stable position during hover.

  It also initialized a threading event to know if the drone is already in the desired position. This is calculated at every rate time in a concurrent thread.

  For last, it starts to publish in the desired setpoint topic available on MAVROS and subscribes to the position or velocity topic in order to determine the real position or velocity that the drone has.

  The subscriber topic is attached to a function that runs every time a new message arrives. For the position class, the function used is the 'reached', and for the velocity class. the **velocity_meter** is used.

- **reached** (position) and **velocity_meter** (velocity)

It's a function that runs every time a new message reaches or from the **local_position** or the **velocity_local** topics from MAVROS. Every time a new message arrives, it calculates if the position/velocity is in a close range to the values set on the **init** or **set** methods, and then sets the **done_evt** threading variable in order to warn the main process that the drone is in the desired position.

Every time the set command is used, this variable is cleared so that the robot can go to the new desired position. This, however, only happens if the UAV is ordered to wait for the setpoint to be reached. Suppose it is not ordered to, and that is the default behavior. In that case, the UAV will go to a new location every time a new position is issued, regardless of reaching the setpoint or not (that works well with the real-time control interface that will be explained below).

- **start**

  It activates the threads for position and velocity control.

- **navigate_position** (position) and **control_pid** (velocity)

  These are the threads responsible for position and velocity control. They publish on the setpoint topic the control values in order to move the drone to the desired state defined on the set function.

- **set**

  It defines the current desired position of the drone;

- **finish**

  The opposite of the start, it deactivates the control work status and, with that ceases the work of the **navigate_position** and/or **control_pid** and threads.

There are also three functions that are only used by the $SetpointVelocity$ class and will be better explained below:

- **activate_fuzzy**

  It configures the **control_pid** function to start adjusting the PID values of the outer control loop.

- **deactivate_fuzzy**

  Stops the dynamic adjust of the PID values.

- **calculate_fuzzy**

The software Matlab calculates the variations that must be applied to each topic. This function fetches the values from the topic **defuzzy_values** and updates the internal controller variables with its values.

The software Matlab knows about the values because on custom topic called **fuzzy_values** was created just for this reason, to send to the software Matlab the info about the Error, Derivative of the error, and the current values for the P, I and D variables of the PID controller.

This, however, was only implemented on the X-axis. For it to work with the Y and Z axis, some modifications to the code would still be necessary. For more information about this work, the authors refer to the following published journal papers of our research group LACEA (ANDRADE et al., 2022; MELO et al., 2022; CARVALHO et al., 2021).

In short, both controllers are very similar, but there is a catch, the velocity control only works if the drone is already in the air in the $OFFBOARD$ mode- So, in order to use the velocity control, it is necessary first to start the drone control in the position mode. Then, set it to $OFFBOARD$ and change to the velocity control on midair.

```
rospy.loginfo("## Initiating module of position control")
setpoint_pos.init(0.0, 0.0, 0.0)
setpoint_pos.start()

rospy.loginfo("Set mode to offboard")
modes.setMode("OFFBOARD")

rospy.loginfo("Takeoff")
setpoint_pos.set(0.0, 0.0, INITIAL_ALTITUDE, wait=True)

rospy.loginfo("## finalizing module of position control")
setpoint_pos.finish()

rospy.loginfo("## Initiating module of velocity control")
setpoint_vel.init(0.0, 0.0, INITIAL_ALTITUDE)
setpoint_vel.start()
setpoint_vel.set(0.0, 0.0, INITIAL_ALTITUDE, wait=True)
```

Then, the GUI for the wpg program made with the $PySimpleGUI$ lib starts. Figure II.7 gives an image demonstrating the interface that the user is presented with when the program runs.

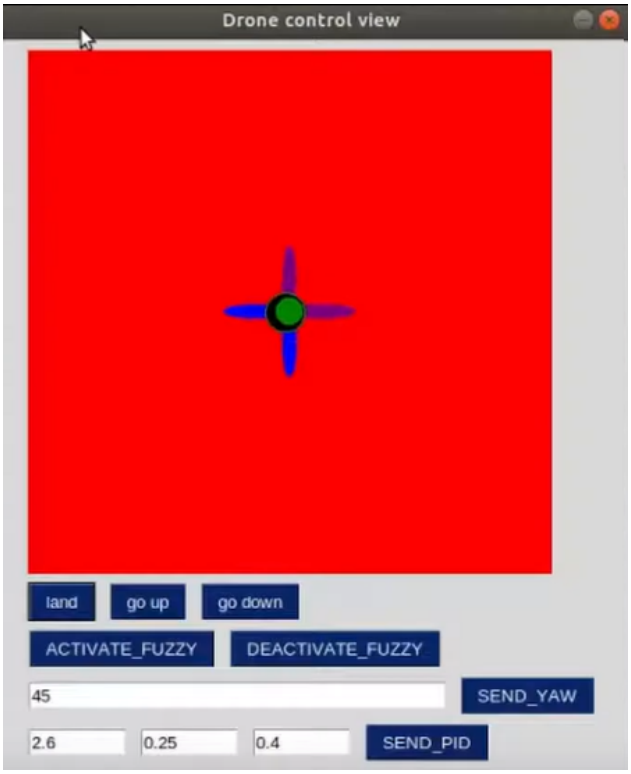


Figure II.7: WPG Software User Interface.

In this video, there is a demonstration of the usage of the program and the program with the optional Matlab/fuzzy integration https://www.youtube.com/watch?v=9c_JVZtPlU8.

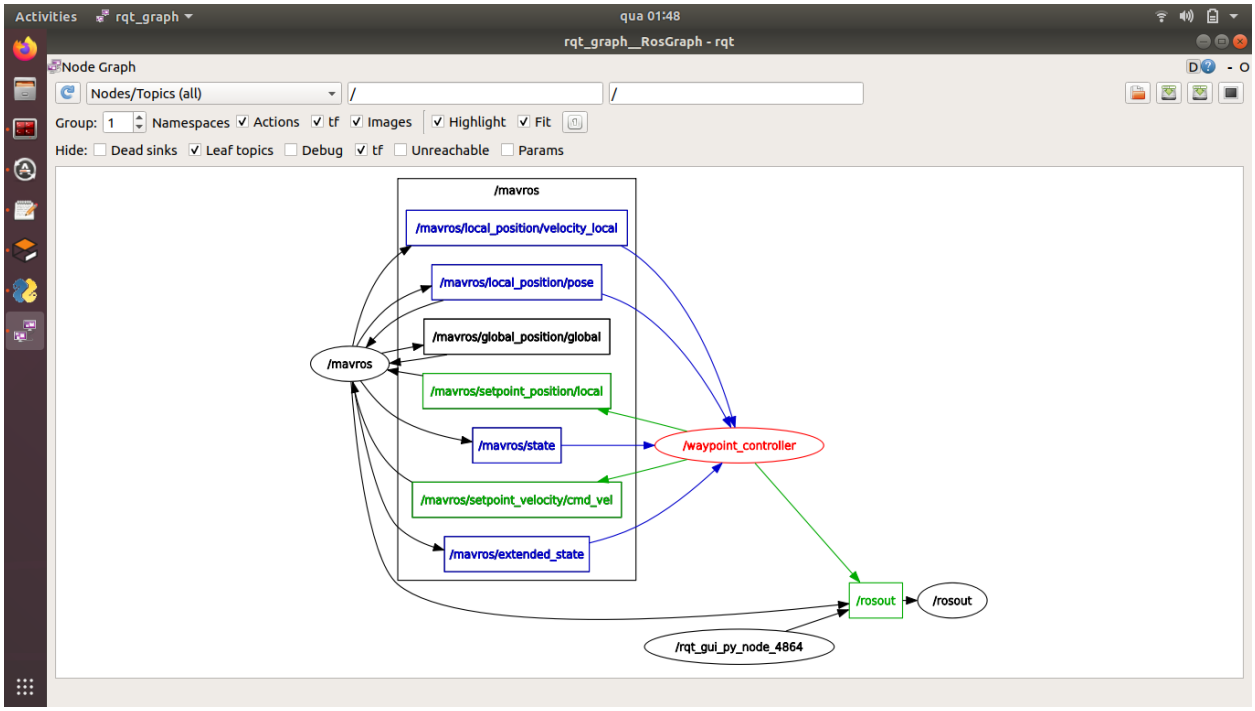


Figure II.8: Ros topics without fuzzy.

II.5.5 Matlab

Plotting

Matlab software was used in this work mainly for plotting reasons. A script was developed to use a BAG file as input and then generate data plots about the stability and the path traveled. More info about this script can be found here https://github.com/piradata/wpg/blob/master/README.md#tools.

Basically, this repository has a Matlab script `matlab_plotter_mavros_v2.m` that plots the mission data based on a BAG file as input. To generate a BAG file that records the simulation data, it is necessary to run the following script in another terminal while the simulation is running:

```
rosbag record -a -O sample_name
```

This will generate a `sample_name.BAG` file on the current directory you are in. Then, it is needed to edit the file `matlab_plotter_mavros_v2.m` or `matlab_plotter_mavros.m` in the root folder of this project and put the bag file there.

```
% Controllers = ["SMC_PID_8", "SMC_PID_8_WW", "PID", "PID_WW"];
Controllers = ["sample_name"];
```

Now, it is possible to run the script. A series of plots will be generated regarding the data of the flight. Below is a fragment of the code in the $matlab_plotter_mavros_v2.m$ on the project's root folder.

```
for i = 1:length(Controllers)
    Controller = Controllers(i);

    BAG{i} = rosbag(sprintf("%s.bag", Controller));

    titulo = sprintf("teste com controlador %s", Controller);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % msgStructs = readMessages(BAG,'DataFormat','struct');

    topic_pos = select(BAG{i}, 'Topic', '/mavros/local_position/pose');
    topic_pos_ref = select(BAG{i}, 'Topic', '/reference_pos');
    topic_vel = select(BAG{i}, 'Topic', '/mavros/local_position/veloc...');
    topic_ref_vel = select(BAG{i}, 'Topic', '/mavros/setpoint_velocit...');
    topic_fuz = select(BAG{i}, 'Topic', '/wpg/fuzzy_values');
    topic_sensor_angacc = select(BAG{i}, 'Topic', '/mavros/local_posi...');
```

Fuzzy Logic

Although the integration with Matlab for fuzzy analysis is not the focus of the current work, it is worth justifying its mention based on the previous text. As mentioned earlier, the initial development of the control strategy for the autonomous UAV involved the study and evaluation of different control methods, including using a Fuzzy controller. The collaboration with Ihannah Pinto Guedes, as documented in her master's thesis (ANDRADE et al., 2022), resulted in the developing of the fuzzy controller alongside the WPG program itself. It is still valid to explain the method as it was developed alongside the WPG program itself.

As said above, the communication between the `matlab` and the control algorithm is not. It is performed through 2 topics called `fuzzy_values` and `defuzzy_values`, which respectively have the custom messages `fuzzy` and `defuzzyfied`. It is possible to see the schema for the `fuzzy` and `defuzzyfied` message types there.

```
std_msgs/Header header %Fuzzy
float64 Error
float64 D_Error
float64 P_val
float64 I_val
float64 D_val
```

```
std_msgs/Header header %Defuzzyfied
float64 Delta_P_val
float64 Delta_I_val
float64 Delta_D_val
```

Figure II.9 presents an image that explains it better. The control node talks directly with Matlab through topics and gets the data from the drone from ROS, while ROS receives this data from MAVROS, which uses the MAVLink protocol to talk with the control firmware on the drone. This can occur in the real world or be simulated in the Gazebo software. The communication with MAVLink can be through a telemetry module or a connection with an address running in the Gazebo software on the computer, respectively.

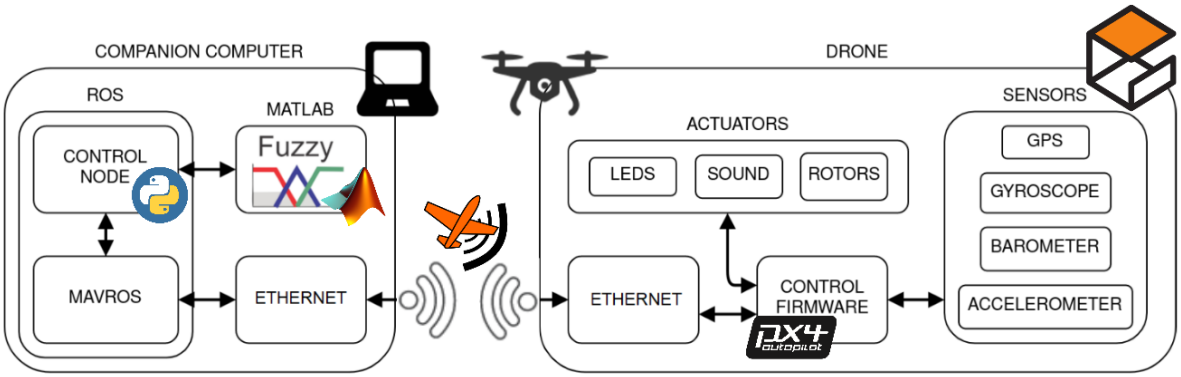


Figure II.9: Matlab fuzzy data flow.

