POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Thesis

Design and Testing of Indoor UAS Control Techniques

POLITECNICO
DI TORINO

Advisor
Prof. Elisa Capello

Co-advisor:
Dr. Matteo Scanavino

Student
Davide Carminati

Academic year 2018 – 2019

Abstract

Unmanned Aerial Vehicles (UAVs) represent an ideal platform for testing advanced control techniques. In recent years, many researches based on modern control theories and design of UAV autopilot algorithms have been completed. Most of the modern autopilots incorporate controller algorithms to meet the always more demanding requirements of flight maneuvers and mission accomplishment. However, due to the complexity and the computational need of the control algorithms, most of the commercial autopilots are based on Proportional Derivative Integrative (PID) philosophy, in which trial and error methods are used for the definition of the control gains.

In this thesis, a PID controller system is proposed and it is compared with a variable structure method (that is a Sliding Mode Controller (SMC)). The controllers are designed and implemented for quadrotor indoor applications, facing the main challenges of this particular application. The overall structure of the proposed controllers is composed by two loops: (i) an outer loop related to the slow dynamics of the UAV, i.e. position, and (ii) an inner loop related to the fast dynamics of the system, i.e. attitude and angular velocities. Firstly, a nonlinear mathematical model of the plant is derived, starting from the well-known equations of motions for an aircraft. Then, the two controllers are designed and tested in a suitable simulation environment reproducing as close as possible a real-world situation so that testing can lead to a meaningful outcome. In this regard, a model of the sensors, an Extended Kalman Filter (EKF) and a Trajectory Planner are included in the simulation environment, which are the main elements of a quadrotor. Different types of trajectories are tested, starting from a hover maneuver and waypoints-following paths. Test results show better performance using the SMC, which is able to follow the proposed trajectories accurately and satisfying the given time constraints, while the PID exhibits larger position errors due to a slower inner loop resulting in slower response times in attitude variations. Furthermore, inertial parameters are varied within a limited range to mimic the presence of different payloads. The quadrotor autopilot is based on the Pixhawk 2.1 Cube and it is able of flying indoor thanks to an infrared-based motion capture sensor (the Otus Tracker), replacing the GPS as it is highly unreliable for indoor applications. Future work includes Software-In-The-Loop testing using code generation supported by the "Embedded coder support package for PX4 autopilot" in Matlab, and eventually code deployment on the target hardware to carry out Processor-In-The-Loop testing.

Acknowledgements

I would like thank all the people who supported and followed me. In particular I would like to thank Dr. Elisa Capello and Dr. Matteo Scanavino as they followed me like no-one else would have done during my thesis work. I also want to thank those people who shared their knowledge on particular issues I encountered during my work: Prof. Fabrizio Dabbene, Fabio from Mavtech srl and my friend Pippo.
Furthermore, I want to thank my friends of the immortal group of Gondor, the friends I made during the Bachelor at the Politecnico of Milan, the people of the brianza and lecchese that are my inevitable drinking and jamming buddies and the friends I had the luck to know and to jam with in Turin. Marta helped me a lot during the time I passed here in Turin as well. Last but not least my family, in particular my parents and my sister who guided and truly supported me in my choices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An Unmanned Aerial Vehicle is an aircraft able to fly without a crew. It uses rapidly spinning propellers to push the air downwards and to perform maneuvers. There are several types of chassis available [6] with four, six, eight or more propellers. One of the most used ones is the *quadrotor*, with four equally spaced propellers, paired two-by-two, linked to a central body by means of arms in which sensors and flight computers are accommodated. Quadrotors – and in general multirotors – represent a complex and challenging machine when it has to be controlled. Due to the way actuators are arranged, quadrotors are severely under-actuated. This means that, in order to move in the three-dimensional space in all 6 Degrees of Freedom (DoF), rotational and translational motions have to be coupled, leading to a highly *nonlinear* dynamics. Furthermore, the absence of any kind of friction coming from outside the quadrotor system except for the air itself forces the quadrotor to continuously correct its position and attitude; otherwise it would drift away.
Nonetheless, a multirotor features various advantages:

- HOVERING AND MANEUVERABILITY: a multirotor do not need constant motion to fly, unlike a fixed-wing solution, and it can make sharper turns.
- VERTICAL TAKE-OFF AND LANDING: thanks to the particular motor placement, a multirotor can take off and land without the need of moving in the horizontal plane.
- CHEAP AND MECHANICALLY SIMPLE: a multirotor have a limited number of moving parts, the rest of its structure is basically a rigid chassis.

On the other hand, it has some critical drawbacks:

- LIMITED BATTERY LIFE: power consumption is generally high; usually the battery can last up to $20min$.
- DIFFICULT TO BE CONTROLLED: as stated previously, the multirotor is an *under-actuated* system with a *nonlinear* dynamics.

- LOW PAYLOAD CAPABILITY: multirotors are not able to lift large payloads, which greatly degrade battery life.

In recent times, multirotors acquired more and more attention, due to the lowering of the electronic components prices and the simultaneous increasing of computation performance, and it became widely used in both military and civil applications. UAVs can be employed in scenarios precluded to human beings – because lethal or dangerous environments are involved, or costs cannot be handled. For this reason they are exploited for exploration, search and rescue missions, monitoring and diagnosing, delivery services – both commercial and medical, but even for hobby purposes such as aerial photography. Nowadays, small drones can be found off-the-shelf in every electronic store. In academics and research, the quadrotor is an ideal platform for testing advance control techniques.
This work is focused on a particular application: UAVs for GPS-denied environments. Thanks to their hover capability and agility, multirotors can easily navigate through narrow spaces. This widened the array of tasks a small UAVs can fulfill. They can be used in purely industrial applications, e.g. inside warehouses to monitor and inspect shelves and/or packages, or for emergency situations, such as exploration of damaged and precarious buildings, e.g. after an earthquake or a fire, or even for natural exploration, e.g. inside inaccessible caves. The most important challenge for what concerns indoor usage is the multirotor localization. When the GPS coverage is not ensured due to thick obstacles, basic on-board sensors cannot provide an accurate and reliable position. For this reason, specific additional sensors must be mounted on the UAV. According to their complexity, position sensors can even provide obstacle avoidance.

State of the art

In literature, quadrotors are studied thoroughly in each of their aspects. This work is focused on the control part. A great variety of controllers have been designed and implemented in quadrotors, both linear and nonlinear, and comparisons have been carried out. Although the quadrotor dynamics is *nonlinear*, several linear controllers have been exploited due to their simplicity. [21] designs and simulates a classical PID controller tuned using the well-known Ziegler-Nichols method in order to stabilize the quadrotor attitude; in [8] a similar PID is also deployed on hardware and tested using a suitable test bench constraining the translational Degrees-of-Freedom. With this method, he showed that a complete PID controller is needed for real world applications as the steady-state error is not negligible. Optimized methods based on a linear quadrotor dynamics are exploited, showing good results: in [7] an optimized PID is used for improving disturbance rejection; or LQ-based methods such as in [8] where two methods of solving the Riccati equation are employed, or also [19] which design two LQR controllers managed through a Gain Scheduling (GS) logic, the first acting when the quadrotor is far from the reference

trajectory and the second when tracking. In general, linear methods can achieve stability of the system around the chosen equilibrium point, but they are not able to consider all the system behaviors due to simplifications in their design process: as a result they tend to be slower and less reactive than nonlinear methods.
On the other hand, *nonlinear* controllers have been employed for the same purpose. GS controllers are exploited to take into account the quadrotor working condition – to monitor the hardware and tackle possible faults as in [20] – and to improve overall performance according to the type and current state of the maneuvers [19]. Back-stepping based methods [17] have been designed too and when implemented proved to achieve better performance compared their linear counterpart. Lastly, more complex methods such as feedback linearization and sliding mode control [10, 14] have been successfully implemented. For example, [22] performs a feedback linearization on the attitude dynamics of a quadrotor and drives it with a simpler LQR, showing promising results for this configuration. In [27], the system is decomposed into a nested structure taking into consideration both inner and outer loop, controlling both position and attitude of the UAV by means of the feedback linearization technique. As for Sliding Mode Controllers, it is always characterized by a faster response and less oscillations compared to classical approaches, but blemished by high-frequency oscillations, called *chattering*. This phenomenon, due to its negative impact, is addressed by many authors in literature, by finding a suitable control law that restrain the oscillations or relying on Higher-Order Sliding Mode Controllers (HOSMC), in which the discontinuity in the command law is moved into higher order derivatives [11]. Usually authors when dealing with control design, do not focus on an existing quadrotor, relying on the well-known dynamic equations of the quadrotor. Nevertheless, others focused their work on a particular commercial or customized to some degree quadrotors, taking care of the deployment of the control algorithm onto the real-world prototype [28, 13]. In particular, [13] adopts the same autopilot framework of this work, designing a PID controller for a commercial quadrotor and eventually deploying it on the Pixhawk 1 Autopilot running the PX4 Flight Stack.
Nonetheless, multirotors can be customized at any level. Various universities and research centers focused on different aspects of the single or the group of multirotors [15], from the UAV shape and the control strategy, to the used algorithms – involving Machine Learning or image recognition algorithms – and software architectures, e.g. Cloud Computing [16]. Cloud computing addresses a critical problem in the management of one or more agents: it allows to move algorithms which need high computational resources from the on-board computer to a high-performance computer using a high-speed low-latency connection. These powerful computers usually are in charge to execute heavy computer vision algorithms or to coordinate the swarm. These last algorithms use *network theory* and *multi-agent systems theory* to organize and govern the swarm. Furthermore, some authors introduced

innovation on the multirotor shape, so that the drone can reconfigure itself according to the scenario completely changing its shape, as in [12] where a foldable drone reconfigures the position of the propellers to access narrow spaces, or relying on additional Degrees of Freedom, as in [18], where tilt rotors has been used to allow more agile movements and additional flight modes.

In this thesis, a prototype quadrotor is built and taken as reference for the modeling, design and simulation of the entire system. Data used in simulation is experimentally evaluated from the prototype. The controller block is designed such that it can undergo the code generation process without any substantial modifications in its structure. The simulation environment is created taking as reference the software structure running on the used PX4 Flight Stack (Figure 1.1). This is a novel approach when designing and tuning a controller that has to be deployed on a Pixhawk-based hardware running PX4 Flight Stack. The advantage given by this choice is that the simulation environment replicates more accurately the framework in which the controller will work, allowing to take into account the effects of the on-board sensors and the on-board Extended Kalman Filter (EKF) directly in Model-In-The-Loop (MIL) simulation. Two solutions are compared: the first one features a SMC in the inner loop driving both the quadrotor attitude and altitude, whereas in the outer loop a simple PD controller is present to control the North-East position; the second one uses a PID controller for the attitude and altitude control and a customized PDD controller – a PD with a derivative action on the derivative channel. Both controllers are characterized by novel features: the SMC makes use of *quaternions* to describe the quadrotor attitude, while the PID solution needed a customization to enhance its nominal performance to make it comparable to the SMC solution. Furthermore, the influence of an additional payload is analyzed, testing both controllers in *non-nominal* scenarios. Lastly, an experimental test is carried out to characterize the employed propellers thrust in function of the duty cycle of the square wave generated by the autopilot to drive the brushless motors.

Sensors

A critical part when designing multirotors and in general machines that have to or need to interact with the surrounding or with their peers or humans, is the sensors choice and integration. Due to their intrinsic dynamical complexity, multirotors need different sensors to navigate effectively. UAVs are equipped with MEMS sensors to save weight and power consumption; usually accelerometres are used to measure the body acceleration, gyroscopes for body angular velocity, magnetometres for the heading, barometres for the altitude and a Global Positioning System (GPS) for the position. The critical datum among those listed is the position. In the case of indoor applications – or more in general for GPS-denied applications – the challenge is to determine the position of the UAV accurately. Both on-board

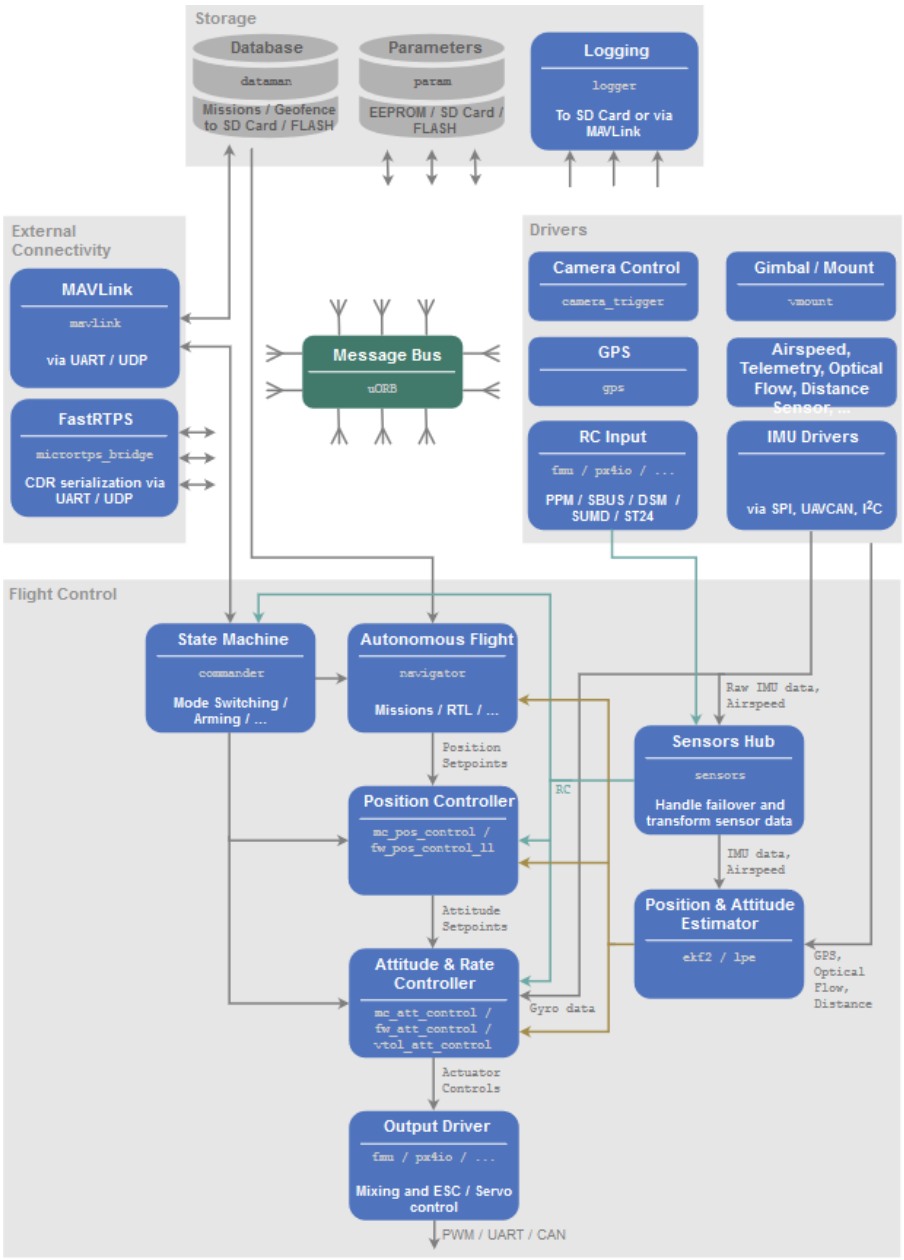


Figure 1.1: Default PX4 High-Level software architecture

and external sensor are exploited. On-board sensors include web-cams, stereoscopic cameras, lidars and Time-of-Flight (ToF) cameras paired with a suitable Simultaneous Localization and Mapping (SLAM) algorithm; conversely external sensors rely on an external infrastructure that interacts with an on-board device. A widely-used system is the Optitrack [1], which tracks the movement of passive spherical components mounted on the multirotor using several high-performance infra-red cameras.

In this work, the cheaper Otus Tracker is exploited (section 2.5), an active on-board sensor, intercepting by means of photo-diode the infra-red beam generated by two external base stations.

1.1 Outline

This work is organized as follows:

- Chapter 2 shows the working principle of a quadrotor, addressing the problem of an *under-actuated* system. Then, it introduces the basic mathematical notions for the derivation of the quadrotor dynamical model and for the controllers design, considering both Euler angles and quaternions to describe the angular quantities. Lastly, a brief description of the prototype quadrotor is present, focusing on its inertial properties, the on-board autopilot (the Pixhawk 2.1 Cube) and companion computer (the Raspberry Pi 3B) and on the sensors used in this particular indoor application.

- Chapter 3 treats the theory and the design of the two controller architectures. First, the typical overall control structure of a quadrotor is presented, then the PID solution is described showing the inner loop and outer loop controllers structures. Lastly, SMC controller theory is introduced and the command law is derived for the attitude control (in the inner loop) and the structure of the outer loop is presented. This chapter describes what is inside the "Controller" block in the simulation model described in the following chapter, addressing the controllers design problem.

- Chapter 4 describes the *ad hoc* simulation environment analyzing each block and providing the basic theoretical notions when needed. The Trajectory Planner, the controller, the plant, its sensors and the Extended Kalman Filter block are described and the numerical value of their main parameters used during simulations are reported. The problem of a too slow outer loop in the PID solution is addressed, justifying the exploited final outer loop PDD structure. Then, a section about evaluation of the thrust curve is included, useful to describe the actuation part included in the plant block. Eventually, simulation results are reported and commented for different patterns. Lastly, the influence of an additional payload is investigated. This chapter highlights the differences between the linear solution and the SMC controller, showing again how PID is limited in the response speed.

- Chapter 5 summarizes the main points and results of this thesis and a proposal about future works is presented, related to the deployment and to the coding aspect.

Chapter 2

Mathematical model

In this section, preliminary mathematical notions and basic notions of the quadrotor are presented. First, theoretical notions about reference frames and spatial transformations are recalled, then the mathematical model of the quadrotor is derived and described in State-Space representation. Lastly, the prototype of the quadrotor used in this work is briefly described.

2.1 Quadrotor working principle

In this section, the conventions and the reference frames used in this work are presented with explicit reference to the quadrotor.

Reference frames

In this work, the North-East-Down (NED) reference frame is used. To describe the motion of the quadrotor, two reference frame are exploited (Figure 2.1): one is fixed, called inertial or NED in which the "Down" arrow points toward the center of the Earth; the other is centered in the quadrotor Center of Gravity (CoG), called body reference frame.

Quadrotor flight mechanisms

Two configurations are available for a quadrotor: an X (cross) configuration, which maximizes the moments generated by the motor thrust, and a + (plus) configurations (Figure 2.2). In this work, the X configuration is used, and the propellers are numbered as in Figure 2.2b. Propeller 1 and 2 spin counter clockwise while 3 and 4 spin clockwise. The spin direction is critical since it counter balances the torque produced by the spinning propellers, preventing the drone from rotating around its Down (z) axis.
As previously stated, the quadrotor is *under-actuated*, so translational motion is

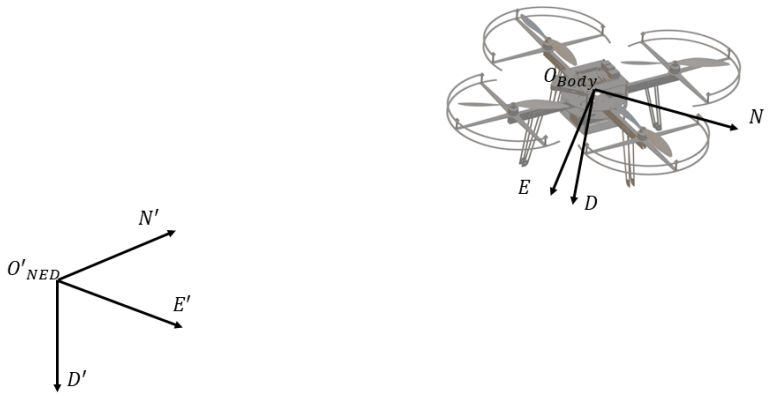
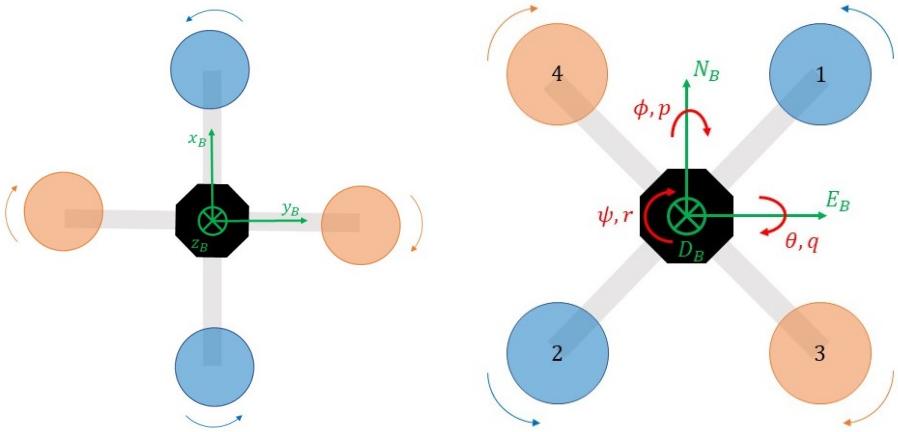


Figure 2.1: The used reference frames



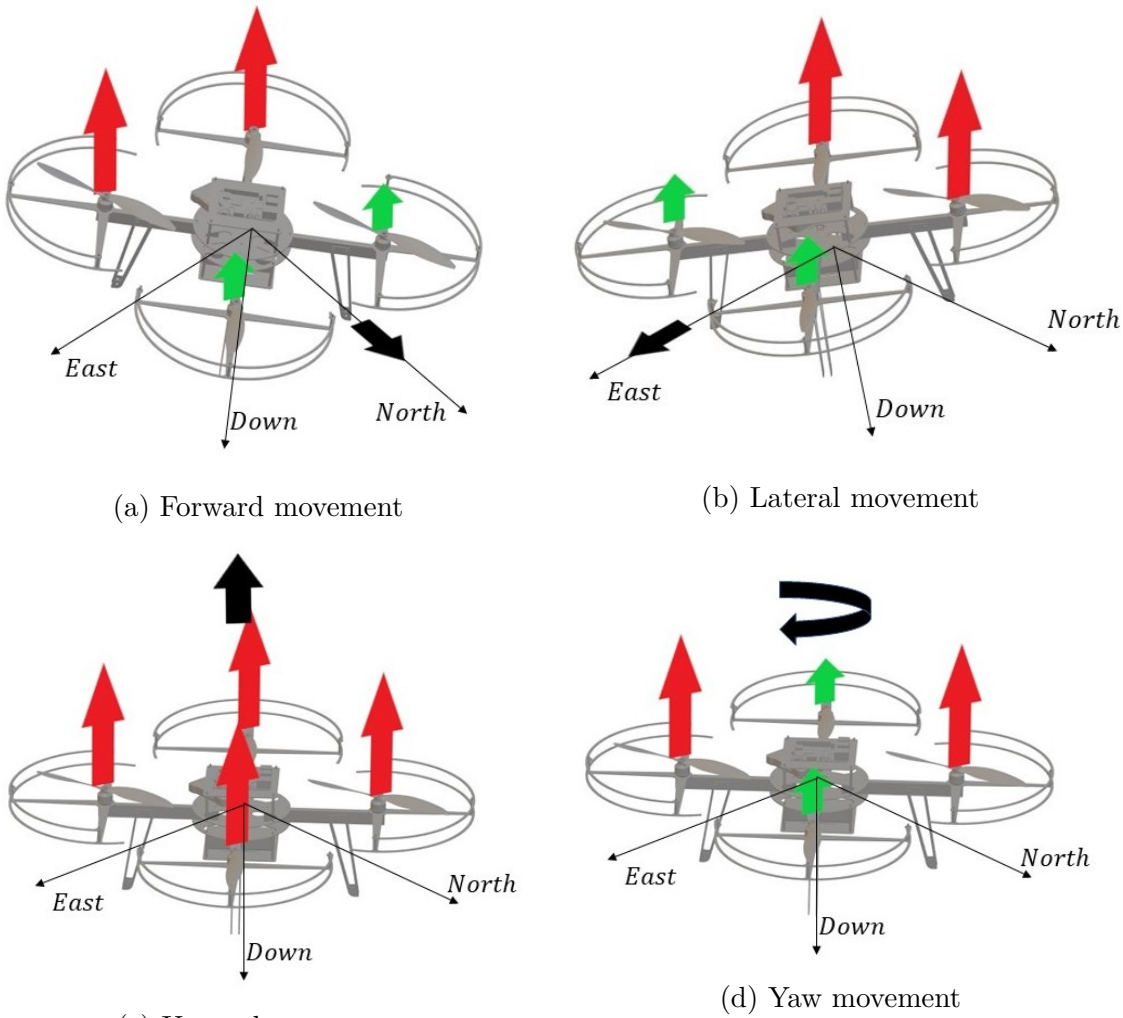
(a) + configuration

(b) X configuration

Figure 2.2: Different types of configuration

strictly coupled with rotational motion. This means that for the quadrotor to move forward, it has to tilt forward, counterbalancing the gravity effect. The same happens for lateral movements. In particular:

- to move **forward** a rotation around negative East axis is needed (Figure 2.3a);
- to move **sideways** a rotation around positive North axis is needed (Figure 2.3b);
- to move **upwards** the four propellers spin such that the attitude remains null (Figure 2.3c);
- to move around its **vertical axis** the propellers moving in the same direction vary the speed simultaneously (Figure 2.3d).



(a) Forward movement

(b) Lateral movement

(c) Upward movement

(d) Yaw movement

Figure 2.3: The 4 basic maneuvers of a quadrotor

2.2 Kinematics

In the following, basic notions of change representation and reference frame rotation are presented.

Reference frame transformation

In this work, both Euler angles and quaternions are used to describe the orientation of the UAV in the three-dimensional space. The former method uses three parameters – namely the angles ϕ,θ and ψ; the latter uses four parameters, the

quaternion.

Euler angles Euler angles can describe any three-dimensional rotation whith a sequence of elementary rotation around three axes (consecutive rotation must be around two different axes). For a better readability, the following notation is introduced:

$$\cos(\cdot) = c_{(\cdot)} \qquad \sin(\cdot) = s_{(\cdot)} \qquad \tan(\cdot) = t_{(\cdot)}$$

The elementary rotations around x,y and z can be described as:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \qquad \mathbf{R}_y = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \qquad \mathbf{R}_z = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Multiplying them, one can obtain a certain rotation around the desired combination of axes. In this case, to describe the orientation of the UAV a ZYX rotation is exploited:

$$\mathbf{R}_{\phi,\theta,\psi} = \mathbf{R}_{body}^{NED} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi + c_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2.2}$$

Furthermore, it is necessary to find a relationship between the Euler angles characterizing the quadrotor attitude w.r.t. the inertial frame and the angular velocities p,q and r in the body reference frame. It results that:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.3}$$

Quaternions The quaternion is an effective method to describe three-dimension rotation with the advantage over Euler angles method of not suffering from singularity, yet needing low computational power to be handled. The quaternion in this thesis is defined as follows:

$$\mathfrak{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \qquad \textit{with} \qquad \mathbf{ijk} = 1$$

A convenient representation is the vector representation:

$$\mathfrak{q} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \tag{2.4}$$

where $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$.

To express rotation from a certain attitude of a certain angle, the *Hamilton* product is used.

Definition 2.2.1 (*Hamilton product*)

Let $\mathfrak{q}$ *and* $\mathfrak{p}$ *be quaternions describing rotations. The* Hamilton *product* $\otimes$ *between* $\mathfrak{q}$ *and* $\mathfrak{p}$ *is equal to:*

$$\mathfrak{q} \otimes \mathfrak{p} = q_0 p_0 - \mathbf{q} \cdot \mathbf{p} + q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p} \tag{2.5}$$

Thanks to this definition, it is possible to define an *error quaternion*, i.e. the distance between the reference quaternion and the actual (estimated) quaternion:

$$\tilde{\mathfrak{q}} = \hat{\mathfrak{q}}^{-1} \otimes \mathfrak{q}_{ref} \tag{2.6}$$

where $\hat{\mathfrak{q}}$ is the current estimated quaternion and $\mathfrak{q}_{ref}$ is the provided reference. Lastly, the quaternion time-derivative must be defined:

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathbf{\Omega q} = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \tag{2.7}$$

2.3 Forces and Moments

In this sections, the mathematical model of the quadrotor is derived. In this process, some assumptions are made in order to simplify the model and force it to a second order ordinary differential equation (ODE). The assumptions are the following:

1. The quadrotor and all its components (i.e. propellers, motors...) are considered as rigid bodies;
2. The quadrotor center of gravity (CoG) is the origin of the body reference frame
3. The actuators are not modeled;
4. The aerodynamic forces are not considered, since they are negligible in indoor applications;
5. The Earth is considered as flat and its rotation is negligible w.r.t. body angular speeds.

The procedure is similar to what is done in [24].
For Newton's second law,

$$\mathbf{F}_B + \mathbf{R}^B_{NED} m\mathbf{g} = m\left[\frac{d\mathbf{v}_B}{dt} + \boldsymbol{\omega}_B \times \mathbf{v}_B\right] \tag{2.8}$$

where $\mathbf{v}_B$ is the relative velocity of the multirotor CoM w.r.t. air mass. Performing the differentiation, one obtains:

$$\frac{\mathbf{F}_B}{m} + \mathbf{R}^B_{NED}\mathbf{g} = [\dot{\mathbf{v}}_B + \boldsymbol{\omega}_B \times \mathbf{v}_B] \tag{2.9}$$

where $\mathbf{v}_B$ is defined as follows and the term $\boldsymbol{\omega}_B\times$ can be thought as:

$$\boldsymbol{\omega}_B\times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \qquad \mathbf{v}_B = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

For Newton's second law applied to rotating bodies:

$$\tau = \left(\frac{d\mathbf{H}}{dt} + \boldsymbol{\omega}_B \times \mathbf{H}\right) \tag{2.10}$$

in which:

$$\mathbf{H} = \mathbf{J}\boldsymbol{\omega}_B = \begin{bmatrix} J_x & J_{xy} & J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -Jyz & -J_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.11}$$

Rewriting Equation 2.10 using Equation 2.11 and isolating $\dot{\boldsymbol{\omega}}_B$:

$$\dot{\boldsymbol{\omega}}_B = -\mathbf{J}^{-1}(\boldsymbol{\omega}_B \times (\mathbf{J}\boldsymbol{\omega}_B)) + \mathbf{J}^{-1}\boldsymbol{\tau} \tag{2.12}$$

2.4 State-Space Model

Starting from the equations presented in the previous section the quadrotor mathematical model can be written as a *nonlinear* differential equation of the form:

$$\dot{\mathbf{x}} = \mathbf{f}_{(x)} + \mathbf{g} \cdot \mathbf{u} \tag{2.13}$$

This is convenient when dealing with control theory, and in particular when dealing with controllers requiring the knowledge of the model (or plant) to be controlled. As detailed in section 3.2, Equation 2.13 is exactly the specific form required to the model by the Sliding Mode Control theory, one of the two solutions implemented in this work.