The Python node controls the UAV in real-time by sending the `fuzzy` message data through the topic `fuzzy_values`. Then, the Matlab software processes these values and, based on them, it can be used to decide how much each variable from the PID controller should change.

Matlab then sends back this data already defuzzified in the format of the `defuzzyfied` message data type through the topic `defuzzy_values`. The Python main node is subscribed to this topic on each new message. It updates the values of the P, I, and D control variables. Below is presented the code on the controller that is responsible for receiving the defuzzified value and modifying the parameters.

```
if self.fuzzyfy:
    rospy.loginfo_once("initalized fuzzyfication for the fist time")
    if not self.defuzzed.P_X == 0:
        rospy.loginfo_once("first value of P_X modified by fuzzy system")
        rospy.loginfo_once(f"P_x = {str(self.defuzzed.P_X)}")
    global KPx
    KPx += self.defuzzed.P_X/SOFTNESS
    if KPx <= 0.6: KPx = 0.6
    if KPx >= 5.6: KPx = 5.6
    global KIx
    KIx += self.defuzzed.I_X/SOFTNESS
    if KIx <= 0.3: KIx = 0.3
    if KIx >= 4.5: KIx = 4.5
    global KDx
    KDx += self.defuzzed.D_X/SOFTNESS
    if KDx <= 0.1: KDx = 0.1
    if KDx >= 0.8: KDx = 0.8
```

Figure II.10 gives an overview of how these nodes and topics interact with each other.

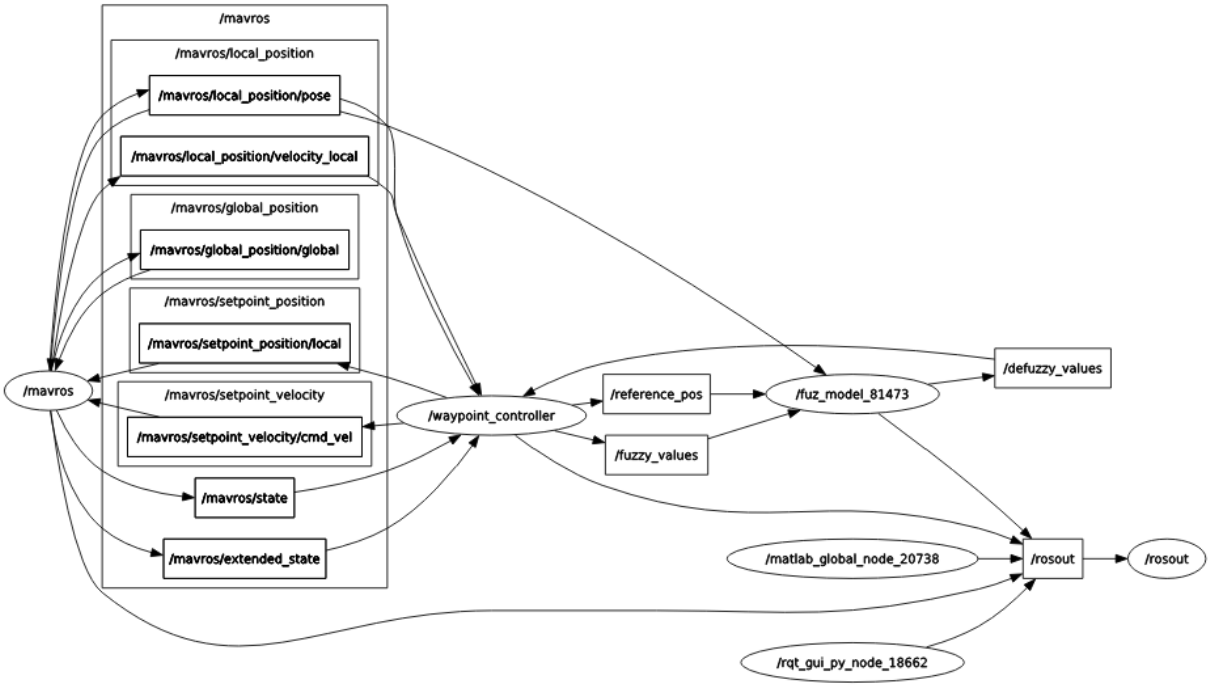


Figure II.10: Ros topics with fuzzy.

It is possible to verify the `waypoint_controller` (Python controller node) node sending messages to the following topics:

- `ref_pos`
- `fuzzy_values`
- `mavros/setpoint_position/local`
- `mavros/setpoint_velocity/cmd_vel`

Meanwhile, it is reading from the topics:

- `defuzzy_values`
- `mavros/state`
- `mavros/extended_state`
- `mavros/local_position/pose`
- `mavros/local_position/velocity_local`

In this specific configuration, the `matlab` represents the node `fuz_model_81473`

Chapter III Proposed Methodology

In the development of the control strategy for the autonomous UAV used in the offshore cargo transport task, the choice of the control approach is fundamental to ensure good performance and operation safety. Different control methods were studied and evaluated, initially focusing on a Fuzzy controller. This preparation involved collaboration with Ihannah Pinto Guedes, also a student from PPEEL/CEFET-RJ, who defended her master's thesis in 2022 based on the development of the fuzzy controller. However, considering the need for robustness in challenging environments like the Oil and Gas industry, the decision was made to implement SMC in the PX4 FCU software for the UAV.

Therefore, this chapter presents the methodology used to implement SMC control in the PX4 FCU software for the UAV and the computer vision technique for the autonomous UAV used in the offshore cargo transport task. Emphasis is placed on the detailed explanation of SMC control, from theory to practical implementation, demonstrating its effectiveness and robustness in autonomous transport operations in challenging environments such as the Oil and Gas industry. The reason to choose the SMC is that this control technique can be applied to nonlinear systems with disturbances and uncertainties. It is based on creating a state space in which the system's dynamic behavior is divided into regions, and the goal is to take the system from one state to another as quickly as possible. To achieve this, a set of control laws are created that ensure the system's stability, even in the presence of disturbances.

III.1 Overall Idea

This research work aims to deliver a small cargo in a marine operation from one vessel platform to another. As stated before, autonomous navigation is derived from previous work from our group research and is available at (RAMOS et al., 2022b). Therefore, this work only focuses on the control strategy to perform the cargo transport in this scenario with stability and safety. For this reason, it was implemented an SMC controller in a PX4 FCU software for the position controller. This controller will be the base for data acquisition and actuator control. It was modified the current PID control strategy as an innovative approach and inserted the SMC logic inside the internal rate control module. Figure III.1 presents the process in a simplified way.

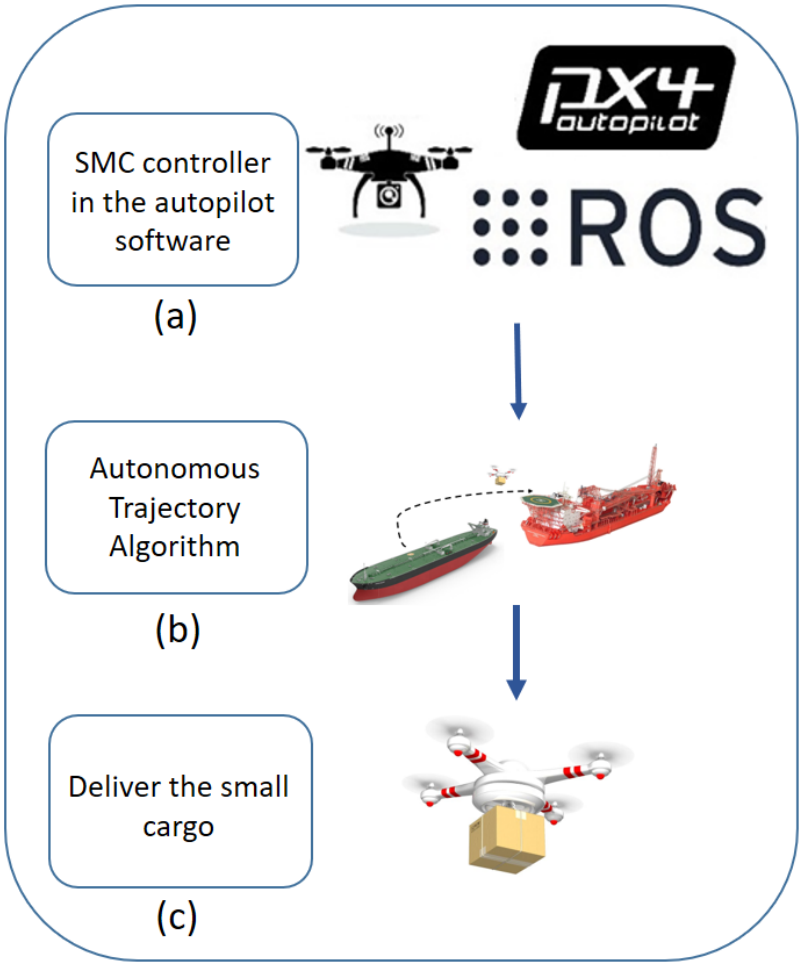


Figure III.1: Overview flowchart.

(A) The SMC controller is implemented to substitute the PID controller in the FCU due to the wind disturbances of the marine operation.

(B) An autonomous trajectory method is used to autonomously guide the UAV to the delivery point in the destination vessel.

(C) When arriving at the delivery point, the UAV delivers the small cargo.

III.2 First attempts to improve PID for external disturbances

The author and former student Ihannah Pinto Guedes from PPEEL/CEFET-RJ, As explained before, developed a strategy to improve the PID controller in the PX4 when facing external disturbances. The proposed first attempt to improve PID when facing disturbances was to use a hybrid scheme combining PID and Fuzzy logic. This approach is based on a cascaded PID with a fuzzy algorithm responsible for calculating the PID gains. This work was published by the author in Carvalho et al. (CARVALHO et al., 2021).

In this first idea, the Fuzzy and the PID will solve the system's nonlinearity, and as a result, the system can accurately converge to the desired position in fewer iterations. Figure III.2 shows the idea of an iterative learning control algorithm and the real-time management of Fuzzy gain computation. Initially, suitable PID values need to be defined. Then, these values are updated dynamically by the Fuzzy algorithm. Note that the rules depend only on the position error and its derivative (CARVALHO et al., 2021).

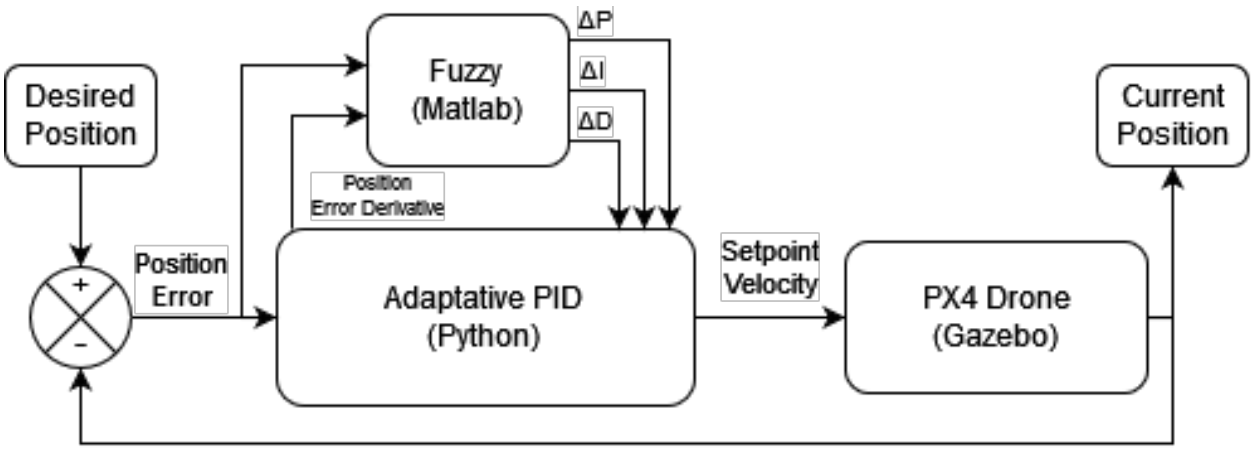


Figure III.2: Scheme of fuzzy addition to the PID controller (CARVALHO et al., 2021).

The choices of the *Membership Functions* were based on the PID gain values that have resulted in a desired closed-loop response. The fuzzy inference technique adopted in this first approach was based on the classical Mamdani inference method (SELVACHANDRAN et al., 2019) and the centroid defuzzification method. Table III.1 shows the fuzzy rule.

Table III.1: Fuzzy rule (CARVALHO et al., 2021).

	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	-	-

Note that the symbol 'B' represents 'Big', 'M' is 'Medium,' and 'L' is 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 (CARVALHO et al., 2021).