Let $\mathbf{x}$ be the vector state and $\mathbf{u}$ defined as:

$$\mathbf{x} = \begin{bmatrix} p_N \\ p_E \\ h \\ \phi \\ \theta \\ \psi \\ u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{2.14}$$

To describe the first three states, it is sufficient a conversion from the linear velocities in the body reference frame to the inertial reference frame. Using Equation 2.2 to rotate $\mathbf{v}_B$, one obtains:

$$\begin{bmatrix} \dot{p}_N \\ \dot{p}_E \\ \dot{h} \end{bmatrix} = \mathbf{R}_{body}^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.15}$$

The attitude variation in the inertial reference frame can be described starting from the body angular velocity of the quadrotor. With reference to Equation 2.3, it results that:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ \dfrac{1}{\cos\theta}(q\sin\phi + r\cos\phi) \end{bmatrix} \tag{2.16}$$

Body reference frame linear acceleration is well described by Equation 2.9, which can be rewritten isolating $\dot{\mathbf{v}}_B$, solving the vector product and multiplying $\mathbf{R}_{body}^{NED}$ by the gravity vector:

$$\dot{\mathbf{v}}_B = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw - g\sin\theta + \frac{F_N}{m} \\ -ru + pw + g\sin\phi\cos\theta + \frac{F_E}{m} \\ qu - pv + g\cos\phi\cos\theta + \frac{F_D}{m} \end{bmatrix} \tag{2.17}$$

Lastly, solving Equation 2.12 introducing convenient constants c_i describing inertia properties, one obtains:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} q(c_1 r + c_2 p) + c_3\tau_x + c_4\tau_z \\ prc_5 - (p^2 - r^2)c_6 + c_7\tau_y \\ q(c_8 p - c_2 r) + c_4\tau_x + c_9\tau_z \end{bmatrix} \tag{2.18}$$

where:

$$c_1 = \frac{(J_y - J_z)J_z - J_{xz}^2}{J_x J_z - J_{xz}^2} \qquad c_2 = \frac{(J_x - J_y + J_z)J_{xz}}{J_x J_z - J_{xz}^2} \qquad c_3 = \frac{J_z}{J_x J_z - J_{xz}^2}$$
$$c_4 = \frac{J_{xz}}{J_x J_z - J_{xz}^2} \qquad c_5 = \frac{J_z - J_x}{J_y} \qquad c_6 = \frac{J_{xz}}{J_y}$$
$$c_7 = \frac{1}{J_y} \qquad c_8 = \frac{(J_x - J_y)J_x + J_{xz}^2}{J_x J_z - J_{xz}^2} \qquad c_9 = \frac{J_x}{J_x J_z - J_{xz}^2}$$

Grouping Equation 2.15, Equation 2.16, Equation 2.17 and Equation 2.18 and rewriting them following Equation 2.13 it is possible to express the dynamical model in state-space form, in which:

$$\mathbf{f}_{(x)} = \begin{bmatrix} \mathbf{R}_{body}^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ \frac{1}{\cos\theta}(q\sin\phi + r\cos\phi) \\ rv - qw - g\sin\theta \\ -ru + pw + g\sin\phi\cos\theta \\ qu - pv + g\cos\phi\cos\theta \\ q(c_1 r + c_2 p) \\ prc_5 - (p^2 - r^2)c_6 \\ q(c_8 p - c_2 r) \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & & & & \vdots \\ 0 & & \ddots & & & \vdots \\ 0 & & & \ddots & & \vdots \\ 0 & & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \frac{1}{m} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \frac{1}{m} & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \frac{1}{m} & 0 & \cdots & 0 \\ 0 & 0 & 0 & c_3 & 0 & c_4 \\ 0 & 0 & 0 & 0 & c_7 & 0 \\ 0 & 0 & 0 & c_4 & 0 & c_9 \end{bmatrix} \tag{2.19}$$

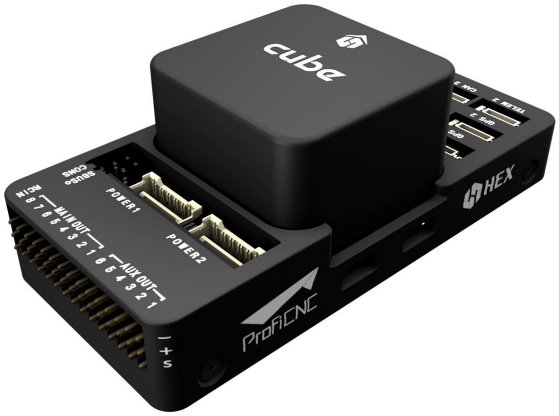
2.5 DR0N3 Quadrotor

The real-world multirotor prototype whose main features and technical characteristics are used in this work is a quadrotor based on the Pixhawk Autopilot. The quadrotor chassis is made of four aluminum-alloy arms and a carbon fiber central body in which hardware is located. The brushless motors are paired with 10″ inch carbon-fiber propellers. Also, carbon-fiber propeller protections are mounted for safety reasons. The chassis is manufactured by MAVTech [2] (Figure 2.4). The inertia properties are computed from the CAD model and are summarized in Table 2.1. The Li-Po battery can guarantee up to 15*min* of flight time. For what concerns the on-board equipment, DR0N3 is provided with the Pixhawk 2.1 Cube Autopilot (Figure 2.5a) and a Raspberry Pi 3B (Figure 2.5b).

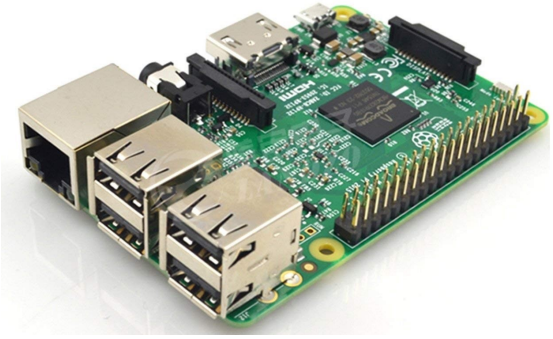
Dimension	Value
Mass m	$1.5kg$
Inerta moment J_x	$0.0170kgm^2$
Inerta moment J_y	$0.0173kgm^2$
Inerta moment J_z	$0.0308kgm^2$
Propellers diameter	$10''$

Table 2.1: Main characteristics of the prototype quadrotor

Figure 2.4: The real-world prototype



(a) The on-board autopilot



(b) The Raspberry Pi

Pixhawk Autopilot

The Pixhawk 2.1 is versatile autopilot, part of the Pixhawk project [3]. It is based on the FMUv3 open hardware design and it features an ARM-based Cortex M4

microprocessor and ST Microelectronics® components. It comes with the necessary sensors and interfaces to fly a drone. The main characteristics (as taken from [4]) are listed here:

- PROCESSOR
  - 32-bit ARM Cortex M4 core with FPU
  - 168 Mhz/256 KB RAM/2 MB Flash
  - 32-bit failsafe co-processor
- SENSORS
  - Three redundant IMUs (accels, gyros and compass)
  - InvenSense MPU9250, ICM20948 and/or ICM20648 as first and third IMU (accel and gyro)
  - ST Micro L3GD20+LSM303D or InvenSense ICM2076xx as backup IMU (accel and gyro)
  - Two redundant MS5611 barometers
- INTERFACES
  - 14x PWM servo outputs (8 from IO, 6 from FMU)
  - S.Bus servo output
  - R/C inputs for CPPM, Spektrum / DSM and S.Bus
  - Analogue / PWM RSSI input
  - 5x general purpose serial ports, 2 with full flow control
  - 2x I2C ports
  - SPI port (un-buffered, for short cables only not recommended for use)
  - 2x CAN Bus interface
  - 3x Analogue inputs (3.3V and 6.6V)
  - High-powered piezo buzzer driver (on expansion board)
  - High-power RGB LED (I2C driver compatible connected externally only)
  - Safety switch / LED
  - Optional carrier board for Intel Edison

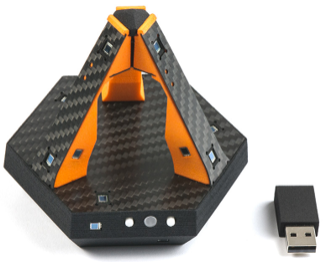
The Pixhawk Cube can support multiple flying stacks, the most used are Ardupilot and PX4. For this particular application, PX4 is the employed firmware. The main features of this flight stack are:

- Controls many different vehicle frames/types, including aircraft (multicopters, fixed wing aircraft and VTOLs), ground vehicles and underwater vehicles.
- Great choice of hardware for vehicle controller, sensors and other peripherals.
- Flexible and powerful flight modes and safety features.
- Runs on NuttX Real-Time OS.
- Can be freely modified and can work with generated code.

The PX4 comes with Software-In-The-Loop and Processor-In-The-Loop modules, very useful when debugging a customized flight stack. Furthermore, it is available a MATLAB®/Simulink® Add-on that allows to build and deploy a Simulink model on the target hardware.

Otus Tracker

Usually, multirotors use sensors able to locate the machine w.r.t. some reference frame. For outdoor applications, a Global Positioning System (GPS) is exploited which can track the drone on the Earth surface. However, this solution is not suitable for indoor application as obstacle between the GPS antenna and the satellite critically degrade the tracking performance, resulting in an unreliable method. Inside buildings, or in general in GPS-denied environments, vision-based, sound-based and Motion Capture (MoCap) systems are exploited. The Otus Tracker (Figure 2.6a) is a MoCap sensor mounted on the prototype quadrotor. It features good accuracy and precision, a high-bandwidth communication and a lightweight chassis.The Otus Tracker needs two base stations to work, part of the HTC Vive product series (Figure 2.6b). It provides the UAV its position as well as its attitude w.r.t. the inertial reference frame.



(a) The MoCap sensor – the Otus Tracker

(b) Base emitters for the Otus Tracker

Chapter 3

Design of control techniques

In this chapter the description of the control framework and the design of different controllers is addressed. The overall structure used in this application is a cascade controller as in Figure 3.1, in which the inner loop is related to the fast dynamics, and produces the signals regulating the attitude and the angular velocities of the multirotor, while the outer loop is related to the slow dynamics of the system, and tracks the desired NED position in the X-Y plane. In this work, a

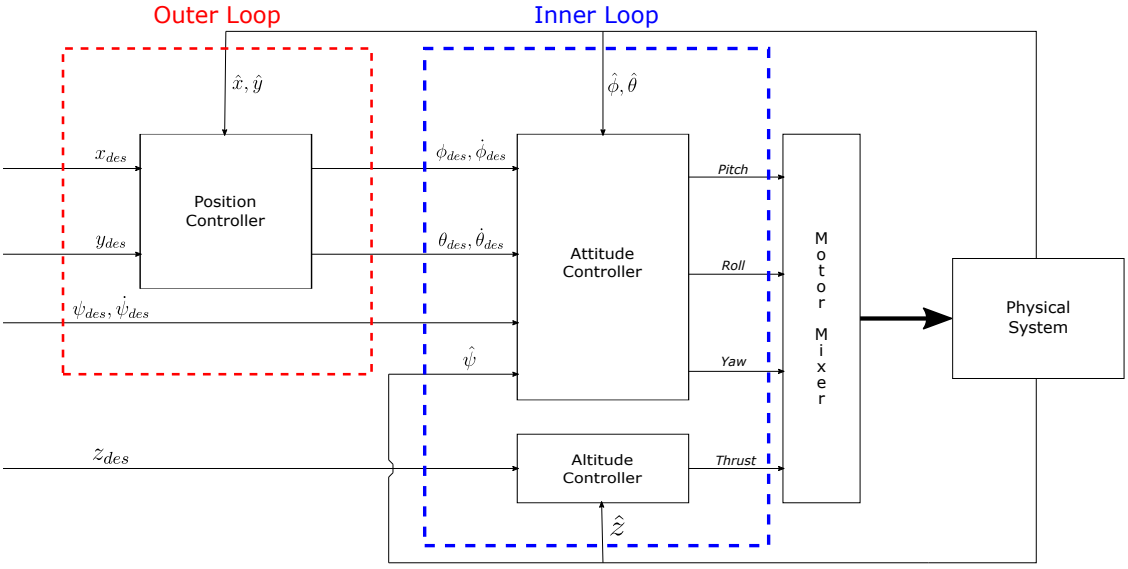


Figure 3.1: Overall control architecture

classical controller, based on a linear model of the system, and a variable structure controller, based on the nonlinear equations of motion, are designed. The former is a Proportional-Integral-Derivative (PID) controller in which the parameters are found by trial and error, the latter is a *First-Order Sliding Mode Controller* (SMC). Both controllers are implemented in the Inner Loop only, and they are compared in

terms of performance and robustness against uncertainties. The initial outer loop structure is the same for both solutions.

3.1 PID controller

The PID controller is a widely-used linear controller in which the command action is generated taking into account the difference between a given setpoint or reference signal and the measured output of the system, that is the error e. A complete PID controller uses information about the error at a certain time (Proportional action – P), its derivative (Derivative action – D) and its integral (Integral action – I) to provide a linear feedback action to the controlled system. The general form of a PID controller is:

$$u_{(t)} = K_P e_{(t)} + K_I \int_0^t e_{(t)} dt + K_D \frac{de_{(t)}}{dt} \tag{3.1}$$

where $e_{(t)} = r_{(t)} - y_{(t)}$. The main advantage of this type of controller is its simplicity in design and tuning. However, when dealing with nonlinear systems as in this work, the PID controller can show degraded performance in function of the current system operating point, and cannot achieve *optimal* control. Nevertheless, in literature various authors adopt a cascade PID controller for quadrotors.
Several approaches to PID design are exploited in literature; the main ones are summarized in [25]. In this work, the structure shown in Figure 3.2 is exploited, which is suitable for practical implementations. The adopted scheme differs from

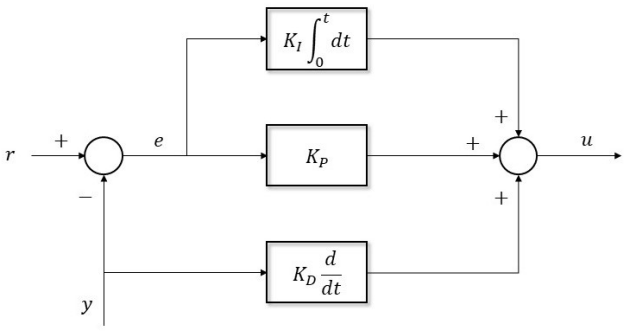


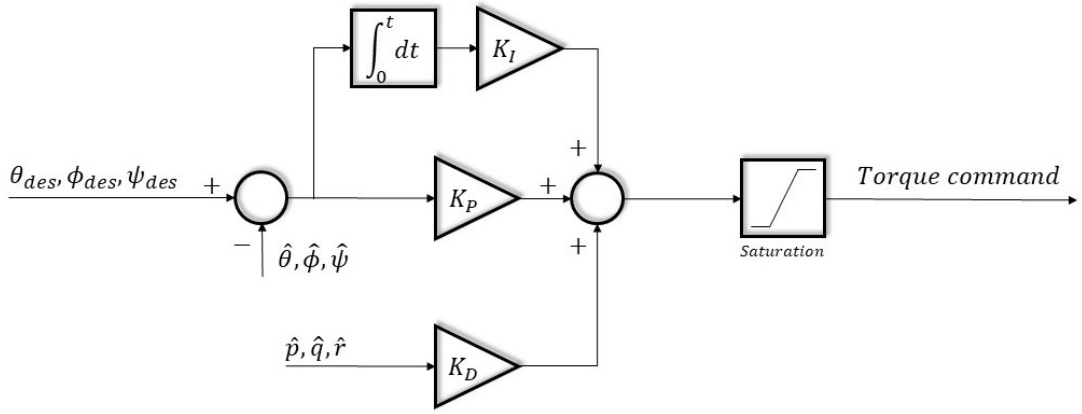
Figure 3.2: Implemented PID architecture

the classical PID controller in the derivative action, which uses directly the output of the system y in place of the error e.

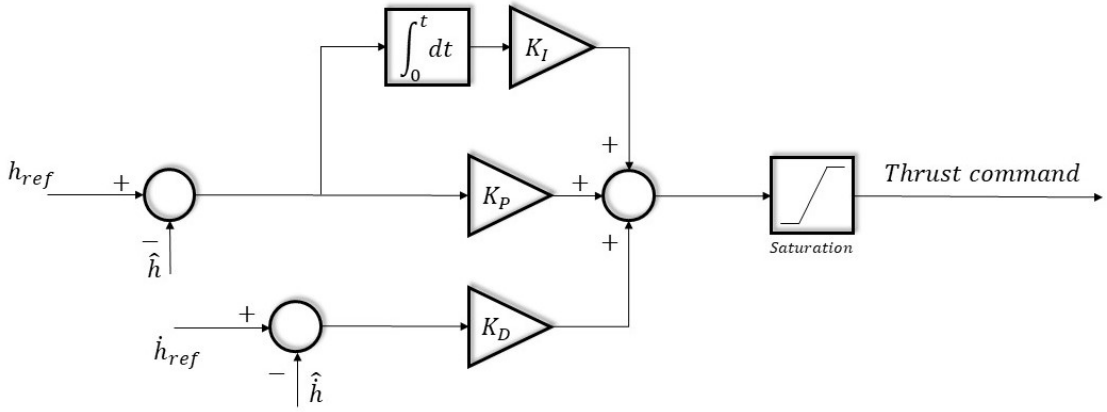
3.1.1 Inner Loop

The inner loop is related to the attitude and altitude control of the quadrotor. It generates torque commands having as input a desired attitude and a thrust having

as input the desired altitude and vertical velocity. The structure of the attitude channel is shown in Figure 3.3a, while the altitude channel is shown in Figure 3.3b. The loop is tuned such that the output values have a physical unit of measurement: the thrust command is in Newton $[N]$ and the torque command is in Newton-meters $[Nm]$. In this way, the design and the tuning are more intuitive.
In order to avoid too large command values, a saturation block is included in the controller to limit the computed force/torque. The upper and lower values are chosen after the propellers are characterized, and they are listed in section 4.



(a) Attitude channel



(b) Altitude channel

Figure 3.3: Implemented inner loop architecture

3.1.2 Outer Loop

The outer loop acts as a position controller. The proposed structure is presented in Figure 3.4. The loop receives as inputs the desired x_{ref} and y_{ref} and provides the needed attitude angles ϕ_{des} and θ_{des} to perform the maneuver and their time derivatives $\dot{\phi}_{des}$ and $\dot{\theta}_{des}$. However, this solution proves to be too slow in simu-

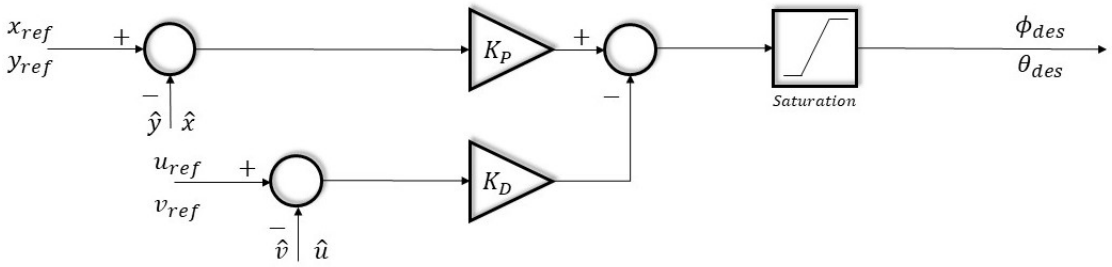


Figure 3.4: Preliminary outer loop PD architecture

lation. When tracking the velocity reference, the PD controller produces a high overshoot and settling time and generates output signals (i.e. ϕ_{des} and θ_{des}) with large oscillations which affect the inner loop. Thus, the controller is modified to overcome this problem. Since the velocity channel exhibits too large oscillation, a Derivative controller D is applied acting on the velocity error as a damper. This solution considerably reduces the oscillations on the velocity channel (Figure 3.5).

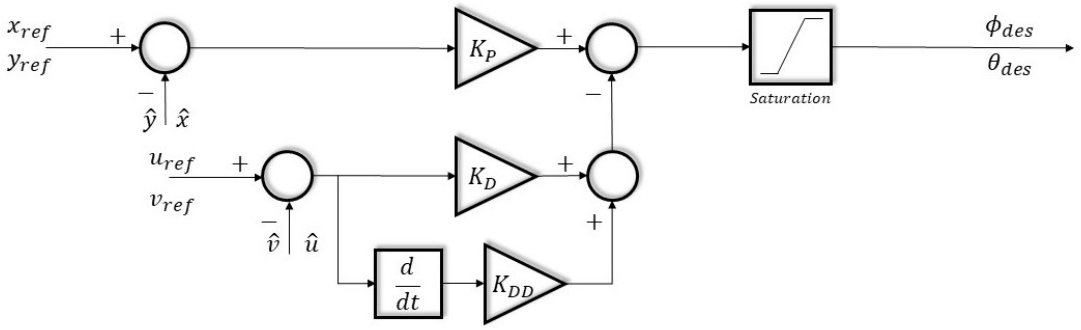


Figure 3.5: Implemented outer loop PD architecture

3.2 First-Order Sliding Mode Controller

Sliding Mode Control (SMC) is a class/type of Variable Structure Control (VSC) in which states are enforced to lie on a suitable sliding surface, which satisfies the attractiveness condition.

Definition 3.2.1

Let the (SISO) system in the affine form be:

$$\dot{x}_{(t)} = f_{(x,t)} + g_{(x,t)} u_{(x,t)} \tag{3.2}$$

and consider a time-varying surface

$$S_{(t)} = \{ s_{(x,t)} \in \mathbb{R}^{(n)} \mid s_{(x,t)} = 0 \} \tag{3.3}$$

If the input u presents a discontinuity in $s_{(x,t)} = 0$ such that:

$$u_{(x,t)} = \begin{cases} u^+ & \text{if } s_{(x,t)} > 0 \\ u^- & \text{if } s_{(x,t)} < 0 \end{cases}$$

and satisfies the attractiveness condition:

$$s_{(x,t)} \dot{s}_{(x,t)} < 0 \tag{3.4}$$

*then the control is said to be in **sliding mode**.*

The sliding surface can be arbitrarily chosen, in general it can be a linear combination of the state variables. In this work the sliding surface proposed by [23] is exploited:

$$s_{(x)} = \left(\frac{d}{dt} + \lambda \right)^{n-1} \tilde{x} \tag{3.5}$$

in which $\tilde{x} = x_{des} - x$ is the tracking error of the state x. The following conditions must be satisfied in order to have an first-order sliding mode controller:

Condition 3.1. $\dot{s} = 0$ *to enforce the system on the sliding surface s – the surface is said to be invariant*

Condition 3.2. $s\dot{s} < 0$ *to guide the system towards the sliding surface – the surface is said to be attractive.*

The control law structure adopted here [26] is:

$$u_{(x,t)} = u_{eq} + u_{dis} = u_{eq} - k \operatorname{sign}(s_{(x)}) \tag{3.6}$$

where u_{eq} is the equivalent control when $\dot{s}_{(x)} = 0$ and u_{dis} is the discontinuous part of the control action.

When the SMC is acting on the system, the state trajectory evolves in time throughout two stages:

1. **Reaching phase**: the system gets closer to the sliding surface until the intersection occurs;

2. **Sliding phase**: the system "slides" on the sliding surface.

The SMC is in general able to deliver high control performance thanks to the *a priori* information about the controlled system dynamics. Furthermore, the SMC is robust against model uncertainties and noise. Nevertheless, the design of such a controller requires the system to be affine in the input u. Also, the command activity of this kind of control is in general high and high frequency oscillations are present in the command action that lead to higher power consumption and higher actuators solicitations. This phenomenon is called *chattering*. Figure 3.6 shows the evolution of the state trajectory for a two-states system highlighting the *chattering* phenomenon.

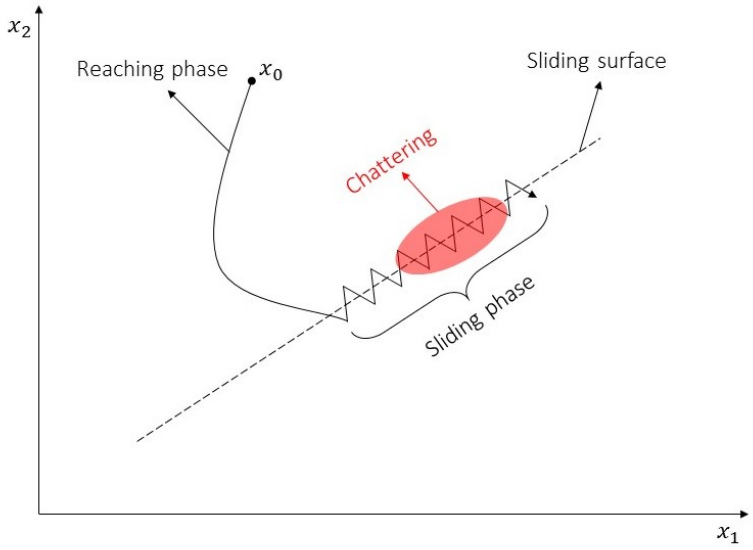


Figure 3.6: Phase portrait of a Sliding Mode controlled system. Note that once the state trajectory approaches the sliding surface (defined in 3.3), high-frequency oscillations appear.

Lyapunov stability

To prove the stability of a candidate controller, the Lyapunov's Direct Method is exploited.

Theorem 1 (*Lyapunov's Stability*)

Let $V_{(x,t)}$ be a scalar function of the state x with continuous first order derivative. If:

- *$V_{(x,t)}$ is positive definite;*
- *$\dot{V}_{(x,t)}$ is negative definite;*
- *$V_{(x,t)} \to \infty$ as $\|x\| \to \infty$,*

then the equilibrium at the origin is ***globally asymptotically stable.***

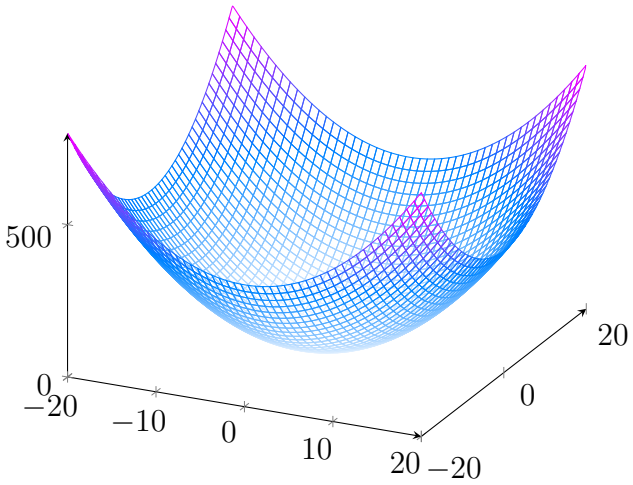


Figure 3.7: Lyapunov function

3.2.1 Inner Loop

Preliminaries

The synthesis of the controller is carried out exploiting the equations of the quadrotor referred to the NED fixed frame. This is done to simplify the control laws and to match the signals the quadrotor generates. The vertical dynamics can be described as:

$$\ddot{h}_{NED} = g - \cos\theta \cos\phi \frac{F_z}{m} \tag{3.7}$$

while the rotational dynamics around the three axis is:

$$\begin{aligned}\ddot{\phi} &= \frac{J_y - J_z}{J_x}\dot{\theta}\dot{\psi} + \frac{\tau_x}{J_x} \\ \ddot{\theta} &= \frac{J_z - J_x}{J_y}\dot{\phi}\dot{\psi} + \frac{\tau_y}{J_y} \\ \ddot{\psi} &= \frac{J_x - J_y}{J_z}\dot{\theta}\dot{\phi} + \frac{\tau_z}{J_z}\end{aligned} \tag{3.8}$$

Altitude control

Let $\tilde{h}$ be the tracking error of the state h, defined as:

$$\tilde{h} = h_{des} - \hat{h}_{NED} \tag{3.9}$$

where h_{des} is the desired altitude in NED fixed frame coordinates and $\hat{h}_{NED}$ is the estimated altitude. Remebering 3.5, a candidate sliding surface could be:

$$s_{(h)} = \dot{\tilde{h}} - \lambda_{alt}\tilde{h} \tag{3.10}$$

and its time derivative is equal to:

$$\dot{s}_{(h)} = \ddot{\tilde{h}} - \lambda_{alt}\dot{\tilde{h}} \tag{3.11}$$

Using 3.9 and substituting 3.7 in 3.11, one obtains:

$$\dot{s}_{(h)} = \ddot{h}_{des} - g + \cos\theta\cos\phi\frac{F_z}{m} + \lambda(\dot{h}_{des} - \dot{\hat{h}}_{NED}) \tag{3.12}$$

For convenience $\ddot{h}_{des}$ is set to zero.

Applying Condition 3.1 and Condition 3.2 by adding the discontinuous term and solving for F_z, the resulting control law is:

$$F_z = \frac{m}{\cos\theta\cos\phi}(g - \lambda_{alt}\tilde{h}) - Ksign(s_{(h)}) \tag{3.13}$$

where $\dot{\tilde{h}}$ and $\tilde{h}$ are the errors on the vertical position and velocity in the inertial frame, m the mass of the quadrotor, g the gravitational acceleration and λ_{alt} and K positive parameters to be tuned. Note that the structure 3.6 is respected.

Stability analysis

The candidate Lyapunov function is:

$$V_{(s)} = \frac{1}{2}s^2$$

The time derivative of the Lyapunov function is equal to:

$$\dot{V}_{(s)} = s\dot{s}$$

It is easy to note that $V_{(s)} \succ 0$ and $\dot{V}_{(s)} \prec 0$ since

$$\dot{s}_{(z)} = -k\,\mathrm{sign}(s_{(z)})$$

The discontinuous term ensures the Lyapunov stabilty.

Attitude control

Let $\dot{\tilde{\phi}}$, $\dot{\tilde{\theta}}$ and $\dot{\tilde{\psi}}$ be the tracking error of the angular velocities in the NED fixed reference frame $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ respectively, defined as:

$$\begin{aligned} \dot{\tilde{\phi}} &= \dot{\phi}_{des} - \dot{\hat{\phi}} \\ \dot{\tilde{\theta}} &= \dot{\theta}_{des} - \dot{\hat{\theta}} \\ \dot{\tilde{\psi}} &= \dot{\psi}_{des} - \dot{\hat{\psi}} \end{aligned} \tag{3.14}$$

where $\dot{\alpha}_{des}, \alpha = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$ is the desired angular speed and $\dot{\hat{\alpha}}$ is the estimated angular speed – in this application those signals come from the *Extended Kalman Filter* (EKF) of the autopilot. Also, let $\tilde{\mathbf{q}}$ be the error quaternion as defined in Equation 2.6:

$$\tilde{\mathbf{q}} = \begin{bmatrix} \tilde{q}_0 \\ \tilde{\mathbf{q}} \end{bmatrix} \tag{3.15}$$

Recalling the chosen structure 3.6, the candidate sliding surface for attitude control is:

$$\mathbf{s}_{(\phi,\theta,\psi,\mathbf{q})} = \begin{bmatrix} \dot{\tilde{\phi}} \\ \dot{\tilde{\theta}} \\ \dot{\tilde{\psi}} \end{bmatrix} + \mathbf{\Lambda}\tilde{\mathbf{q}} = \begin{bmatrix} \dot{\tilde{\phi}} \\ \dot{\tilde{\theta}} \\ \dot{\tilde{\psi}} \end{bmatrix} + \begin{bmatrix} \lambda_{roll} & 0 & 0 \\ 0 & \lambda_{pitch} & 0 \\ 0 & 0 & \lambda_{yaw} \end{bmatrix} \tilde{\mathbf{q}} \tag{3.16}$$

and its time derivative is equal to:

$$\dot{\mathbf{s}}_{(\phi,\theta,\psi,\mathbf{q})} = \begin{bmatrix} \ddot{\tilde{\phi}} \\ \ddot{\tilde{\theta}} \\ \ddot{\tilde{\psi}} \end{bmatrix} + \mathbf{\Lambda}\dot{\tilde{\mathbf{q}}} \tag{3.17}$$

where $\dot{\tilde{\mathbf{q}}}$ can be found using Equation 2.7. Using 3.14 and substituting 3.8 in 3.17, one obtains:

$$\dot{\mathbf{s}} = \begin{bmatrix} \ddot{\phi}_{des} - \dfrac{J_y - J_z}{J_x}\dot{\psi}\dot{\theta} - \dfrac{\tau_x}{J_x} \\ \ddot{\theta}_{des} - \dfrac{J_z - J_x}{J_y}\dot{\psi}\dot{\phi} - \dfrac{\tau_y}{J_y} \\ \ddot{\psi}_{des} - \dfrac{J_x - J_y}{J_z}\dot{\theta}\dot{\phi} - \dfrac{\tau_z}{J_z} \end{bmatrix} + \mathbf{\Lambda}(\dot{\mathbf{q}}_{des} - \dot{\hat{\mathbf{q}}}) \tag{3.18}$$

Similarly as before, solving for τ_i and applying Condition 3.1 and Condition 3.2, the resulting control law is:

$$\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = -\begin{bmatrix}(J_y - J_z)\dot{\psi}\dot{\theta}\\ (J_z - J_x)\dot{\psi}\dot{\phi}\\ (J_x - J_y)\dot{\theta}\dot{\phi}\end{bmatrix} + \mathbf{\Lambda}\begin{bmatrix}J_x & 0 & 0\\ 0 & J_y & 0\\ 0 & 0 & J_z\end{bmatrix}\tilde{\mathbf{q}} - \mathbf{K}\,\mathrm{sign}(\mathbf{s}) \tag{3.19}$$

where $\dot{\tilde{\phi}}$, $\dot{\tilde{\theta}}$ and $\dot{\tilde{\psi}}$ are the derivative of the error on the euler angles, $\mathbf{\Lambda}$ and $\mathbf{K}$ positive-definite matrices of the parameters to be tuned and $\tilde{\mathbf{q}}$ is the imaginary part of the error quaternion $\tilde{\mathbf{q}}$

Stability analysis

The analysis is similar to the one performed for the altitude control. The candidate Lyapunov function is:

$$V_{(\mathbf{s})} = \frac{1}{2}\mathbf{s}^\intercal \cdot \mathbf{s}$$

The time derivative of the Lyapunov function is equal to:

$$\dot{V}_{(\mathbf{s})} = \dot{\mathbf{s}}^\intercal \mathbf{s}$$

3.2.2 Outer Loop

The outer loop of the SMC solution is a PD controller following the architecture of Figure 3.2. The structure is shown in Figure 3.8. It is worth noting the presence of an another output the loop provides, that is $\hat{\phi}_{des}$ and $\hat{\theta}_{des}$. This is needed due to the fact that the SMC is designed so that it can accept reference values of the body angular speed. $\hat{\phi}_{des}$ and $\hat{\theta}_{des}$ are determined simply multiplying the euler angles and then saturating the output, as if it was a Proportional controller with gain K.