It is essential to highlight some general potential challenges when implementing a hybrid PID-fuzzy controller. This approach involves making trade-offs between the advantages and limitations of PID and fuzzy logic control, where PID controllers provide stability and responsiveness but may struggle with uncertain or nonlinear systems. On the other hand, Fuzzy logic controllers excel in handling complex and uncertain systems but can be more challenging to tune and may introduce additional computational overhead.

Besides, in the case of the scenario proposed in this work, the UAV operates with wind disturbances as environmental conditions along with the payload. Therefore, ensuring the robustness of the hybrid PID-fuzzy controller to handle these uncertainties is complex and crucial. In this sense, an alternative control approach that offers several advantages over the hybrid PID-fuzzy controller approach is the SMC controller.

The SMC tends to exhibit less sensitivity to measurement noise than approaches like PID-fuzzy control. The robustness properties of SMC allow it to tolerate measurement noise and disturbances to a certain extent without compromising stability and performance. The following section better explains the implementation and design of the SMC controller in the PX4 FCU.

III.3 SMC Control Design and Implementation

The authors will use the PX4 firmware as a code base to implement and test the SMC control in a quadrotor UAV. The intention is to modify the part of the code responsible for controlling the UAV angular rate, that is, the most internal part of the cascading controller, as shown in Figure II.2.

Also, this work uses the simulation of performance compared with the PID control of the ROS with Gazebo software to simulate a real mission in an open environment. The mission used for testing and comparison was chosen as a cargo delivery between vessels in the Oil and Gas sector.

The idea for this approach is to replace the angular rate controller, which is the lowest level controller in the cascade, by the first order SMC with a relative degree of 2 represented by the Equation III.1 in which $\kappa > 0 \in \Re$ is the modulation constant, $\alpha,\ \beta,\ \gamma > 0 \in \Re$ are design parameters, and $\sigma \in \Re$ defines the sliding surface. In III.2. The $tanh(\cdot)$ function is adopted in the following for smoothness purposes. The objective is to reduce the chattering phenomenon in the control command to be applied to the actuators. Since SMC is applied in the subsystems of Equations (II.1)-(II.2), to represent them in a standardized way, it was defined generalized control and output error signals u and e, respectively, as

$$u = \kappa\ tanh(\sigma) \tag{III.1}$$

$$\sigma = \alpha e + \beta \int e\ dt + \gamma \dot{e} \tag{III.2}$$

In Equations (II.1) and (II.2), the variable σ stands for the generalized sliding surface that is computed for every subsystem.

It is important to mention that, in practice, the control signals generated by Equation (III.2) are transformed to U_2, U_3 and U_4 of Equation (II.2), which are torque commands for roll, pitch and yaw axis, respectively.

This controller is implemented at the top of the PX4 firmware as a base for data acquisition and actuator control. A fork of the official repository was made, and following the provided documentation (DEVS, 2022), it was modified the current cascade PID control and inserted the SMC logic inside the internal rate control module. All the documentation on how to run this simulation yourself and the source code change can be found at piradata-WPG (python module for tests) and piradata-Firmware (Modified PX4 Firmware, mainly file RateControl.cpp on directive RateControl::update). Figure III.3 presents a general idea of this work proposition.

Note that the number of the system state variable describes the sliding surface. The sliding surface is a set of points on a space that a system is desired to be inside. When the system goes away from this region, it is expected that the controller used on the region outside this point makes the system converge to the sliding surface. This generates the chattering effect due to imperfect control

switching (SLOTINE, 1991) around a boundary layer. This chattering can be smoothed using the method of continuous approximation of the switching control law. This work uses a Sigmoid or some other function instead of the signal to treat the non-continuity of the S value.

The sliding surface defines the S value. Basically, its value is zero when the system is on the sliding surface and assumes other values, typically one or minus one, otherwise. When a smoothing function is applied on the boundary layer instead of a sign function or when one uses higher-order sliding mode surfaces, this value can also be found in the region between 1 and -1.

This controller will be implemented at the top of the PX4 firmware as a base for data acquisition and actuator control. A fork of the official repository was made. Following the provided documentation (DEVS, 2022), we modified the current cascade PID control and implemented the SMC logic inside the internal rate control module. Figure III.3 presents a general idea of this work proposition.

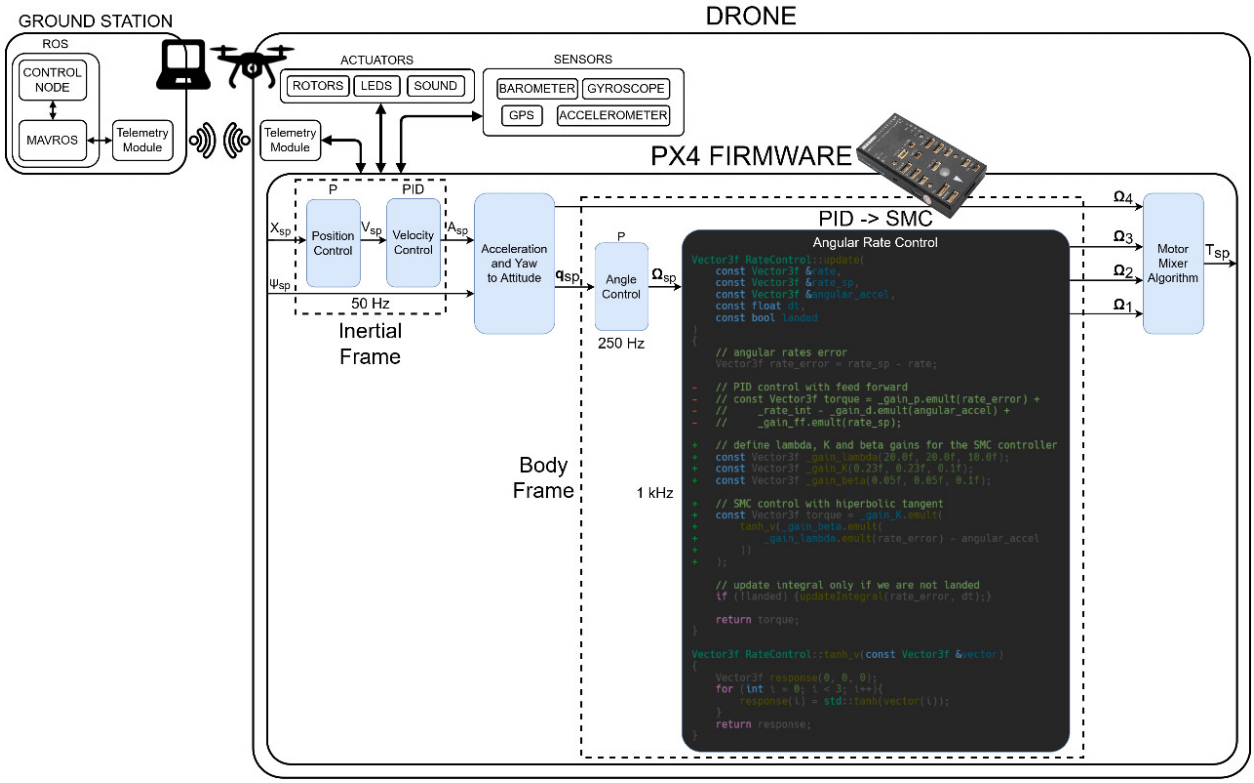


Figure III.3: SMC implemented in the PX4 Firmware.

III.4 Autonomous Trajectory

For the autonomous trajectory, the authors refer to a previous work of our group lab (RAMOS et al., 2022b). In this scheme, the location of the delivery point is configured in the UAV's off-board through GPS coordination. Besides, the operator informs the desired distance between the ship and the vehicle for security purposes. The trajectory calculation starts from reading the coordinates of the takeoff point through GPS, making this the origin of a three-dimensional Cartesian system (x-axis oriented towards the east, y-axis oriented towards magnetic north, and z-axis being altitude). The designed trajectory is presented in Figure III.4. Note that this straight line is to avoid gas escapes points and classified areas around the vessels.

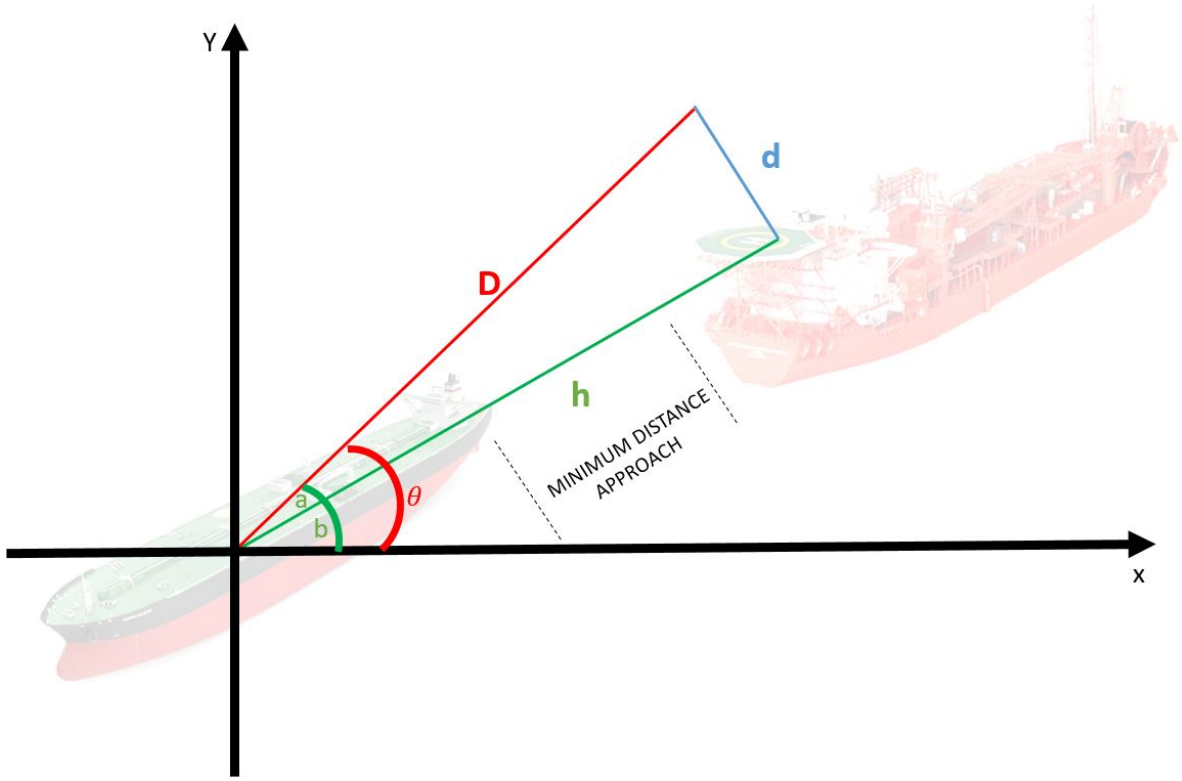


Figure III.4: Linear trajectory.

Given the destination and the takeoff coordinates, it is possible to calculate the distance D between the takeoff point and the final point of the first step of the trajectory, as shown in Equation III.3.

$$\begin{aligned} D &= R \cdot \arccos(\sin(Lat_i) \cdot \sin(Lat_f) \\ &\quad + \cos(Lat_i) \cdot \cos(Lat_f) \cdot \cos(Lon_i - Lon_f)) \end{aligned} \tag{III.3}$$

where the parameters Lat and Lon represent the GPS coordinates (in radians) and the notations "i" and "f" are the takeoff location and destination point. R symbolizes the Earth's radius, admitted

as 6.373×10^6 m. Equation (III.4) gives the destination azimuthal direction θ from the takeoff point.

$$\theta = \arctan \cdot \frac{\Delta_{(Lon)}}{\Delta_{(\phi)}}, \tag{III.4}$$

where

$$\Delta\phi = \log \left[\frac{\tan(\frac{Lat_f}{2} + \frac{\pi}{4})}{\frac{Lat_i}{2} + \frac{\pi}{4}} \right] \tag{III.5}$$

$$\Delta lon = |Lon_i - Lon_f| \tag{III.6}$$

Note that the destination point of the vessel is not where the UAV delivers the cargo but a location distant from the delivery point by a distance d. The control node receives this information and measures the UAV's distance from the target. The trajectory calculation publishes this route to the controller with the points (x, y), the flight height at each point (z), and the robot's orientation (yaw) at each point.

To approach the destination, it is implemented a hybrid methodology in which the controller detects the vessel and its generic regions (bow, mid-ship, and stern) by a convolutional neural network for object detection and fuses it with position and attitude data using an Extended Kalman Filter (EKF) to navigate towards the delivery point, anticipating any drifting caused by wind or yaw controller error while maintaining a safe approaching speed to the destiny, as detailed in Ramos et al. (RAMOS et al., 2022b).