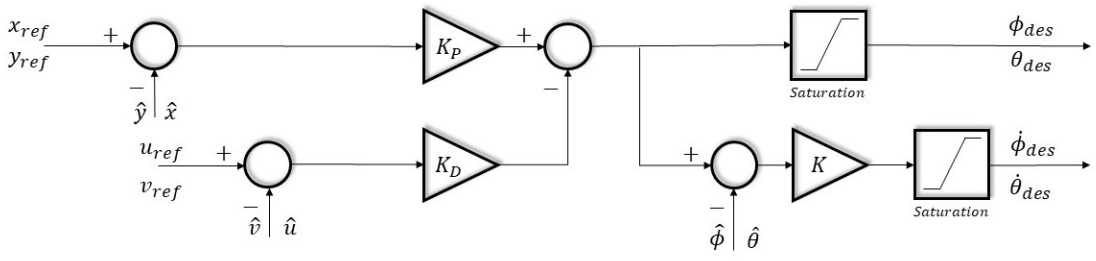


Figure 3.8: SMC outer loop PD architecture

3.3 Motor Mixer

The motor mixer block transforms the physical values (thrust and moments) generated by the controller into the Power-Width Modulation (PWM) Duty Cycle (DC), so that the controller can communicate with the Electronic Speed Controllers (ESC) of the motors. The values ν of the DC used by the autopilot ranges between 1000 (minimum when armed) and 2000 (maximum). The equations relating the inputs and the outputs are those used as default in the PX4 Firmware.
Calling R, P and Y respectively the required roll, pitch and yaw moment to track the given reference and T the required thrust, the equations are:

$$\begin{aligned}
\nu_{(1)} &= \left(\frac{-R+P-Y}{2}T+T\right)1000+idle \\
\nu_{(2)} &= \left(\frac{R-P-Y}{2}T+T\right)1000+idle \\
\nu_{(3)} &= \left(\frac{R+P+Y}{2}T+T\right)1000+idle \\
\nu_{(4)} &= \left(\frac{-R-P+Y}{2}T+T\right)1000+idle
\end{aligned}$$

where $\nu_{(i)}$ is the DC of the i^{th} motor and $idle$ is the DC set when armed (usually is equal to 1000). This type of motor mixer shows better performance than a simple motor mixer in which the output DC is a linear combination of the required forces and moments.

Chapter 4

Simulation results

In this chapter, the complete model used in simulation is presented and test outcomes are reported and analyzed. In particular, in the first part the Simulink model is broken down and each block is commented and characterized with its most important values. Then an evaluation of the thrust curve is carried out. The second part deals with simulation results.
Simulation are performed to tune, validate and compare the synthesized controllers without the need to act directly on the quadrotor, leading to a significant saving on time and material. The software used is MATLAB®/Simulink® R2019a.

To make the simulations as close as possible to the real world scenario, the simulation model includes a model of the on-board sensors and an Extended Kalman Filter as an observer. In addition to this, a trajectory planner is implemented to generate complex trajectories and perform waypoint following. The overall scheme is shown in Figure 4.1.

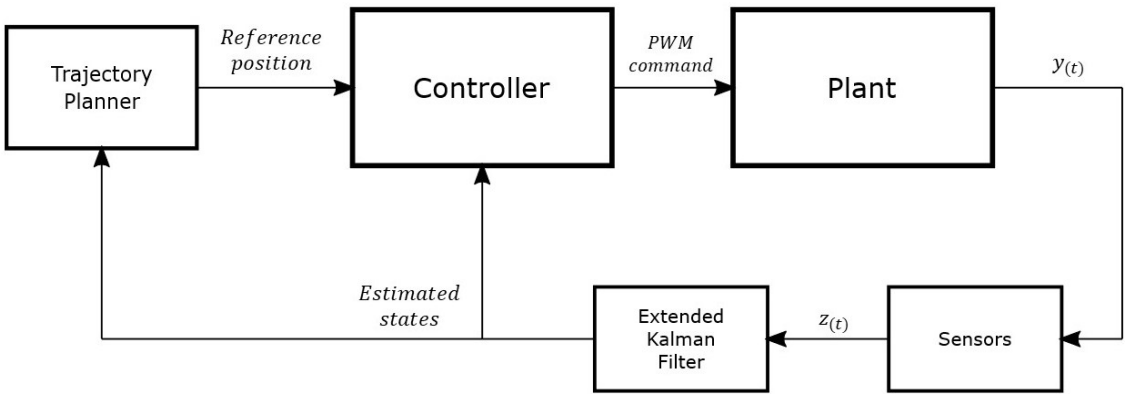
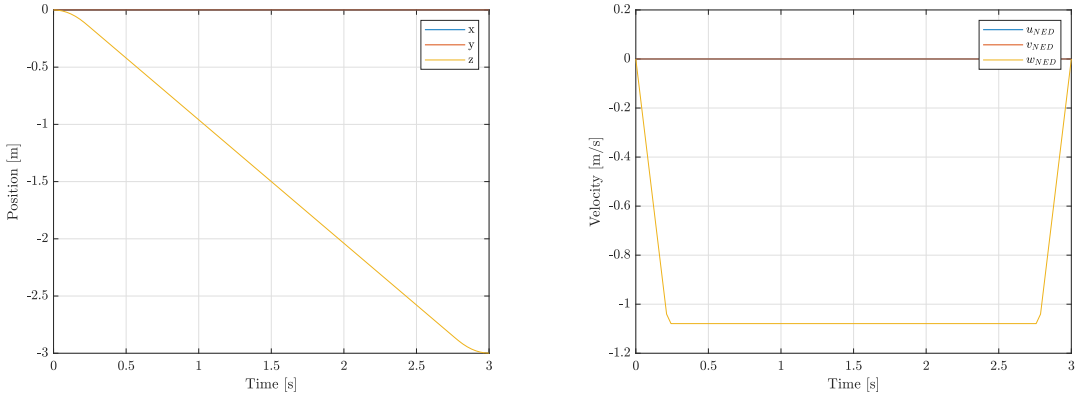


Figure 4.1: Simulation model

In the following, each block is quickly presented.

Trajectory Planner

The trajectory planner is an algorithm that generates a time-dependent path given a set of waypoints with their Times of Arrival (ToA). It imposes a trapezoidal linear velocity profile on the segment linking two consecutive waypoints and it computes by integration the quadrotor reference NED position as a function of time. The maximum acceleration is set and it is equal to $\frac{g}{2}$, while the maximum velocity is determined such that the maneuver can be completed respecting the assigned ToA. The algorithm receives as input arguments the waypoints in the NED inertial frame with their Time of Arrival (ToA) and the number of points between two consecutive waypoints and gives as output the reference trajectory in the NED reference frame and the reference time vector. Figure 4.2 shows the generated profiles for a takeoff maneuver.



(a) Position profile in time

(b) Trapezoidal velocity profile in time

Figure 4.2: Trajectory planner output for a take off maneuver

Plant

The plant block includes the mathematical model of the quadrotor Equation 2.13 and of the actuators used in the target quadrotor (Figure 4.3). To design the "Actuator" block, experimental tests are performed using a thrust stand (RCBenchmark Series 1520) on which the brushless motor with its Electronic Speed Controller (ESC) and its propeller are mounted. The resulting model provides a relationship between the Pulse-Width Modulation(PWM) signal provided to the motor ESC and the force and moment generated. This allows to link the signal coming from the motor mixer in the "Controller" block expressed as a PWM duty cycle to the mathematical model of the quadrotor dynamics, which accepts as signals forces and moments. The experimental test is treated in section 4.1.

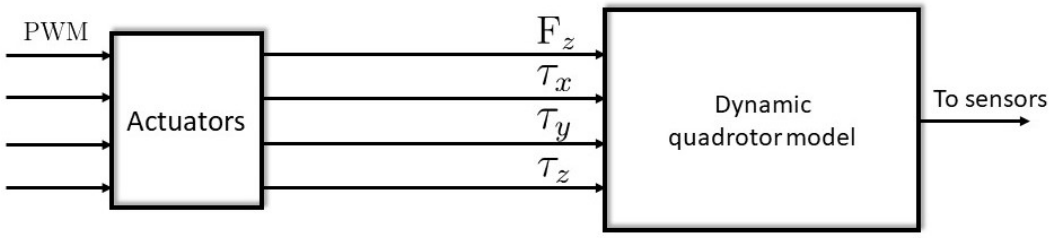


Figure 4.3: The plant block

Controller – PID

In the following, the controller parameters determined by the trial and error tuning are reported. Values listed in Table 4.1 are those used in simulation. Refer to Figure 3.3, Figure 3.4 and Figure 3.5. As described in subsection 3.1.2, the outer

Inner Loop		
Channel	Parameter	Value
Pitch	K_P	0.08
	K_I	0.002
	K_D	0.08
	$Saturation$	$\pm 3.2Nm$
Roll	K_P	0.08
	K_I	0.002
	K_D	0.08
	$Saturation$	$\pm 3.2Nm$
Yaw	K_P	0.15
	K_I	0.0001
	K_D	0.2
	$Saturation$	$\pm 1Nm$
Thrust	K_P	60
	K_I	70
	K_D	40
	$Saturation$	$32N$

(a) Inner Loop PID parameters

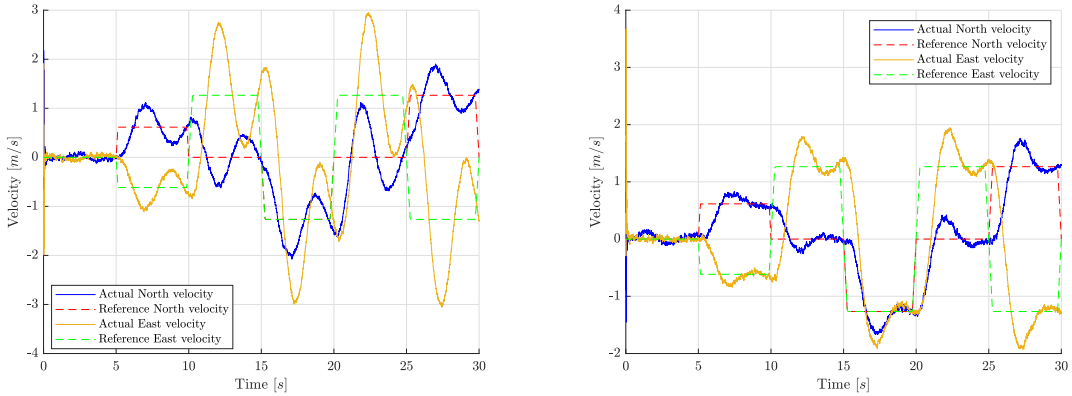
Outer Loop		
Channel	Parameter	Value
Position (North)	K_P	0.02
	K_D	0.3
	K_{DD}	0.1
	$Saturation$	$\pm\frac{\pi}{6}rad$
Position (East)	K_P	0.02
	K_D	0.3
	K_{DD}	0.1
	$Saturation$	$\pm\frac{\pi}{6}rad$

(b) PID Outer Loop PDD parameters

Table 4.1: PID parameters

loop needs a modification to improve dramatically its performance. In fact, looking at Figure 4.4 it is noticeable the high overshoot and settling time of the linear velocity channel when not using a double Derivative action DD compared to the

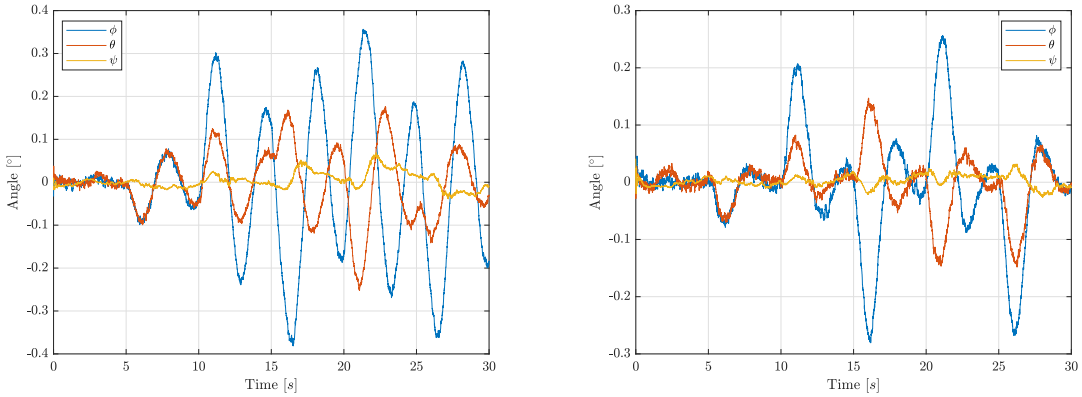
employed controller. Furthermore, the oscillations propagate into the inner loop, as shown in Figure 4.5. This results in bad performance overall (Figure 4.6). Saturation is imposed to avoid infeasible behavior during simulation. In the outer loop the saturation limits how much the quadrotor can tilt around its North-East axes, in this case $\frac{\pi}{6}$, or 30°. Instead, the inner loop the saturation limits the force or torque required to the actuators. The maximum force provided by one actuator is about $8N$, while the torque is about $0.25Nm$, as reported by experimental results (section 4.1). Thus, the total maximum force delivered by the actuators is $32N$, the torque is $1Nm$ and the torque generated around N or E is the force of two actuators times the distance from the N or E axis, that is $3.2Nm$.



(a) Velocity behavior with single D

(b) Velocity behavior with double D

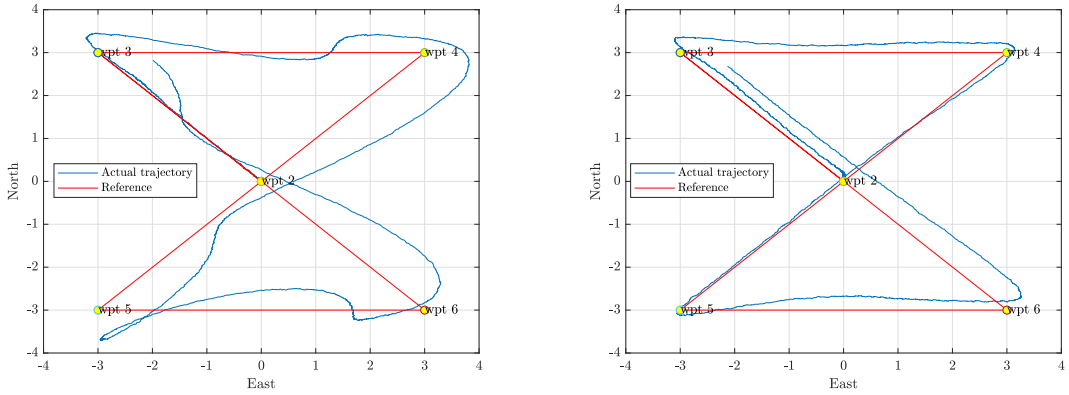
Figure 4.4: Velocity behavior comparison



(a) Attitude with single D

(b) Attitude with double D

Figure 4.5: Attitude behavior comparison



(a) NE plot with single D

(b) NE plot with double D

Figure 4.6: Comparison of waypoint-following performance

Controller – SMC

Table 4.2 gathers the parameters of the SMC solution, with reference to Figure 3.8. Again, the parameters are chosen through the trial and error method. In this case, no double derivative action is needed in the outer loop, as the controller shows good performance in this configuration.