Chapter IV Results and Discussion

IV.1 Simulation Environment

In order to evaluate the proposed strategy safely, this work used the Gazebo software to simulate the test environment. The ROS interface with UAV works as follows. The computer receives the information from a telemetry module available on both the computer and the UAV, using the MAVLink communication protocol. The PX4 software, which runs on the PixHawk flight controller hardware, is responsible for acquiring information from the peripheral sensors and modifying these values in the actuators to which it has access. The GCS runs the operating system Ubuntu 20.04.5 LTS 64 bits with an Intel®Core^{TM} i7-5500U CPU 2.40GHz x 4, Intel®HD Graphics 5500 (Broadwell GT2) with 15.6 GB RAM.

The UAV used in the experiments (3DR iris model) and the world created to simulate the cargo transport between vessels are shown in Figures IV.1 and IV.2, respectively.



Figure IV.1: Quadrotor used in the Gazebo software.

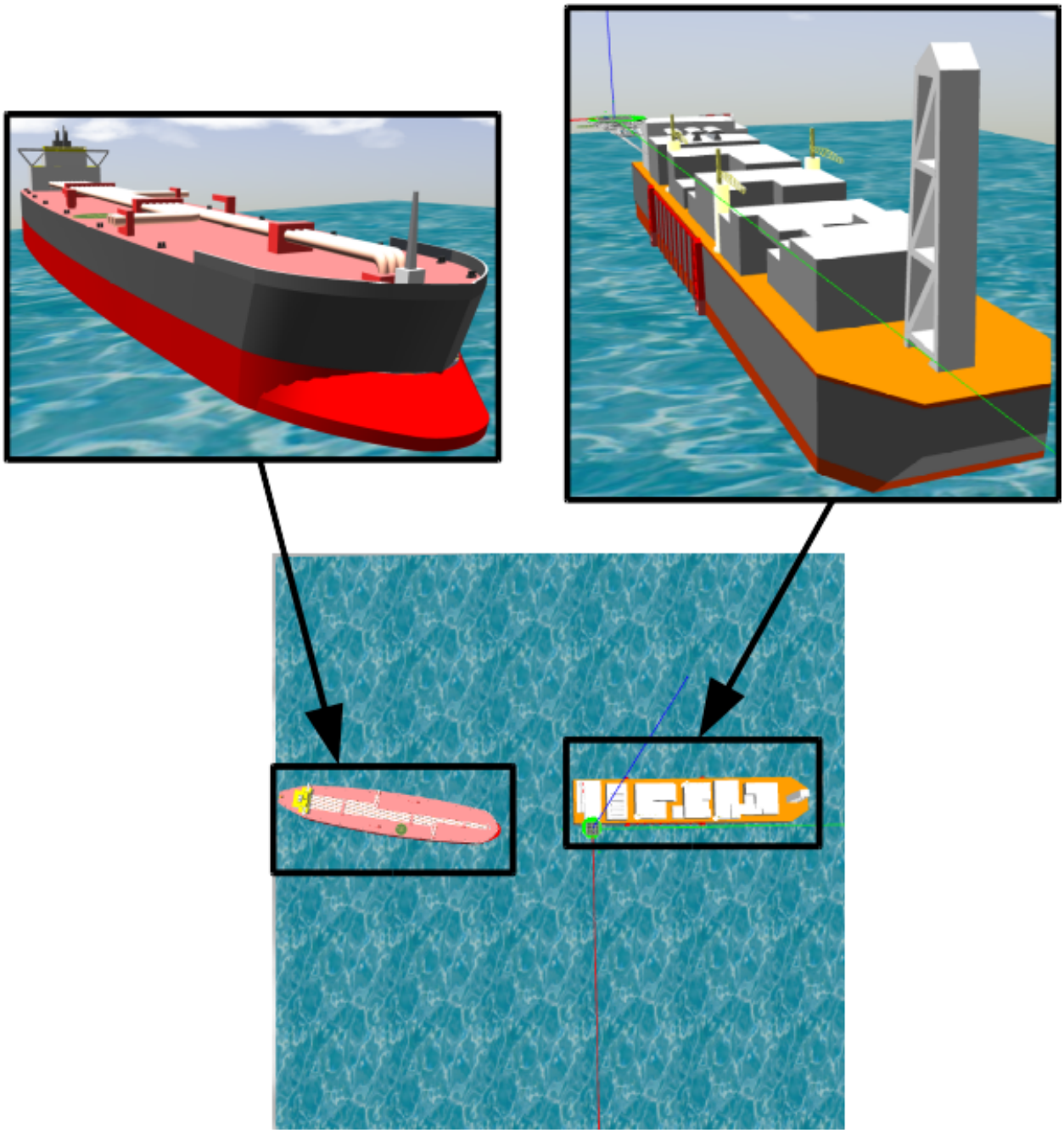


Figure IV.2: World created in the Gazebo software.

After carrying out the code modifications on the PX4, is important to compare the UAV performance by using the original PID controller and the modified one with SMC. The parameters, chosen empirically, for the SMC controller (Equation III.1) used in these tests can be found in Table IV.2. The default PID gains used in the tests, for comparison purposes, are shown in Table IV.1.

Table IV.1: PID parameters

Parameter	ϕ	θ	ψ
P	0.15	0.15	0.20
I	0.20	0.20	0.10
D	0.03	0.03	0.00

Table IV.2: SMC parameters

Parameter	ϕ	θ	ψ
κ	0.26	0.26	0.08
α	1	1	1.20
β	0.2	0.2	0.10
γ	0.006	0.006	0.00

The tests were conducted in three stages, as shown in Figure IV.3. In the first stage, the UAV takes off from the original vessel. Then, in the second stage, it approaches the destination vessel by using the autonomous trajectory algorithm. Finally, it starts to land when it reaches a limited distance, which is the end of the third stage.

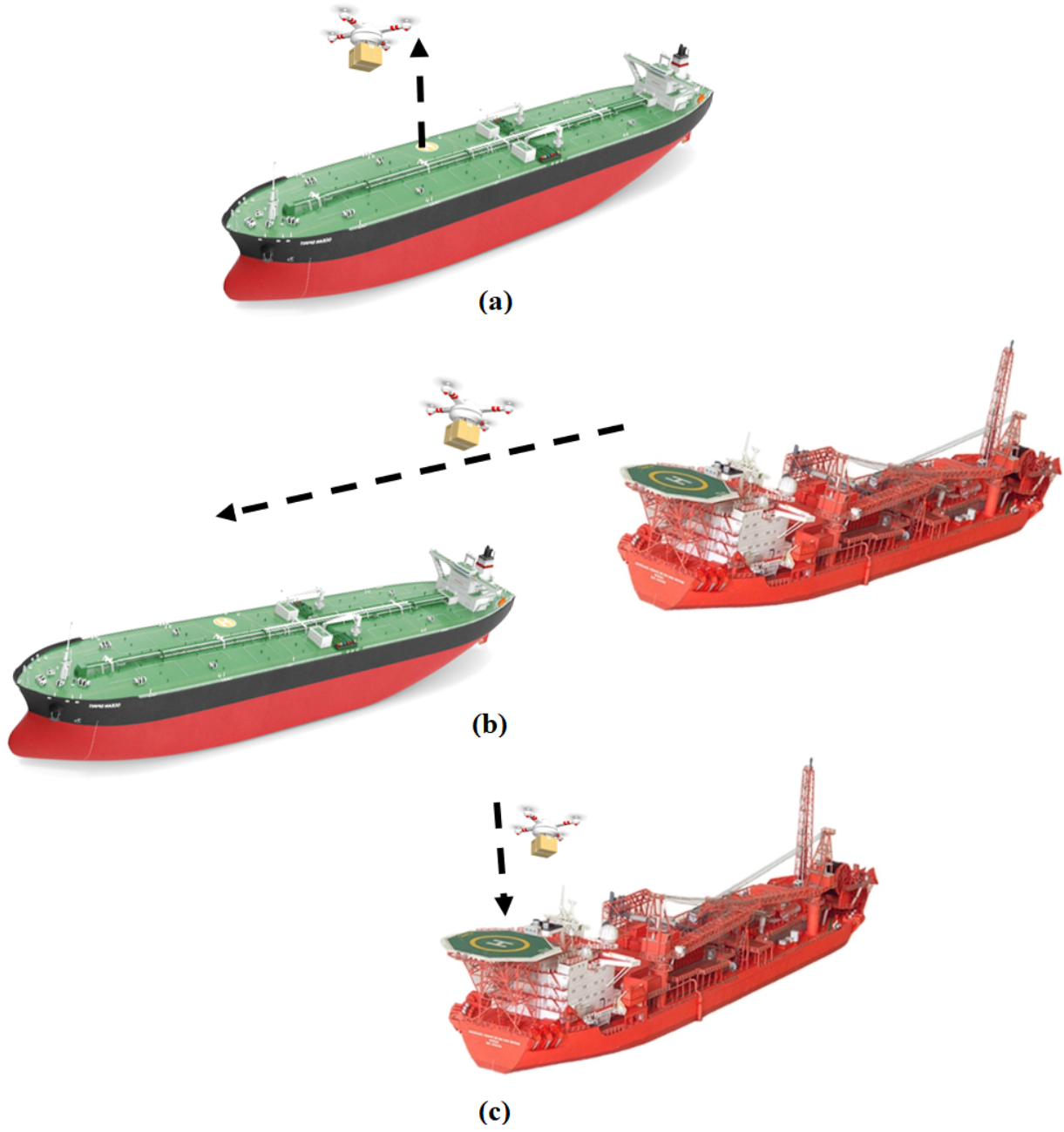


Figure IV.3: Tests conducted in three stages: (a) Take off. (b) Approaching vessel. (c) Landing.

These stages were performed using the proposed SMC as well as the original PID algorithms on the internal rate control module of the PX4. State estimation is obtained from the FCU data. Note that the proposed SMC solution and the classical PID control were both applied to the system for comparison purposes both in the absence and in the presence of disturbances (i.e., wind).

Below in table IV.3 we can see all the parameters tested on gazebo while tuning the SMC with an empirical approach.

In the end, the best ones were the tests 7, 8, 10 and 18, but 7 and 10 have exactly the same parameter values and 18 was excluded for having a high jump on the setpoint switch.

After better analyzing in matlab the flight graphics generated from ros .bag files, we found that despite test 7 and 8 having similar travel times, test 8 had a better response to the wind disturbances in the simulation.