Inner Loop		
Channel	Parameter	Value
Pitch	λ_{pitch}	20
	k_{pitch}	-0.2
	$Saturation$	$\pm 3.2Nm$
Roll	λ_{roll}	20
	k_{roll}	-0.2
	$Saturation$	$\pm 3.2Nm$
Yaw	λ_{yaw}	10
	k_{yaw}	-0.2
	$Saturation$	$\pm 1Nm$
Thrust	λ_{thrust}	40
	k_{thrust}	2
	$Saturation$	$32N$

(a) Inner Loop SMC parameters

Outer Loop		
Channel	Parameter	Value
Position (North)	K_P	0.15
	K_D	0.22
	K	2
	$Sat.\ \phi_{des}, \theta_{des}$	$\pm\frac{\pi}{6}rad$
	$Sat.\ \dot{\phi}_{des}, \dot{\theta}_{des}$	$\pm 1rad/s$
Position (East)	K_P	0.15
	K_D	0.22
	K	2
	$Sat.\ \phi_{des}, \theta_{des}$	$\pm\frac{\pi}{6}rad$
	$Sat.\ \dot{\phi}_{des}, \dot{\theta}_{des}$	$\pm 1rad/s$

(b) SMC Outer Loop PID parameters

Table 4.2: SMC parameters

Sensors model

The sensors are modeled as a block that alter the input signal by adding to it a bias and Gaussian noise. Also, a scale factor and a misalignment matrix are considered. The target multicopter is equipped with an Inertial Measurement Unit (IMU) integrated in the Pixhawk Autopilot, which provides acceleration and angular velocity with respect to the body reference frame and the altitude in the inertial reference frame. Moreover, an indoor localization system is exploited, capable of giving the position and the attitude of the multicopter. Thus, the output to be considered in the simulation is:

$$\mathbf{y}_{(t)} = \mathbf{h}_{(\mathbf{x},t)}^{noiseless} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ p_{gyro} \\ q_{gyro} \\ r_{gyro} \\ h_{baro} \\ p_N \\ p_E \\ h_{Otus} \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{4.1}$$

Therefore, the noisy output $\mathbf{z}_{(t)}$ is:

$$\mathbf{z}_{(t)} = \mathbf{h}_{(\mathbf{x},t)} = \begin{bmatrix} M_{acc} \begin{bmatrix} rv - qw - g\sin\theta \\ -ru + pw + g\sin\phi\cos\theta \\ qu - pv + g\cos\phi\cos\theta + \dfrac{F_z}{m} \end{bmatrix} S_{acc} + \Delta_{acc} + \mathbf{v}_{acc} \\ M_{gyro} \begin{bmatrix} p \\ q \\ r \end{bmatrix} S_{gyro} + \Delta_{gyro} + \mathbf{v}_{gyro} \\ k_B p_0 e^{-\dfrac{gh}{RT_0}} + \Delta_{baro} + v_{baro} \\ M_{Otus}^{pos} \begin{bmatrix} p_N \\ p_E \\ h \end{bmatrix} S_{Otus}^{pos} + \Delta_{Otus}^{pos} + \mathbf{v}_{Otus}^{pos} \\ M_{otus}^{att} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} S_{Otus}^{att} + \Delta_{Otus}^{att} + \mathbf{v}_{Otus}^{att} \end{bmatrix} \tag{4.2}$$

where $\Delta_{(*)}$ is the sensor bias, $\mathbf{v}_{(*)}$ is the sensor noise and $M_{(*)}$ and $S_{(*)}$ are the misalignment and scale factor matrices defined as:

$$M_{(*)} = M_{(*)}^T = \begin{bmatrix} 1 & M_{xy} & M_{xz} \\ M_{yx} & 1 & M_{yz} \\ M_{zx} & M_{zy} & 1 \end{bmatrix} \qquad S_{(*)} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

Extended Kalman Filter

The Extended Kalman Filter (EKF) is a *near-optimal, nonlinear* observer. In this application, the EKF is used to filter the noise coming from the sensors, to fuse redundant signals and as an estimator. However, the EKF works under the assumption that $\mathbf{v}_1$ and $\mathbf{v}_{(*)}$ are white noises with zero mean value and variance V_1 and V_2:

$$\mathbf{v}_1 \sim WN_{(0,V_1)} \qquad \mathbf{v}_{(*)} \sim WN_{(0,V_2)}$$

Consider the continuous-time *nonlinear* mathematical model of the quadrotor presented in Equation 2.13 with its outputs (4.1;4.2) and *process noise* $\mathbf{v}_1$

$$\begin{cases} \dot{\mathbf{x}}_{(t)} = \mathbf{f}_{(\mathbf{x},t)} + \mathbf{g} \cdot \mathbf{u}_{(t)} + \mathbf{v}_1 \\ \mathbf{y}_{(t)} = \mathbf{h}_{(\mathbf{x},t)}^{noiseless} \\ \mathbf{z}_{(t)} = \mathbf{h}_{(\mathbf{x},t)} \end{cases} \tag{4.3}$$

To design a discrete-time EKF the continuous-time system has to be discretized. The method chosen for this application is *"Forward Euler"*, which states:

$$\dot{x} = \frac{x_{(k+1)} - x_{(k)}}{T_s} \tag{4.4}$$

where T_s is the sample time. Applying 4.4 to the state evolution 4.3, one obtains:

$$\mathbf{x}_{(k+1)} = \mathbf{x}_k + T_s(\mathbf{f}_{(\mathbf{x}_{(k)})} + \mathbf{g} \cdot \mathbf{u}_{(k)})$$
$$\mathbf{x}_{(k+1)} = \mathbf{f}_{(\mathbf{x}_{(k)},\mathbf{u}_{(k)})}^{DT}$$

leading to the discrete-time, nonlinear system:

$$\begin{cases} \mathbf{x}_{(k+1)} = \mathbf{f}_{(\mathbf{x}_{(k)},\mathbf{u}_{(k)})}^{DT} + \mathbf{v}_1 \\ \mathbf{y}_{(k)} = \mathbf{h}_{(\mathbf{x}_{(k)})}^{noiseless} \\ \mathbf{z}_{(k)} = \mathbf{h}_{(\mathbf{x}_{(k)})} \end{cases} \tag{4.5}$$

The EKF equations are:

- Initialization:

$$\hat{x}_{(0|0)} \sim \mathcal{N}(\mu_0, \sigma_0)$$
$$P_{(1|0)} = \mathbb{I}_n$$

- Prediction:

$$\hat{x}_{(k|k-1)} = f^{DT}_{(\hat{x}_{(k-1|k-1)}, u_{(k-1)})}$$
$$P_{(k|k-1)} = F_{(k-1)} P_{(k-1|k-1)} F^T_{(k-1)} + V_1$$

- Correction/Filtering:

$$K_{(k)} = P_{(k|k-1)} H^T_{(k)} [H_{(k)} P_{(k|k-1)} H^T_{(k)} + V_2]^{-1}$$
$$e_{(k)} = z_{(k)} - \hat{y}_{(k|k-1)}$$
$$\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K_{(k)} e_{(k)}$$
$$P_{(k|k)} = [\mathbb{I}_n - K_{(k)} H_{(k)}] P_{(k|k-1)}$$

where:

$$F_{(k-1)} = \left.\frac{\partial \mathbf{f}^{DT}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{(k-1|k-1)}} \qquad H_{(k)} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{(k|k)}}$$

and V_1 and V_2 are the covariance matrices of the process and sensor noise respectively such that:

$$V_1 = \mathbb{E}[\mathbf{v}_1 \mathbf{v}_1] \qquad V_2 = \mathbb{E}[\mathbf{v}_{(*)} \mathbf{v}_{(*)}]$$

These quantities are crucial when tuning the EKF. While the sensor noise covariance can be determined by looking at the sensor datasheet or through experimental tests, the process noise covariance is found by a trial and error method. In the case of the considered quadrotor, the noise covariance matrix is found inspecting the log files produced by the autopilot when powered up. The standard deviation is taken directly from plots showing raw sensor data when the quadrotor is disarmed, to avoid the vibrations generated by the motors and the propellers. Instead, V_1 is

determined in simulation. The values of V_1 and V_2 are set equal to:

$$V_1 = \begin{bmatrix} 10^{-6} & & & \\ & 10^{-6} & & \\ & & \ddots & \\ & & & 10^{-6} \end{bmatrix} \tag{4.6}$$

$$V_2 = \begin{bmatrix} diag(\mathbf{v}_{acc}) & & & & & \\ & diag(\mathbf{v}_{gyro}) & & & & \\ & & v_{baro} & & & \\ & & & diag(\mathbf{v}_{Otus}^{pos}) & & \\ & & & & diag(\mathbf{v}_{Otus}^{att}) \end{bmatrix} \tag{4.7}$$

where:

$$\mathbf{v}_{acc} = \begin{bmatrix} 9 \\ 12.25 \\ 4.84 \end{bmatrix} \left(\frac{m}{s^2}\right)^2 \qquad \mathbf{v}_{gyro} = \begin{bmatrix} 0.0305 \\ 0.0076 \\ 0.0049 \end{bmatrix} \left(\frac{rad}{s^2}\right)^2$$

$$\mathbf{v}_{Otus}^{pos} = \begin{bmatrix} 0.0001 \\ 0.0001 \\ 0.0001 \end{bmatrix} m^2 \qquad \mathbf{v}_{Otus}^{att} = \begin{bmatrix} 0.0175 \\ 0.0175 \\ 0.0175 \end{bmatrix} rad \qquad v_{baro} = 0.01 m^2$$

Figure 4.7 shows the block diagram of the implemented EKF.

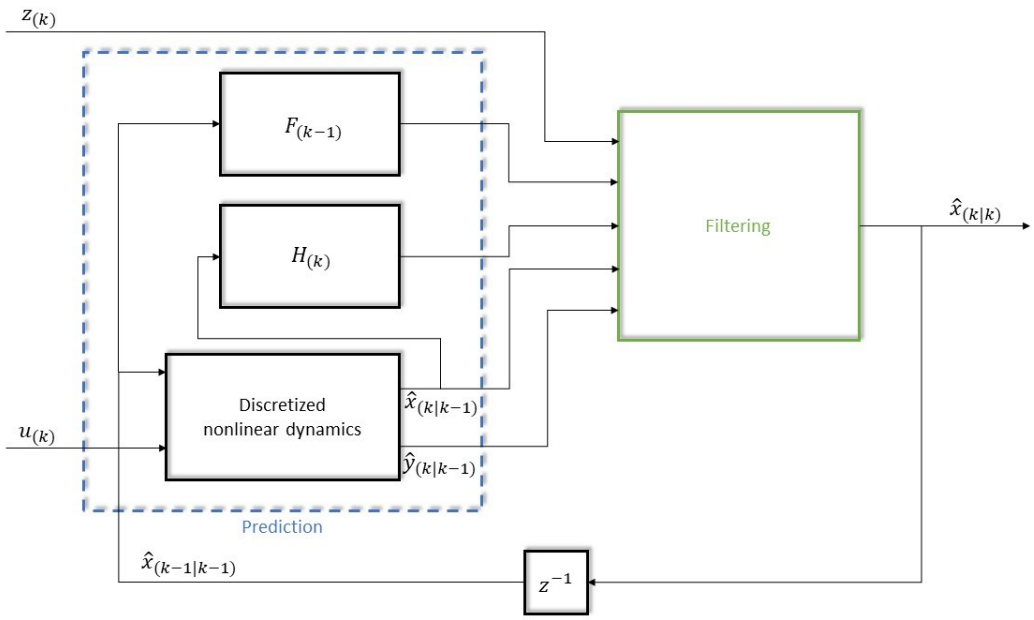


Figure 4.7: Extended Kalman Filter block representation

4.1 Evaluation of the thrust curve

The aim of the thrust tests is to find an analytical relationship between the PWM signal provided to the ESC by the Pixhawk/controller and the thrust the motors can produce. This allows the conversion from a command input expressed as a PWM signal to the physical input vector **u** defined as:

$$\mathbf{u} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

The test bench used is a RCBenchmark Series 1520 Thrust Stand [5] on which it is possible to mount the motor with its propeller. The stand has an extensometer connected to a board to measure the thrust generated. The board provides the signal to the motor ESC and allows connecting the bench to a PC to collect test data. To have a sufficient number of samples, the chosen test is a steps test in

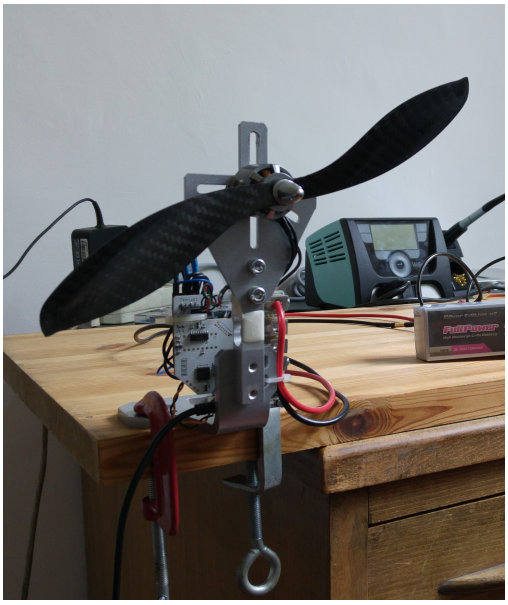
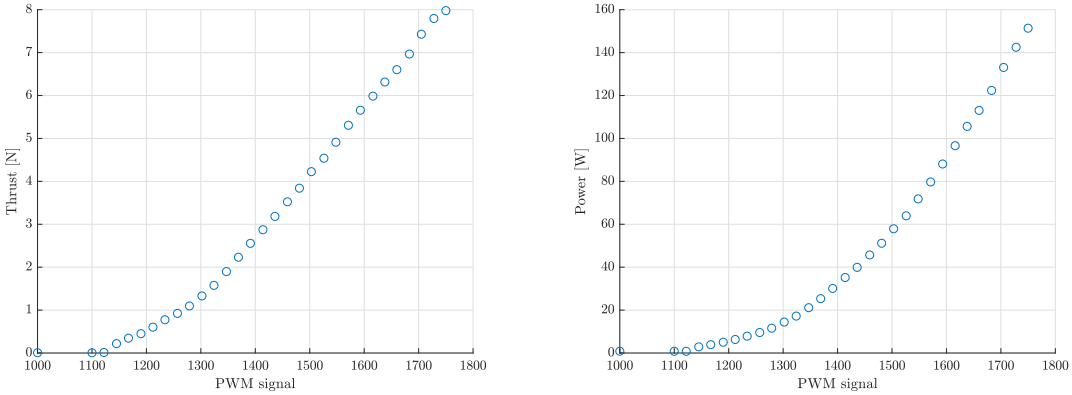


Figure 4.8: Thrust test bench

which the motor varies its velocity between the range $1000 \div 1750$. The behavior of one propeller is shown in Figure 4.9. From data describing the thrust it is possible to find a 2^{nd} degree function fitting the collected points. The obtained data is fitted using a 2^{nd} function for the thrust and a 3^{rd} order function for the power. To obtain



(a) Thrust behavior (b) Power absorption

Figure 4.9: Test results

the moment behavior, one can note that:

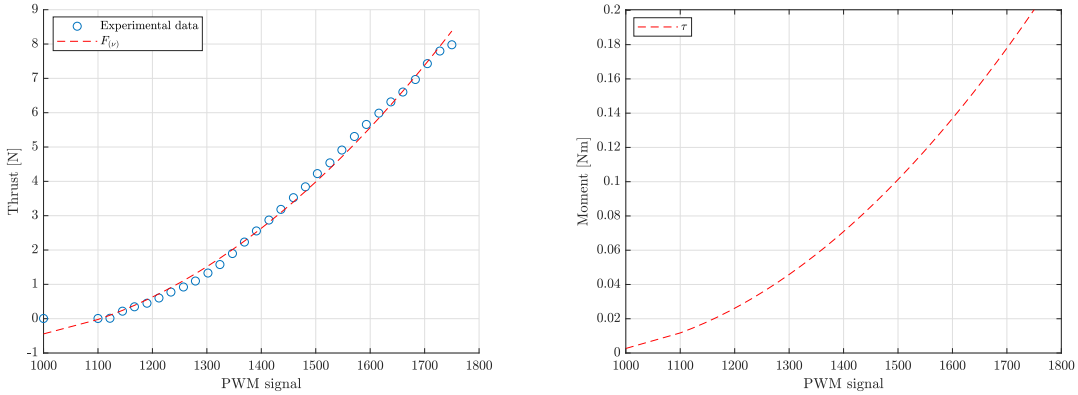
$$P = \tau\omega$$
$$\tau = f_{(\omega)}$$
$$\Rightarrow P = f_{(\omega)}\omega$$

thus, to obtain the moment behavior the power absorption data must be fitted with a 2^{nd} order function. Fitting is performed by means of the Matlab function *"polyfit"* and as a result the thrust F and the moment τ in function of the PWM signal ν are:

$$F_{(\nu)} = 1.1632 \cdot 10^{-5}\nu^2 - 0.0202\nu + 8.1513 \tag{4.8}$$
$$\tau_{(\nu)} = 2.6604 \cdot 10^{-7}\nu^2 - 4.6797 \cdot 10^{-4}\nu + 0.2046 \tag{4.9}$$

Figure 4.10 shows 4.8 overlapping the experimental data and the moment curve described by 4.9:
Lastly, the command input can be rewritten in function of the i^{th} motor PWM signal $\nu^{(i)}$:

$$\mathbf{u}_{(\nu)} = \begin{bmatrix} 0 \\ 0 \\ -(F_{(\nu^{(1)})} + F_{(\nu^{(2)})} + F_{(\nu^{(3)})} + F_{(\nu^{(4)})}) \\ (F_{(\nu^{(2)})} + F_{(\nu^{(3)})} - F_{(\nu^{(1)})} - F_{(\nu^{(4)})})l \\ (F_{(\nu^{(1)})} + F_{(\nu^{(3)})} - F_{(\nu^{(2)})} - F_{(\nu^{(4)})})l \\ \tau_{(\nu^{(3)})} + \tau_{(\nu^{(4)})} - \tau_{(\nu^{(1)})} - \tau_{(\nu^{(2)})} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{4.10}$$



(a) Thrust fitting

(b) Moment curve

Figure 4.10: Thrust and moment behavior

4.2 Model-In-the-Loop simulation

Model-In-the-Loop simulation is performed setting waypoints and (feasible) ToA replicating real world maneuvers. Notice that since the quadrotor is intended for indoor application, the paths have a size comparable to the one of a room. The performance of the candidate controllers are tested individually and then compared. The chosen patterns were used also in [9].