Table IV.3: Tuning parameters tested in gazebo simulator

	SMC_{xyz}				PID_{xyz}				Name	Comments
	Kappa	Alpha	Beta	Gamma	P	I	D	ff		
roll	-	-	-	-	0,15	0,2	0,003	0		
pitch	-	-	-	-	0,15	0,2	0,003	0	PID_1	Normal drone PID without modifications
yaw	-	-	-	-	0,2	0,1	0	0		
roll	0,23	1	-	0.05	-	-	-	-		
pitch	0,23	1	-	0.05	-	-	-	-	SMC_PD_1	SMC with stable PD surface
yaw	0,1	2	-	0,1	-	-	-	-		
roll	0,23	1	0,2	0.05	-	-	-	-		
pitch	0,23	1	0,2	0.05	-	-	-	-	SMC_PID_1	SMC with PID surface, there was no improvement significant in relation to the PD surface
yaw	0,1	2	0,1	0,1	-	-	-	-		
roll	0,25	1	0,2	0,005	-	-	-	-		
pitch	0,25	1	0,2	0,005	-	-	-	-	SMC_PID_2	Changing variables showed a significant improvement
yaw	0,12	2	0,1	0	-	-	-	-		
roll	0,26	0,3	0,2	0,003	-	-	-	-		
pitch	0,26	0,3	0,2	0,003	-	-	-	-	SMC_PID_3	It was extremely unstable compared to the previous one
yaw	0,12	0,2	0,1	0	-	-	-	-		
roll	0,3	0,5	0,2	0,006	-	-	-	-		
pitch	0,3	0,5	0,2	0,006	-	-	-	-	SMC_PID_4	It's hard to say if there's been an improvement or a deterioration without looking at the graphs.
yaw	0,1	1	0,1	0	-	-	-	-		
roll	0,28	1	0,2	0,006	-	-	-	-		
pitch	0,28	1	0,2	0,006	-	-	-	-	SMC_PID_5	Possibly the best result so far, I will try to reduce slightly kappa gain for Z and increase gamma for XY
yaw	0,1	1,3	0,1	0	-	-	-	-		
roll	0,28	1	0,2	0,008	-	-	-	-		
pitch	0,28	1	0,2	0,008	-	-	-	-	SMC_PID_6	Seems to have gotten a little worse...
yaw	0,08	1,2	0,1	0	-	-	-	-		
roll	0,26	1	0,2	0,006	-	-	-	-		Was much faster this time, almost as much as the
pitch	0,26	1	0,2	0,006	-	-	-	-	SMC_PID_7	PID, but it's unknown how much of that is the influenced
yaw	0,08	1,3	0,1	0	-	-	-	-		by the wind behaving different in each simulation
roll	0,26	1	0,2	0,006	-	-	-	-		Almost no change from the previous one.
pitch	0,26	1	0,2	0,006	-	-	-	-	SMC_PID_8	Does it get worse by increasing the derivative again?
yaw	0,08	1,2	0,1	0	-	-	-	-		
roll	0,26	1	0,2	0,008	-	-	-	-		The derivative possibly has to be very low
pitch	0,26	1	0,2	0,008	-	-	-	-	SMC_PID_9	anyway, because the test was very unsatisfactory
yaw	0,08	1,2	0,1	0	-	-	-	-		
roll	0,26	1	0,2	0,006	-	-	-	-		Ran the PID_7 test again which had been the best so far.
pitch	0,26	1	0,2	0,006	-	-	-	-	SMC_PID_10	It ended at about the same time as benchmark 7, so the
yaw	0,08	1,3	0,1	0	-	-	-	-		results at least seem reproducible and not mere chance
roll	0,26	1	0,2	0,005	-	-	-	-		Without significant improvement
pitch	0,26	1	0,2	0,005	-	-	-	-	SMC_PID_11	It was even a little worse.
yaw	0,08	1,3	0,1	0	-	-	-	-		
roll	0,25	1	0,2	0,006	-	-	-	-		
pitch	0,25	1	0,2	0,006	-	-	-	-	SMC_PID_12	It got worse, it even presented instability during the flight
yaw	0,07	1,1	0,1	0	-	-	-	-		
roll	0,28	1,2	0,2	0,0055	-	-	-	-		
pitch	0,28	1,2	0,2	0,0055	-	-	-	-	SMC_PID_13	Improved compared to 11 but still worse than 10 or 7
yaw	0,07	0,8	0,1	0	-	-	-	-		
roll	0,27	1,2	0,2	0,0055	-	-	-	-		The flight was stable, but the runtime deteriorated again.
pitch	0,27	1,2	0,2	0,0055	-	-	-	-	SMC_PID_14	I'll try next time to increase only the alpha's z to 1
yaw	0,06	0,7	0,1	0	-	-	-	-		
roll	0,27	1,2	0,2	0,0055	-	-	-	-		A little better than 13, but similar performance.
pitch	0,27	1,2	0,2	0,0055	-	-	-	-	SMC_PID_15	I will try to lower the alpha and lower the kappa
yaw	0,06	1	0,1	0	-	-	-	-		
roll	0,265	1,25	0,2	0,0055	-	-	-	-		
pitch	0,265	1,25	0,2	0,0055	-	-	-	-	SMC_PID_16	It was worse than the previous one
yaw	0,055	1,05	0,1	0	-	-	-	-		
roll	0,24	1	0,2	0,006	-	-	-	-		Relating to 10: Changed x and y from kappa
pitch	0,24	1	0,2	0,006	-	-	-	-	SMC_PID_17	and z from gamma.
yaw	0,08	1,3	0,1	0,6	-	-	-	-		Got worse
roll	0,24	1	0,2	0,006	-	-	-	-		Relative to 10: Changed only x and y from kappa.
pitch	0,24	1	0,2	0,006	-	-	-	-	SMC_PID_18	It performed similarly to 10, so either it didn't
yaw	0,08	1,3	0,1	0	-	-	-	-		difference in changing this parameter or was it very little
roll	0,26	1,1	0,2	0,006	-	-	-	-		Relative to 10: Changed x and y from alpha
pitch	0,26	1,1	0,2	0,006	-	-	-	-	SMC_PID_19	It got a lot worse
yaw	0,08	1,3	0,1	0	-	-	-	-		
roll	0,26	1	0,2	0,006	-	-	-	-		Relating to 10: Changed z from kappa.
pitch	0,26	1	0,2	0,006	-	-	-	-	SMC_PID_20	Very bad
yaw	0,06	1,3	0,1	0	-	-	-	-		
roll	0,26	1	0,2	0,006	-	-	-	-		Relating to 10: Changed z from alpha.
pitch	0,26	1	0,2	0,006	-	-	-	-	SMC_PID_21	Again bad
yaw	0,08	1,4	0,1	0	-	-	-	-		

IV.1.1 Simulation results considering the absence of external disturbances

Regarding the proposed scenario of marine small cargo transport, which comprises take off, moving along a trajectory, and then landing, the first set of results presents a comparative performance analysis between the classical PID controller and the SMC strategy for position control (Figure IV.4) and derivative error (Figure IV.5) along the X, Y, and Z axis of all three stages.

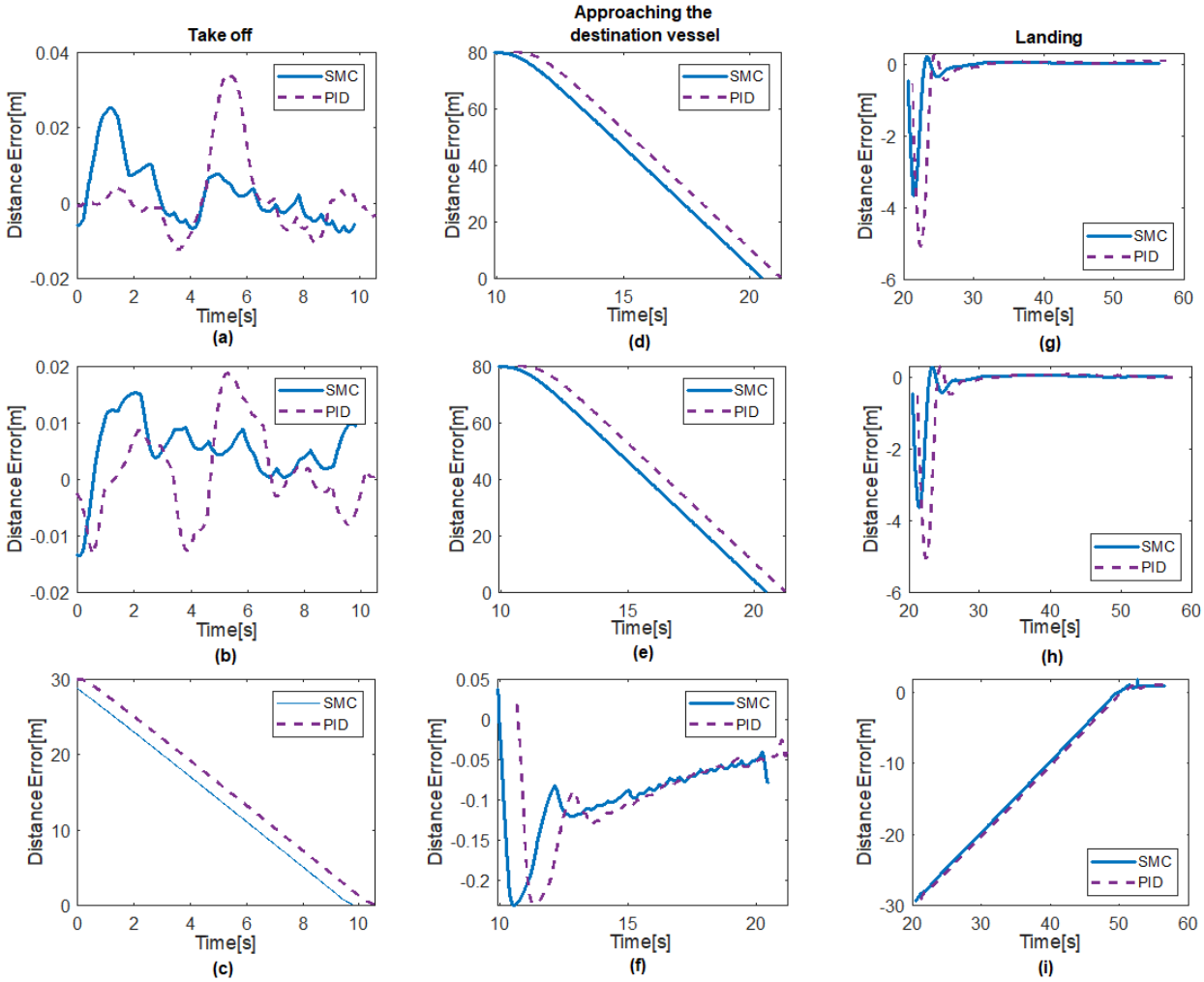


Figure IV.4: Position error without disturbances for X-axis (a,d,g), Y-axis (b,e,h) and Z-axis (c,f,i)

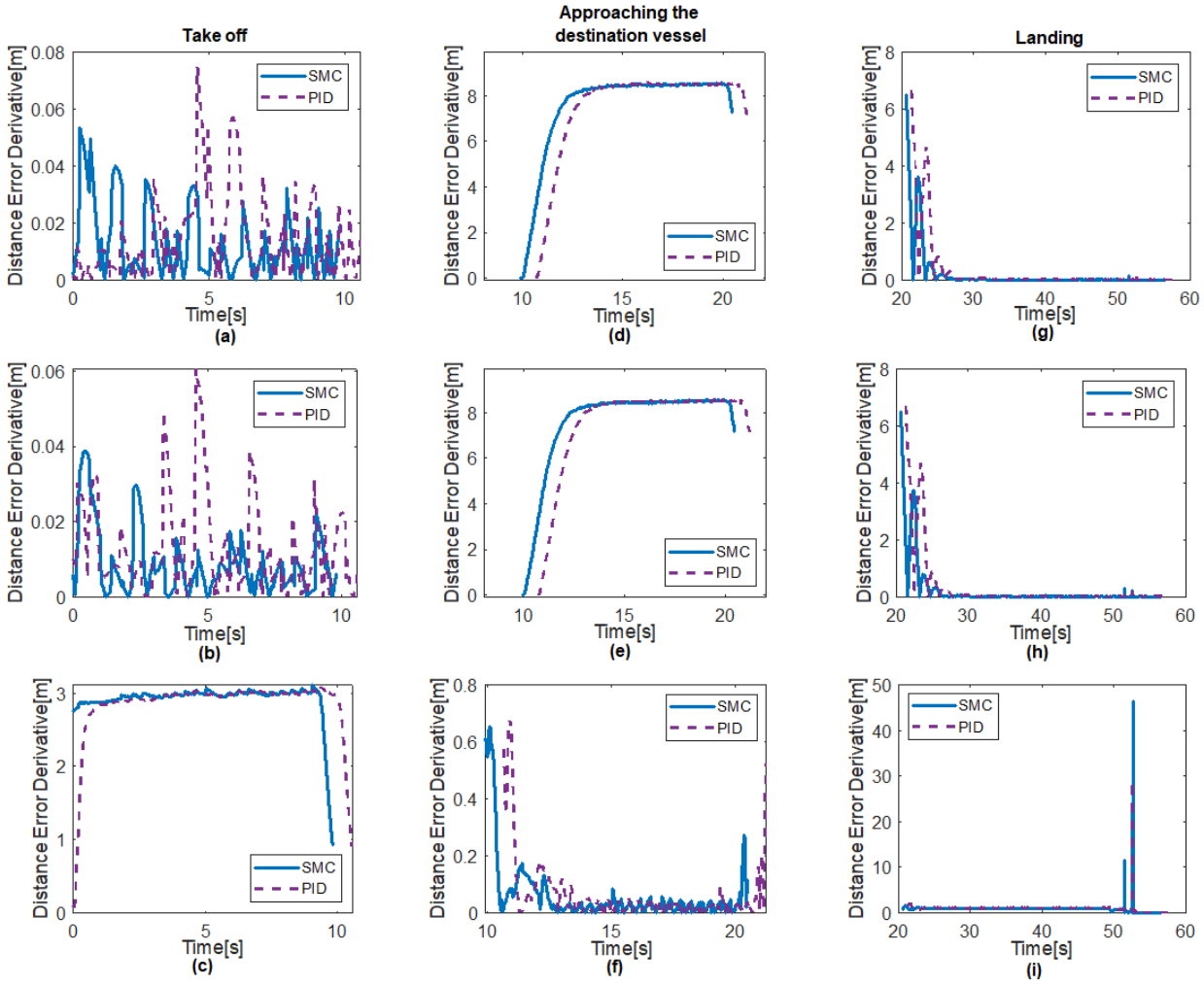


Figure IV.5: Derivative error without disturbances for X-axis (a,d,g), Y-axis (b,e,h) and Z-axis (c,f,i)

The figure IV.6 gives us the outcome for the entire 3D path performed by both controllers without considering wind in the scenario of going from one ship to another.

As expected, when achieving desired maneuvers, the SMC provides a satisfactory performance very similar to the one obtained with the PID. That is, both controllers were capable of tracking the desired position with negligible steady-state errors, demonstrating fewer performance differences in the case of the absence of external disturbances.

The results for the positioning error in the X, Y, and Z axis of the three stares are shown, respectively, in Figure IV.4. This is the same test case as the former results in Figure IV.6, in which there are no external disturbances such as wind current and/or variable mass cargo.

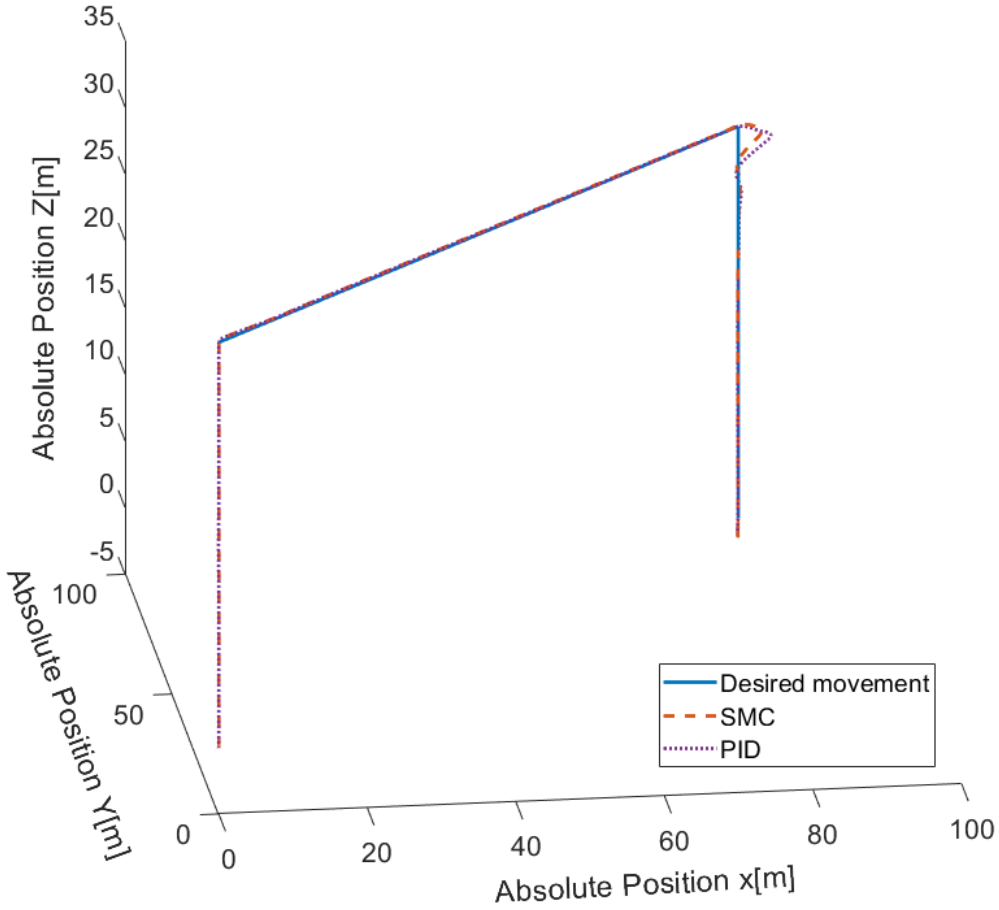


Figure IV.6: Position tracking comparing SMC and PID controllers without disturbances

Note that on takeoff and landing tasks, the results in the Z axis are not comparable as they seem to be almost identical, and the same occurs on the X and Y axis while moving between ships. That happens because these axes are the ones subjected to the largest changes in the desired position along the step.

It is possible to observe that the proposed SMC strategy has similar behaviour to that of the classical PID available for the PX4 firmware, under these conditions. However, some disturbances are compensated better by using the SMC controller. Small overshoots on the X and Y axis are observed in the landing phase right after they reach their targets when moving from one ship to another.

IV.1.2 Simulation results considering the presence of external disturbances

Figure IV.7 shows the entire 3D path for the position reference tracking with the purpose of observing both controllers' behaviour when dealing with external disturbances. To demonstrate the effectiveness and robustness of the SMC controller, this simulation considers the same wind current in both simulation trials. Note that the SMC performed better than the classical PID in the sense of tracking the reference inputs.

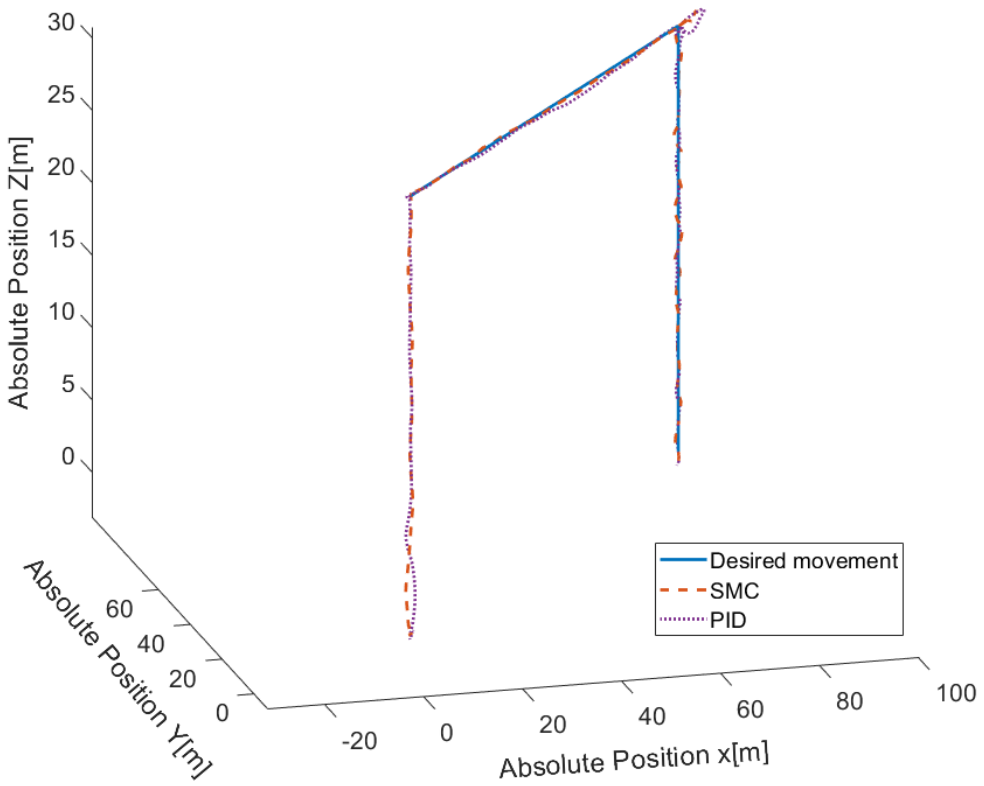


Figure IV.7: Position tracking comparing SMC and PID controllers with disturbances

Figure IV.8 gives the position errors of both controllers along the X, Y, and Z axis, for all three stages of the proposed scenario, respectively. Note that the output obtained with SMC has less oscillatory behaviour around the reference value than that obtained with the PID. The SMC also presented even better performance under disturbance circumstances in terms of the convergence velocity of the tracking error. Thus, this kind of result can help to understand the importance of the implementation of this robust controller into the PX4 hardware.

In the initial takeoff step, there is a gust of wind that the SMC counteracts better. There is also an upward gust on the movement step that almost does not interfere with the quadcopter when the rate controller is under the SMC algorithm.

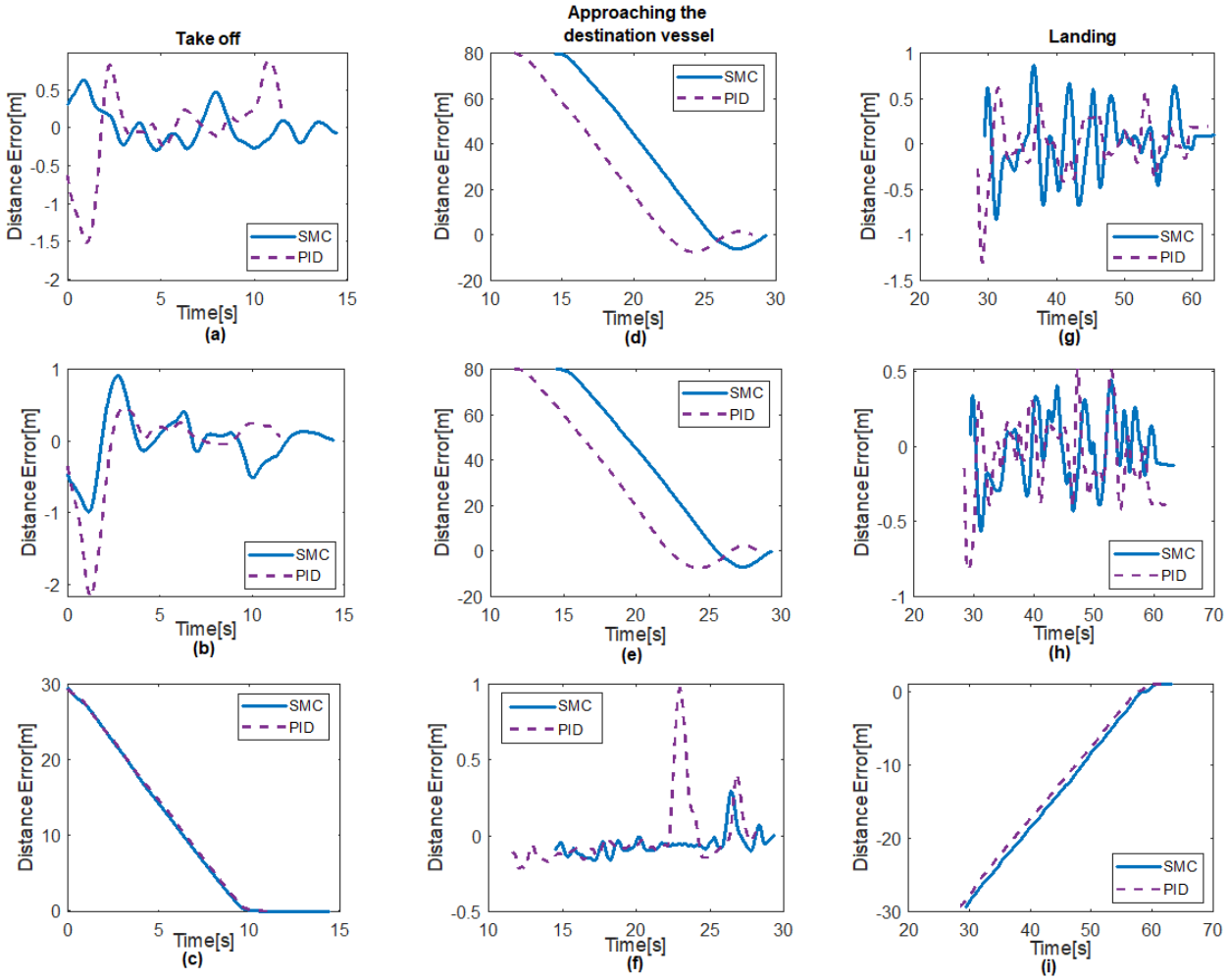


Figure IV.8: Position error with disturbances for X-axis (a, d, g), Y-axis (b, e, h), and Z-axis (c, f, i).

Figure IV.9 gives the position errors of both controllers along the X, Y, and Z axis, for all three stages of the proposed scenario.

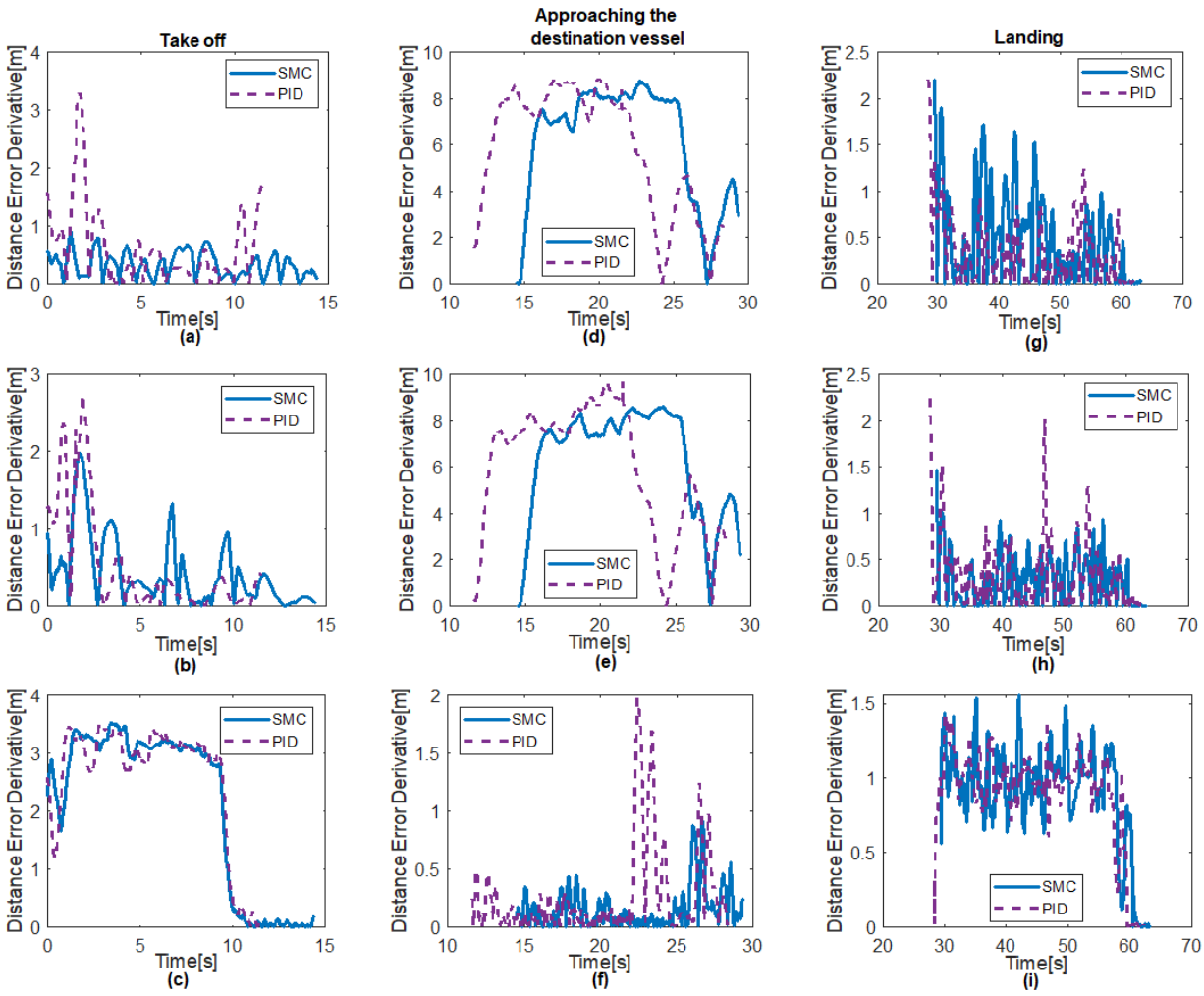


Figure IV.9: Derivative error with disturbances for X-axis (a, d, g), Y-axis (b, e, h), and Z-axis (c, f, i).