4.2.1 Step signal

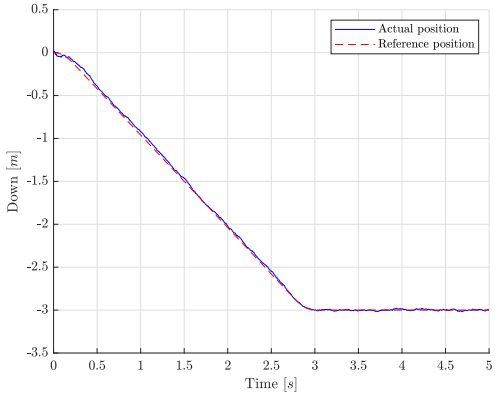
The step signal represent a typical behavior of quadrotors when asked to take off and maintain a given altitude. The quadrotor has to reach an altitude of $3m$ in $3s$.

SMC

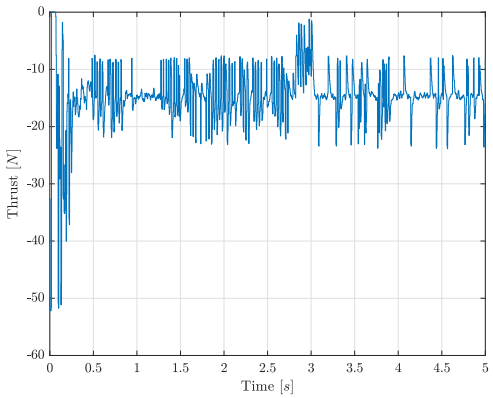
Figure 4.11a shows the response following the reference signal accurately: no overshoot nor steady-state error are present. Nevertheless, Figure 4.11b clearly shows the main drawback of a SMC: the *chattering* is present throughout the maneuver, and solicits the actuators with a high-frequency signal.

PID

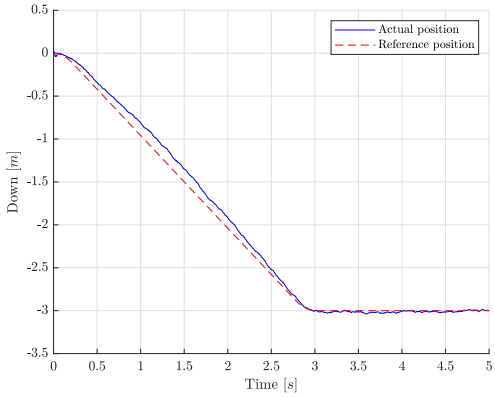
Using a simpler PID, the response to a step reference signal is similar to what happens for the SMC, even though less accurate when tracking position and velocity. The maneuver is completed in the given ToA and no relevant overshoot or steady-state error is present (Figure 4.11c). Furthermore, Figure 4.11d shows that with PID the thrust command exhibits oscillations around the mean value of $\pm 3N$



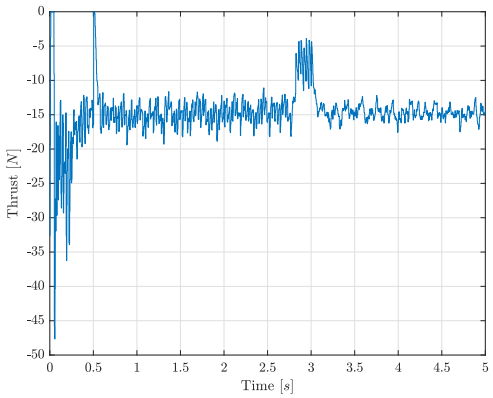
(a) Time response and reference signal (SMC)



(b) Thrust behavior (SMC)



(c) Time response and reference signal (PID)



(d) Thrust behavior (PID)

Figure 4.11: Step response

compared to the $\pm 7N$ of the SMC due to chattering. Comparing the two solutions, it can be seen that in steady-state the SMC exhibits high accuracy, while the PID has a higher but still acceptable steady-state error:

$$e_{\infty}^{SMC} = -2.620 \cdot 10^{-4} m \qquad e_{\infty}^{PID} = 0.0111 m$$

Thus, the SMC is able to provide higher tracking performance compared to the PID controller, yet with a higher command activity.

4.2.2 Square pattern

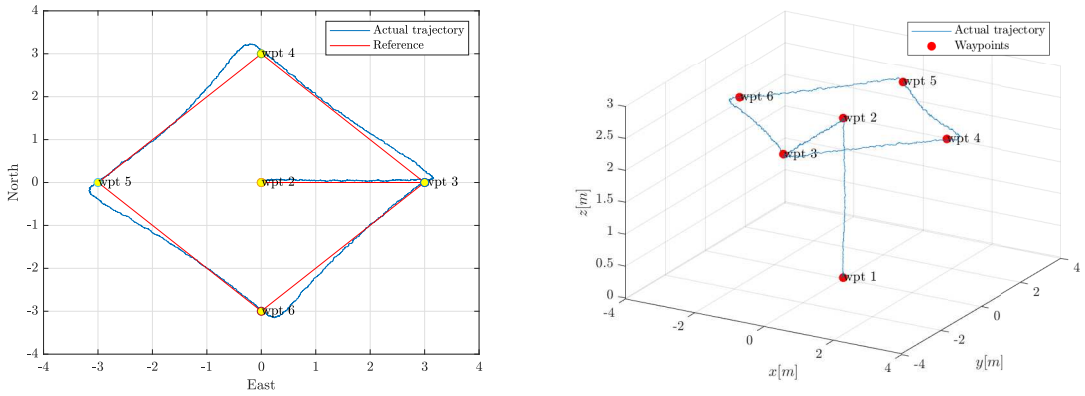
Table 4.3 lists the waypoints provided to the trajectory planner for the square pattern.

	NED coordinates $[m]$	ToA $[s]$
#1	(0,0,0)	0
#2	$(0, 0, -2.5)$	5
#3	$(0, 3, -2.5)$	10
#4	$(3, 0, -2.5)$	15
#5	$(0, -3, -2.5)$	20
#6	$(-3, 0, -2.5)$	25
#7	$(0, 3, -2.5)$	30

Table 4.3: Waypoint list for square pattern

SMC

The square pattern is executed within the given time limits. The SMC exhibits good tracking performance, even if some overshoot is present in the N-E plane (Figure 4.12a). Again, *chattering* is present (Figure 4.13b).



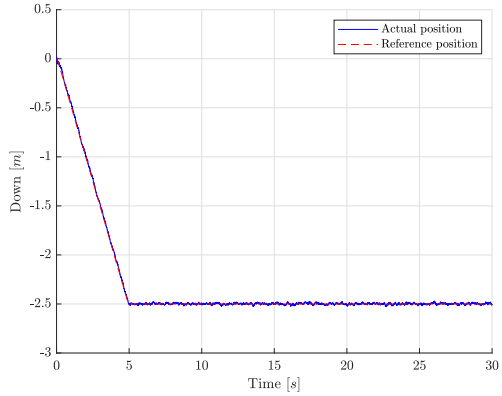
(a) North-East view

(b) XYZ view

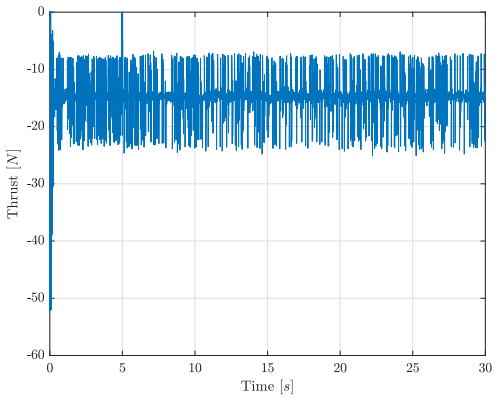
Figure 4.12: Square pattern with waypoints highlighted (SMC)

PID

As can be noticed in Figure 4.14a, the PID controller is not as accurate the SMC and it is not capable of finishing the maneuver in the given time lapse. Figure 4.15 reveals a delay of $0.5s \div 1s$ between the reference trajectory and the simulated maneuver in the NE plane. When inspecting how North-East velocities behave during the maneuver, it can be noticed that PID controller has a higher rise/settling time compared to the SMC (Figure 4.17). For what concerns the altitude control, the PID executes the take off quite accurately and within the time constraints

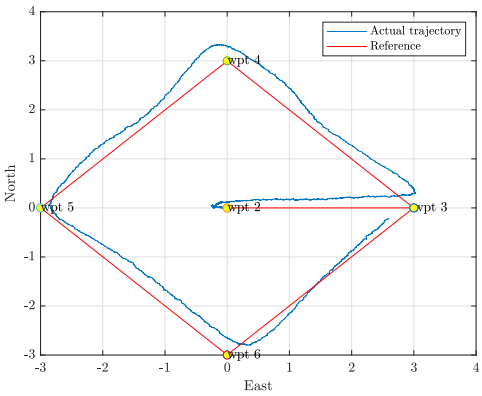


(a) Time response and reference signal in altitude

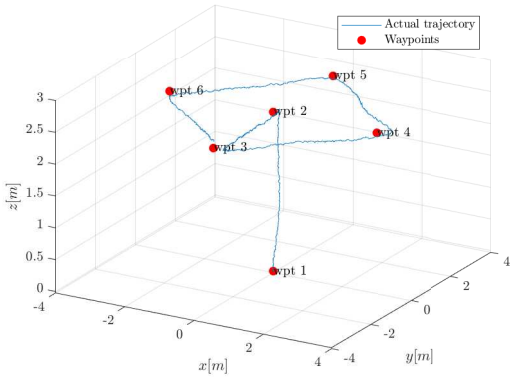


(b) Thrust behavior

Figure 4.13: Altitude response for square pattern (SMC)



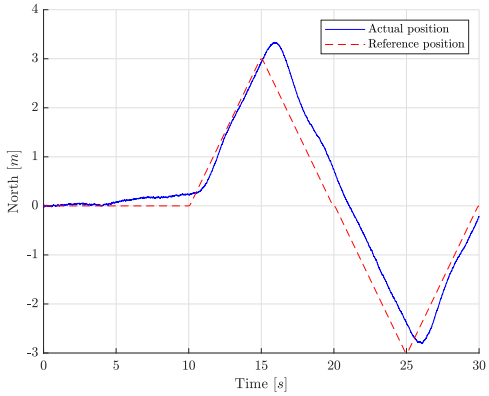
(a) North-East view



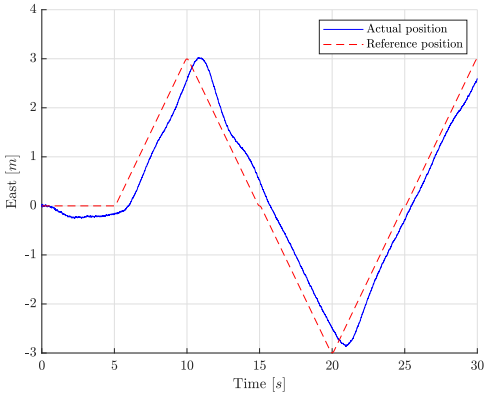
(b) XYZ view

Figure 4.14: Square pattern with waypoints highlighted (PID)

(Figure 4.16a). No large oscillations occur during the simulation (Figure 4.16b).

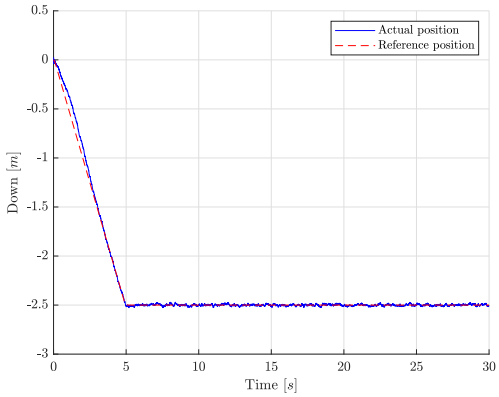


(a) Delay in the North dimension

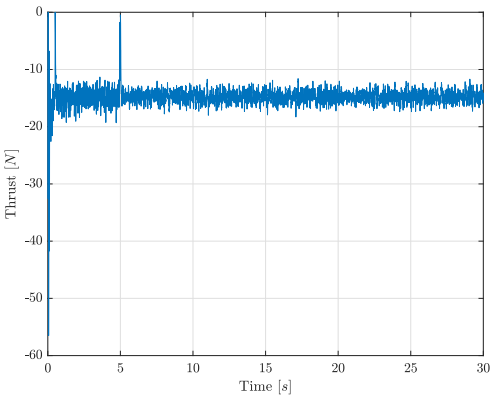


(b) Delay in the East dimension

Figure 4.15: Delay of the trajectory w.r.t. reference (PID)



(a) Time response and reference signal in altitude



(b) Thrust behavior

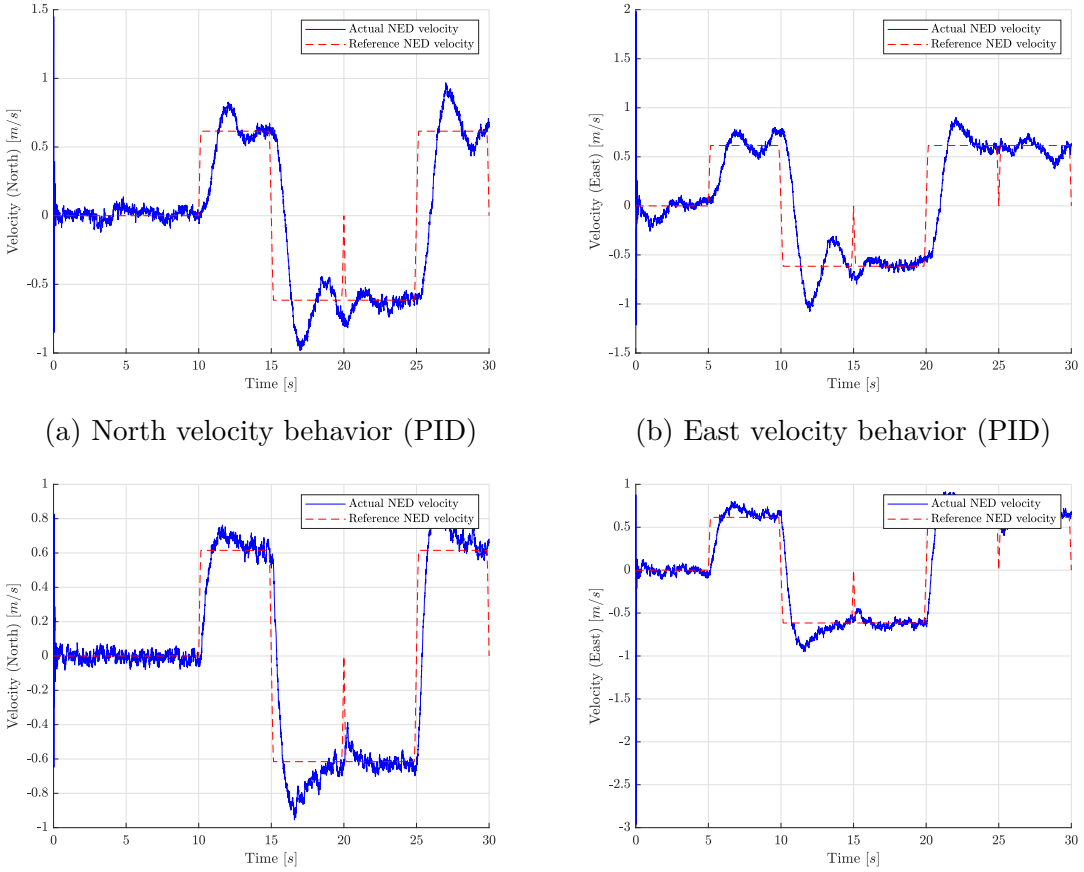
Figure 4.16: Altitude response for square pattern (PID)

4.2.3 Butterfly pattern

Table 4.4 lists the waypoints provided to the trajectory planner for the butterfly pattern.

SMC

As before, the SMC exhibits high tracking performance Figure 4.18a with an overshoot of $0.36m$ due to a too high required deceleration Figure 4.19d.