One point that is valid to mention is that the path controller algorithm of the WPG package considers that the UAV arrives at the destination if the distance from the current point to the desired point is less than 0.1 meters in all directions. Basically, inside a cube (side of 20 cm) centred on the desired point. And only when that happens does the WPG package send the drone to the next desired destination. So, the algorithm that does the path in less time in an environment with high disturbances is not necessarily the better, as one could reach by chance the "checkpoint area" faster.

In order to define the best controller in this situation, we need to observe the Maximum Average Error (MAE) that both present. The PID has an MAE of 0.35 meters, and the SMC controller with the PID surface has an MAE of 0.16 meters

Therefore, the trajectory tracking performance demonstrated the effectiveness of the SMC solution embedded in a PX4 FCU against the adverse condition, as expected from theoretical analysis.

IV.2 Autonomous Trajectory Results

These simulations were conducted under the same environment, one using the classic PX4 PID controller and one using the implemented SMC. The objective was to verify if the implemented SMC responds better when the UAV is carrying cargo subjected to offshore environmental conditions. Figure IV.10 presents the simulation results.

When compared to the ideal trajectory, the PID has an MAE of 0.35 meters, and the SMC controller with the PID surface has an MAE of 0.16 meters. As presented before, the initial trajectory is performed with the UAV following geospatial waypoints, so in this trajectory, both controllers execute almost the same course.

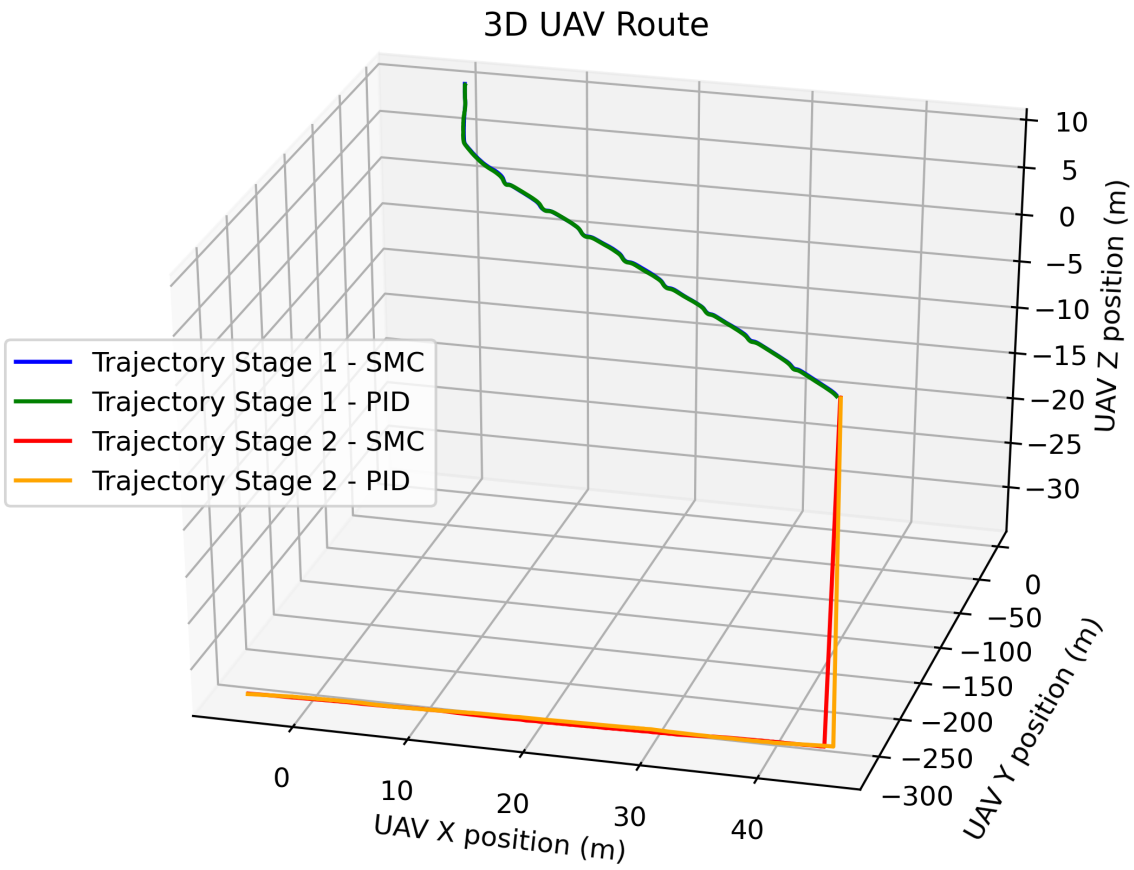


Figure IV.10: Comparison of the SMC and PID controllers in autonomous trajectory simulation

The major difference was in stage 2, where the UAV navigates from one ship to the other. In this stage, the SMC has fewer errors than PID. The SMC also provides a quicker operation than the PID, finding the marker within 2 minutes 48 seconds, while the PID took 3 minutes 34 seconds, probably by being more stable and therefore the camera input for the OpenCV in smoother. Therefore, the implemented SMC controller is an advanced option for flight robustness for operations in harsh environments like the one proposed in this work.

Chapter V Conclusions and Future Work

V.1 Final Conclusions

The PID controller has been widely used in literature and industry due to its simple structure and less dependence on the model. However, this controller adjustment can be difficult in practical terms. Regarding UAVs, external disturbances, such as wind, actuator failure, and parametric uncertainties, can cause physical instability of the UAVs platform. UAVs commonly use a classical PID controller. A drawback is the necessity of expertise from the operator to adjust the parameters to deal with nonlinearities.

An SMC is a robust control technique that can control different types of systems, including UAVs such as those using the PX4 flight control software. In this sense, this work developed and implemented an SMC controller in a PX4 FCU software for quadrotor UAV to apply an autonomous UAV in an offshore cargo task for the Oil and Gas sector. Simulation tests were conducted in the Gazebo/ROS software with a realistic simulation environment to verify the performances of the methodology.

The results demonstrated the robustness of the SMC implemented in the PX4 FCU software. However, it comes with a price, as the SMC controller has a bigger energetic cost with the actuator (rotor) being requested to change operation at a higher frequency. As an advantage of an SMC controller, the UAV can adapt to different disturbances, such as wind gusts, turbulence, and unexpected obstacles, allowing it to navigate through challenging environments with excellent stability and accuracy.

It is important to highlight that the implementation of the SMC in a PX4 system performed in this work has the potential to bring significant benefits to the UAV control and navigation field. SMC is a robust control technique that can handle uncertainties and disturbances in real-time, making it well-suited for use in UAVs operating in complex and challenging environments. In this sense, this work represents a significant step forward in the development of robust and efficient control systems for UAVs, which are becoming increasingly important in a wide range of industries and applications.

V.2 Future Ideas

The implementation of a slide mode controller in a PX4 system opens up many possibilities for future research and development in the field of UAV control and navigation. Therefore, in terms of evaluation, this research opens up several possibilities to develop sophisticated path planning capable of working in global and local environments. In future works, the authors intend to perform the same test executed in the Gazebo software simulation on the real world.

The quadrotor will move between ships carrying cargo transport. It is intended to implement failures in the system to test the controller. The SMC is fault-tolerant and can handle various types of system failures. By incorporating SMC into the PX4 control system, it is possible to improve the overall system's reliability and safety, which is crucial for UAV applications. Another possible future work is the application of SMC in PX4 for object tracking. SMC can be used to track and follow objects in real-time, even in the presence of uncertainties and disturbances. This could be useful in various applications such as search and rescue, inspection, and surveillance.

Bibliography

ABRO, G. E. M. et al. Model-free-based single-dimension fuzzy smc design for underactuated quadrotor uav. In: MULTIDISCIPLINARY DIGITAL PUBLISHING INSTITUTE. *Actuators*. [S.l.], 2021. v. 10, p. 191. Citado 2 vezes nas páginas 22 e 23.

AHMED, N.; CHEN, M. Sliding mode control for quadrotor with disturbance observer. *Advances in Mechanical Engineering*, SAGE Publications Sage UK: London, England, v. 10, n. 7, p. 1687814018782330, 2018. Citado na página 12.

AHMED, S. F. et al. Attitude stabilization of quad-rotor (uav) system using fuzzy pid controller (an experimental test). In: IEEE. *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*. [S.l.], 2015. p. 99–104. Citado 2 vezes nas páginas 16 e 18.

ALATTAS, K. A. et al. Barrier function adaptive nonsingular terminal sliding mode control approach for quad-rotor unmanned aerial vehicles. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 22, n. 3, p. 909, 2022. Citado 2 vezes nas páginas 22 e 24.

ALMAKHLES, D. J. Robust backstepping sliding mode control for a quadrotor trajectory tracking application. *IEEE Access*, IEEE, v. 8, p. 5515–5525, 2019. Citado na página 12.

ANDRADE, F. A. A. et al. Unmanned aerial vehicles motion control with fuzzy tuning of cascaded-pid gains. *Machines*, v. 10, n. 1, 2022. ISSN 2075-1702. Disponível em: <"https://www.mdpi.com/2075-1702/10/1/12">. Citado 5 vezes nas páginas 11, 12, 14, 34 e 37.

ANSARI, U.; BAJODAH, A. H. Robust generalized dynamic inversion quadrotor control. *IFAC-PapersOnLine*, Elsevier, v. 50, n. 1, p. 8181–8188, 2017. Citado na página 12.

ASADZADEH, S.; OLIVEIRA, W. J. de; FILHO, C. R. de S. Uav-based remote sensing for the petroleum industry and environmental monitoring: State-of-the-art and perspectives. *Journal of Petroleum Science and Engineering*, Elsevier, v. 208, p. 109633, 2022. Citado na página 11.

BENALLEGUE, A.; MOKHTARI, A.; FRIDMAN, L. High-order sliding-mode observer for a quadrotor uav. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, Wiley Online Library, v. 18, n. 4-5, p. 427–440, 2008. Citado na página 12.

BIUNDINI, I. Z. et al. A framework for coverage path planning optimization based on point cloud for structural inspection. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 2, p. 570, 2021. Citado na página 11.

CAI, W. et al. Disturbance suppression for quadrotor uav using sliding-mode-observer-based equivalent-input-disturbance approach. *ISA transactions*, Elsevier, v. 92, p. 286–297, 2019. Citado na página 12.

CARVALHO, G. et al. Hybrid pid-fuzzy controller for autonomous uav stabilization. In: IEEE. *2021 14th IEEE International Conference on Industry Applications (INDUSCON)*. [S.l.], 2021. p. 1296–1302. Citado 9 vezes nas páginas viii, ix, 12, 14, 18, 20, 34, 42 e 43.

CHEN, F. et al. Robust backstepping sliding-mode control and observer-based fault estimation for a quadrotor uav. *IEEE Transactions on Industrial Electronics*, IEEE, v. 63, n. 8, p. 5044–5056, 2016. Citado 2 vezes nas páginas 22 e 23.

DAI, Y. et al. Uav-assisted data offloading for smart container in offshore maritime communications. *China Communications*, IEEE, v. 19, n. 1, p. 153–165, 2022. Citado na página 11.

DAS, A.; LEWIS, F.; SUBBARAO, K. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, Springer, v. 56, n. 1, p. 127–151, 2009. Citado na página 12.

DAS, A.; SUBBARAO, K.; LEWIS, F. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET control theory & applications*, IET, v. 3, n. 3, p. 303–314, 2009. Citado na página 12.

DEVS, P. *An In-depth Look at the Multicopter Control System Architecture*. 2020. https://px4developersummitvirtual2020.sched.com/event/cjOm/an-in-depth-look-at-the-multicopter-control-system-architecture. Acessado em 02/03/2022. Citado 3 vezes nas páginas 13, 18 e 29.

DEVS, P. *PX4 systems architecture documentation*. 2022. https://docs.px4.io/master/en/concept/px4_systems_architecture.html. Acessado em 02/04/2022. Citado 2 vezes nas páginas 44 e 45.

DOITSIDIS, L. et al. A framework for fuzzy logic based uav navigation and control. In: IEEE. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. [S.l.], 2004. v. 4, p. 4041–4046. Citado na página 20.

DONG, J.; HE, B. Novel fuzzy pid-type iterative learning control for quadrotor uav. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 1, p. 24, 2019. Citado na página 20.

DURDEVIC, P. et al. Vision aided navigation of a quad-rotor for autonomous wind-farm inspection. *IFAC-PapersOnLine*, v. 52, n. 8, p. 61–66, 2019. ISSN 2405-8963. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019. Disponível em: <"https://www.sciencedirect.com/science/article/pii/S2405896319303696">. Citado na página 16.

ELTAG, K.; ASLAMX, M. S.; ULLAH, R. Dynamic stability enhancement using fuzzy pid control technology for power system. *International Journal of Control, Automation and Systems*, Springer, v. 17, n. 1, p. 234–242, 2019. Citado 2 vezes nas páginas 12 e 22.

ENGINEERS, S. Society of P. *Tugboat Company Takes Its Business Flying With Drones*. 2018. Accessed: 2020-06-15. Disponível em: <"https://jpt.spe.org/tugboat-company-takes-its-business-flying-drones">. Citado na página 16.

FAUST, A. et al. Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, Elsevier, v. 247, p. 381–398, 2017. Citado na página 17.

GUAN, H. et al. Uav-lidar aids automatic intelligent powerline inspection. *International Journal of Electrical Power & Energy Systems*, Elsevier, v. 130, p. 106987, 2021. Citado na página 11.

GUPTA, A. et al. Advances of uavs toward future transportation: The state-of-the-art, challenges, and opportunities. *Future Transportation*, v. 1, n. 2, p. 326–350, 2021. ISSN 2673-7590. Disponível em: <"https://www.mdpi.com/2673-7590/1/2/19">. Citado na página 16.

HLAS, M.; STRAUB, J. Autonomous navigation and control of unmanned aerial systems in the national airspace. In: IEEE. *2016 IEEE Aerospace Conference*. [S.l.], 2016. p. 1–7. Citado na página 16.

HU, X.; LIU, J. Research on uav balance control based on expert-fuzzy adaptive pid. In: IEEE. *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. [S.l.], 2020. p. 787–789. Citado 2 vezes nas páginas 12 e 22.

HUO, X.; HUO, M.; KARIMI, H. R. Attitude stabilization control of a quadrotor uav by using backstepping approach. *Mathematical Problems in Engineering*, Hindawi, v. 2014, 2014. Citado na página 12.

JOYO, M. K. et al. Altitude and horizontal motion control of quadrotor uav in the presence of air turbulence. In: IEEE. *2013 IEEE Conference on Systems, Process & Control (ICSPC)*. [S.l.], 2013. p. 16–20. Citado 2 vezes nas páginas 12 e 21.

KELLERMANN, R.; BIEHLE, T.; FISCHER, L. Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, v. 4, p. 100088, 2020. ISSN 2590-1982. Disponível em: <"https://www.sciencedirect.com/science/article/pii-/S2590198219300879">. Citado na página 16.

LEE, D.; KIM, H. J.; SASTRY, S. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of control, Automation and systems*, Springer, v. 7, n. 3, p. 419–428, 2009. Citado na página 12.

LEE, S. J.; JANG, I.; KIM, H. J. Fail-safe flight of a fully-actuated quadrotor in a single motor failure. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 4, p. 6403–6410, 2020. Citado na página 22.

LIERET, M. et al. Automated in-house transportation of small load carriers with autonomous unmanned aerial vehicles. In: IEEE. *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. [S.l.], 2019. p. 1010–1015. Citado na página 17.

Liu, D. et al. Robust fault-tolerant formation control for tail-sitters in aggressive flight mode transitions. *IEEE Transactions on Industrial Informatics*, v. 16, n. 1, p. 299–308, 2020. Citado na página 22.

LIU, Y. et al. Control-oriented uav highly feasible trajectory planning: A deep learning method. *Aerospace Science and Technology*, Elsevier, p. 106435, 2020. Citado na página 16.

MADRIDANO, Á. et al. 3d trajectory planning method for uavs swarm in building emergencies. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 3, p. 642, 2020. Citado na página 11.

MAES, W. H.; STEPPE, K. Perspectives for remote sensing with unmanned aerial vehicles in precision agriculture. *Trends in plant science*, Elsevier, v. 24, n. 2, p. 152–164, 2019. Citado na página 11.

MARTINS, L.; CARDEIRA, C.; OLIVEIRA, P. Feedback linearization with zero dynamics stabilization for quadrotor control. *Journal of Intelligent & Robotic Systems*, Springer, v. 101, n. 1, p. 1–17, 2021. Citado na página 12.

MELO, A. G. et al. Fuzzy gain-scheduling pid for uav position and altitude controllers. *Sensors*, MDPI, v. 22, n. 6, p. 2173, 2022. Citado 2 vezes nas páginas 14 e 34.

MELO, A. G. et al. 3d correspondence and point projection method for structures deformation analysis. *IEEE Access*, IEEE, v. 8, p. 177823–177836, 2020. Citado na página 11.

MELO, A. G. et al. Dynamic optimization and heuristics based online coverage path planning in 3d environment for uavs. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 4, p. 1108, 2021. Citado na página 11.

MOBAYEN, S. Design of cnf-based nonlinear integral sliding surface for matched uncertain linear systems with multiple state-delays. *Nonlinear Dynamics*, Springer, v. 77, n. 3, p. 1047–1054, 2014. Citado na página 12.

MOFID, O.; MOBAYEN, S. Adaptive sliding mode control for finite-time stability of quad-rotor uavs with parametric uncertainties. *ISA transactions*, Elsevier, v. 72, p. 1–14, 2018. Citado 3 vezes nas páginas 12, 22 e 23.

MURRAY, C. C.; RAJ, R. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 110, p. 368–398, 2020. Citado na página 11.

NAJM, A. A.; IBRAHEEM, I. K. Nonlinear pid controller design for a 6-dof uav quadrotor system. *Engineering Science and Technology, an International Journal*, Elsevier, v. 22, n. 4, p. 1087–1097, 2019. Citado na página 20.

NGUYEN, N. P. et al. Adaptive sliding mode control for attitude and altitude system of a quadcopter uav via neural network. *IEEE Access*, IEEE, v. 9, p. 40076–40085, 2021. Citado na página 12.

NGUYEN, P. H. et al. Lightdenseyolo: A fast and accurate marker tracker for autonomous uav landing by visible light camera sensor on drone. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 6, p. 1703, 2018. Citado na página 11.

PETILLOT, Y. R. et al. Underwater robots: From remotely operated vehicles to intervention-autonomous underwater vehicles. *IEEE Robotics & Automation Magazine*, IEEE, v. 26, n. 2, p. 94–101, 2019. Citado na página 16.

PINTO, M. F. et al. Ekf design for online trajectory prediction of a moving object detected onboard of a uav. In: IEEE. *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*. [S.l.], 2018. p. 407–412. Citado na página 11.

PINTO, M. F. et al. Arcog: An aerial robotics cognitive architecture. *Robotica*, Cambridge University Press, p. 1–20, 2020. Citado 3 vezes nas páginas 11, 16 e 18.

PINTO, M. F. et al. A robotic cognitive architecture for slope and dam inspections. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 16, p. 4579, 2020. Citado 3 vezes nas páginas 11, 16 e 18.

PINTO, M. F. et al. Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle. In: *XXXX*. [S.l.: s.n.], 2017. p. 1324–1329. Citado na página 19.

PINTO, M. F. et al. Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle. In: IEEE. *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. [S.l.], 2017. p. 1324–1329. Citado na página 11.

QINGQING, L. et al. Towards active vision with uavs in marine search and rescue: Analyzing human detection at variable altitudes. In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. [S.l.: s.n.], 2020. p. 65–70. Citado na página 16.

RAMOS, G. et al. Technical and economic feasibility study for implementing a novel mooring-assisting methodology in offloading operations using autonomous unmanned aerial vehicles. *SPE Production & Operations*, OnePetro, v. 37, n. 01, p. 72–87, 2022. Citado na página 11.

RAMOS, G. S. et al. Hybrid methodology based on computational vision and sensor fusion for assisting autonomous uav on offshore messenger cable transfer operation. *Robotica*, Cambridge University Press, p. 1–29, 2022. Citado 7 vezes nas páginas 11, 13, 16, 17, 41, 46 e 47.

RÍOS, H.; GONZÁLEZ-SIERRA, J.; DZUL, A. Robust tracking output-control for a quad-rotor: A continuous sliding-mode approach. *Journal of the Franklin Institute*, Elsevier, v. 354, n. 15, p. 6672–6691, 2017. Citado na página 12.

SCALEA, J. R. et al. Successful implementation of unmanned aircraft use for delivery of a human organ for transplantation. *Annals of surgery*, U.S. National Library of Medicine, 2021. Disponível em: <"https://pubmed.ncbi.nlm.nih.gov/31663974/">. Citado na página 16.

Selecký, M. et al. Framework for incremental development of complex unmanned aircraft systems. In: *2015 Integrated Communication, Navigation and Surveillance Conference (ICNS)*. [S.l.: s.n.], 2015. p. J3–1–J3–9. Citado na página 19.

SELVACHANDRAN, G. et al. A new design of mamdani complex fuzzy inference system for multi-attribute decision making problems. *IEEE Transactions on Fuzzy Systems*, IEEE, 2019. Citado na página 42.

SLOTINE, W. L. J.-J. E. *Applied Nonlinear Control*. [S.l.]: Prentice-Hall, 1991. Citado 3 vezes nas páginas 24, 25 e 45.

STAVINOHA, S. et al. Challenges of robotics and automation in offshore oil&gas industry. In: IEEE. *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*. [S.l.], 2014. p. 557–562. Citado 2 vezes nas páginas 11 e 16.

TAN, Y. et al. Automatic inspection data collection of building surface based on bim and uav. *Automation in Construction*, Elsevier, v. 131, p. 103881, 2021. Citado na página 11.

TANVEER, M. H. et al. Nmpc-pid based control structure design for avoiding uncertainties in attitude and altitude tracking control of quad-rotor (uav). In: IEEE. *2014 IEEE 10th International Colloquium on Signal Processing and its Applications*. [S.l.], 2014. p. 117–122. Citado na página 21.

THANH, H. L. N. N.; HONG, S. K. Quadcopter robust adaptive second order sliding mode control based on pid sliding surface. *IEEE Access*, IEEE, v. 6, p. 66850–66860, 2018. Citado na página 12.

WANASINGHE, T. R. et al. Unmanned aerial systems for the oil and gas industry: Overview, applications, and challenges. *IEEE access*, IEEE, v. 8, p. 166980–166997, 2020. Citado na página 11.

WANG, B.; ZHANG, Y. Adaptive sliding mode fault-tolerant control for an unmanned aerial vehicle. *Unmanned Systems*, v. 05, n. 04, p. 209–221, 2017. Disponível em: <"https://doi.org/10.1142-/S2301385017400052">. Citado na página 21.

WANG, Y. et al. Optimization of air–fuel ratio control of fuel-powered uav engine using adaptive fuzzy-pid. *Journal of the Franklin Institute*, Elsevier, v. 355, n. 17, p. 8554–8575, 2018. Citado na página 20.

WEN, F. et al. Unmanned aerial vehicles for survey of marine and offshore structures: a classification organization's viewpoint and experience. In: ONEPETRO. *Offshore Technology Conference*. [S.l.], 2018. Citado na página 11.

XU, R.; OZGUNER, U. Sliding mode control of a quadrotor helicopter. In: IEEE. *Proceedings of the 45th IEEE Conference on Decision and Control*. [S.l.], 2006. p. 4957–4962. Citado na página 12.

Yuan, X. et al. Robust h control for hovering of a quadrotor uav with slung load. In: *2019 12th Asian Control Conference (ASCC)*. [S.l.: s.n.], 2019. p. 114–119. Citado na página 22.