(a) North velocity behavior (PID)

(b) East velocity behavior (PID)

(c) North velocity behavior (SMC)

(d) East velocity behavior (SMC)

Figure 4.17: North-East velocities comparison for square pattern

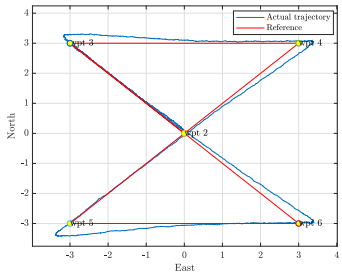
	NED coordinates $[m]$	ToA $[s]$
#1	(0,0,0)	0
#2	$(0, 0, -2.5)$	5
#3	$(3, -3, -2.5)$	10
#4	$(3,3, -2.5)$	15
#5	$(-3, -3, -2.5)$	20
#6	$(-3,3, -2.5)$	25
#7	$(3, -3, -2.5)$	30

Table 4.4: Waypoint list for butterfly pattern

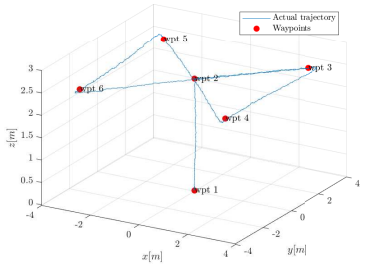
PID

The PID solution proves to be less accurate when tracking and due to collected delay cannot complete the pattern within the given time constraint, being about

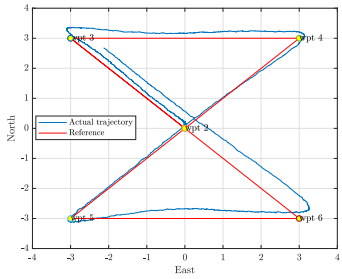
0.85m far from the last waypoint. Figure 4.19 inspects the velocity channel and the attitude behavior during the maneuver. It shows that the SMC solution is faster during attitude changes (Figure 4.19c) and it is able to better follow the reference velocity signal (Figure 4.19d), even though some overshoot is present, which depends on how fast the quadrotor is going during the specific maneuver. On the other hand, the PID controller is less accurate in tracking the reference velocity. Moreover, in Figure 4.19a can be noticed that the inner loop is slower with high rise times and oscillations around the zero. Also, the angle values are smaller compared to the SMC, leading to a less stressing command activity. Figure 4.20 highlights the four-times smaller average command activity of the PID solution compared to the SMC.



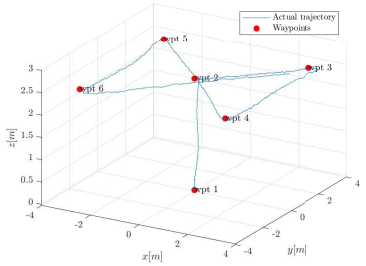
(a) North-East view (SMC)



(b) XYZ view (SMC)

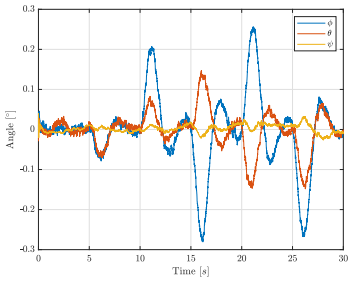


(c) North-East view (PID)

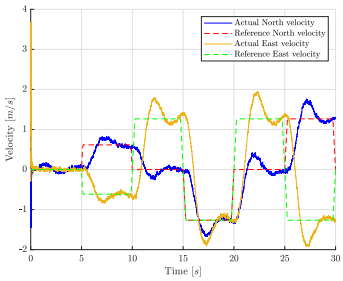


(d) XYZ view (PID)

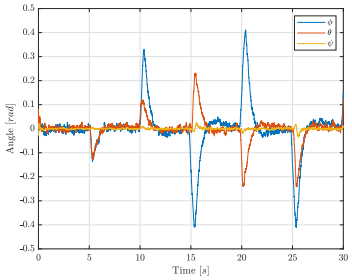
Figure 4.18: Butterfly pattern



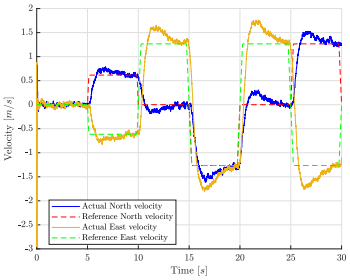
(a) Euler angles during butterfly pattern (PID)



(b) Reference and actual body velocity in NE plane (PID)

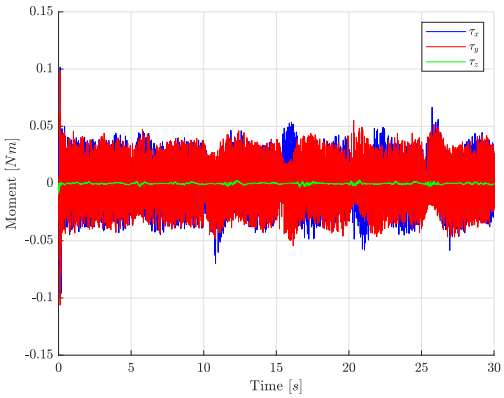


(c) Euler angles during butterfly pattern (SMC)

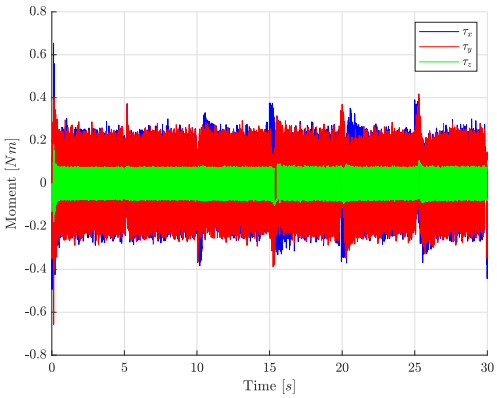


(d) Reference and actual body velocity in NE plane (SMC)

Figure 4.19: Comparison between SMC and PID with the butterfly pattern



(a) PID solution



(b) SMC solution

Figure 4.20: Command activity for the attitude channels

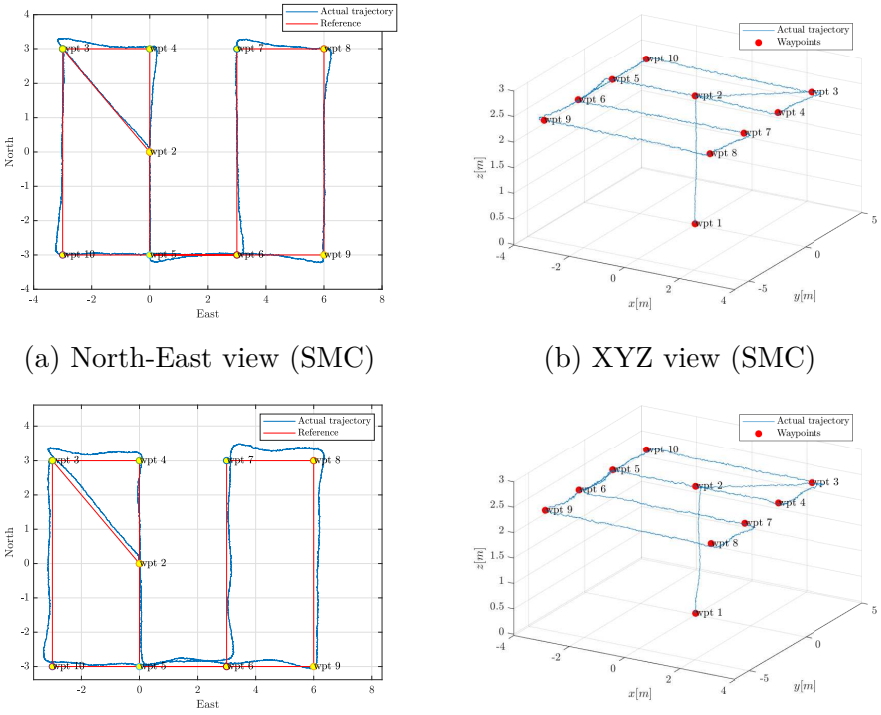
4.2.4 Snake pattern

Table 4.5 lists the waypoints provided to the trajectory planner for the snake pattern. Figure 4.21 shows an overall view of the snake pattern execution, comparing

	NED coordinates $[m]$	ToA $[s]$
#1	(0,0,0)	0
#2	$(0, 0, -2.5)$	5
#3	$(3, -3, -2.5)$	10
#4	$(3,0, -2.5)$	15
#5	$(-3,0, -2.5)$	25
#6	$(-3,3, -2.5)$	30
#7	$(3,3, -2.5)$	40
#8	$(3,6, -2.5)$	45
#9	$(-3,6, -2.5)$	55
#10	$(-3, -3, -2.5)$	70
#11	$(3, -3, -2.5)$	80

Table 4.5: Waypoint list for snake pattern

the two solutions.



(a) North-East view (SMC)

(b) XYZ view (SMC)

(c) North-East view (PID)

(d) XYZ view (PID)

Figure 4.21: Snake pattern

4.2.5 Simulating a payload

In real-world applications, the quadrotor could be required to carry a payload with different masses, without having its controller re-tuned each time it happens. Usually, small quadrotors are not able to provide a high capacity in terms of added mass. In the case of the considered quadrotor, a mass of $0.3kg$ is added is simulation to test and compare how the two solutions react to this kind of change. The payload is assumed to be point-wise and centered in the CoG of the quadrotor, so that inertia moments are unaffected. The tests carried out are a simple take off maneuver to analyze the behavior at steady-state (i.e. in hover) and a square pattern to test the payload influence on the North-East plane.

Step response – $0.3kg$ payload

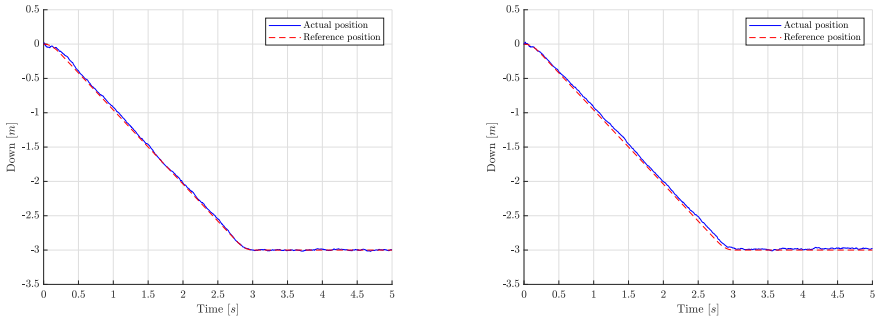
Figure 4.22 shows how the payload affects the performance of the SMC: the controller follows less accurately the reference signal and exhibits a steady-state error higher than the nominal configuration. Note that the SMC controller relies on a dynamical model of the quadrotor which is not updated when the plant (or the real-world prototype) is altered in its properties.

$$(SMC) \qquad e_{\infty}^{payload} = -0.017m \qquad e_{\infty} = -2.620 \cdot 10^{-4}m$$

The very same considerations hold for the PID solution, where the performance is slightly degraded by the presence of the payload. The steady-state errors are:

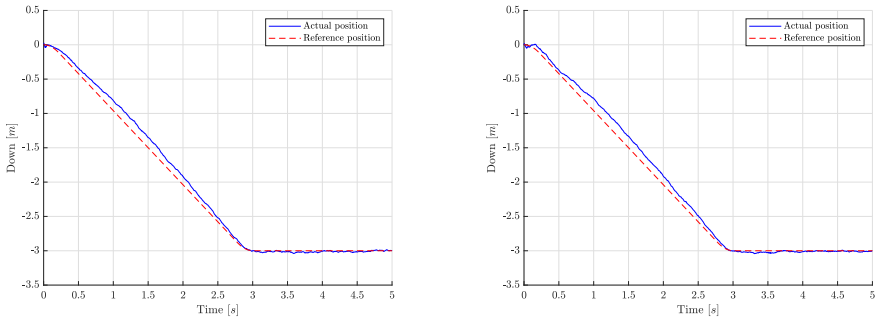
$$(PID) \qquad e_{\infty}^{payload} = 0.0128m \qquad e_{\infty} = 0.0111m$$

Comparing the values of steady-state error, it is clear how the payload affects more the SMC performance in steady-state.



(a) Without payload (b) With a $0.3kg$ payload

Figure 4.22: Step response comparison (SMC)



(a) Without payload (b) With a $0.3kg$ payload

Figure 4.23: Step response comparison (PID)

Chapter 5

Conclusion

In this thesis, two novel control architectures are presented: a PID based controller and a First-Order SMC based controller. Both solutions are designed and tuned in a specific simulation environment that considers not only the well-known dynamical model of a quadrotor but also critical hardware and software components of the prototype DR0N3 quadrotor, i.e. a simple trajectory planner, sensors and EKF. Also, the controller Simulink model is such that it can be used for code generation without heavy modifications, making the entire work "prototype-oriented", i.e. ready for a future deployment on the target Pixhawk hardware. Simulation results showed for different paths a comparison in terms of time-domain performance and accuracy between the two controllers. To improve performance, the outer loop of the PID solution is modified, adding a Derivative action to damp the high oscillations on the velocity channel which would yield to an oscillating inner loop reference signal leading to an overall low performance and accuracy. The SMC solution makes use of both Euler angles and quaternions to generate the command action.
In simulation, it can be noticed how the SMC solution exhibits high nominal performance, executing all the patterns in the given Time of Arrival, showing a smaller steady-state error. On the other hand, the PID solution is much slower and less accurate, and it cannot finish the given maneuver respecting the ToA. Nevertheless, when a parametric uncertainty is introduced in the dynamical system – as a payload or more in general as a measurement error of the inertia properties of the prototype quadorotor – the SMC performance are degraded in terms of accuracy and steady-state error, while the PID shows an almost negligible changing in its behavior. The PID proves to be a slow controller w.r.t. SMC but it is not too influenced by uncertainties. In fact, to preserve the high performance, the SMC should be re-tuned each time the dynamical model varies. A solution could be converting the actual SMC into an *adaptive* one which considers changes occurring to the plant dynamics.

This work is meant to be a preliminary study on an indoor application of a Pixhawk-based quadrotor using the PX4 Flight Stack. Future works are aimed to improve the simulation environment previously presented and to build a customized debug framework to test more in depth the PX4 firmware produced by the code generation process. The first objective can be achieved by better characterizing the quadrotor model. Inertia properties have to be experimentally determined, a model of aerodynamic forces acting on the quadrotor and floor/ceiling effect have to be added. Also, a more complex trajectory planner can be designed, e.g. one employing polynomials or search algorithms on graphs such as Dijkstra's or A*. Also, an adaptive controller can be designed, so that it takes into account critical flight information, e.g. the battery status. The second objective is performing SIL and PIL to test the correctness and the reliability of the generated code. This is done by creating an environment suitable for debugging the customized PX4 as it is builded by the MATLAB®/Simulink® Add-on "PX4 Autopilots Support from Embedded Coder". This would allow to recreate a solid and reliable development and testing process of a quadrotor, or more in general a robot based on the PX4 Flight Stack and Simulink, following the widespread "V-Model" shown in Figure 5.1. Using this

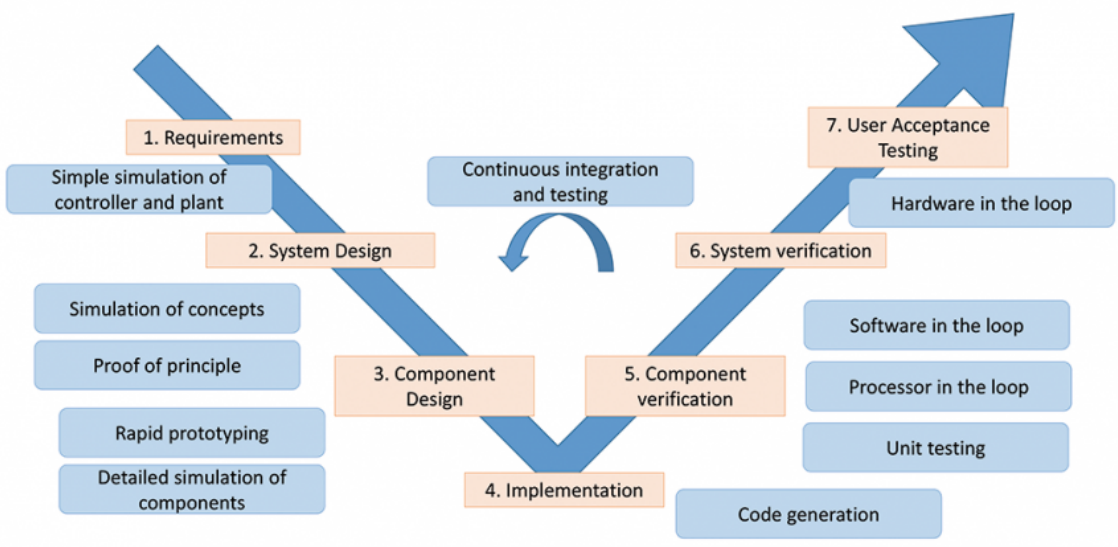


Figure 5.1: V-model used in software engineering

model, a higher integration with the target hardware and a easier issues traceability throughout the involved processes.
Lastly, different positioning sensors can be exploited and compared with the Otus Tracker, such as vision sensors – stereoscopic cameras, lidars and Time-of-Flight cameras or sound based, e.g. ultrasound sensors. The usage of sensors not depending on an external infrastructure is critical as it would allow the quadrotor to fly in

a great variety of scenarios, including non-structured and not known *a priori* environments where the UAV may be required to be able to perform obstacle avoidance maneuvers or a mapping of a unknown or unexplored areas. On the other hand, the drawbacks represent a critical aspect of using vision-based sensors. In fact, they require a higher computational cost that would be moved to an external machine and usually feature a lower positioning precision.

Bibliography

[1] URL: https://optitrack.com/.

[2] URL: http://www.mavtech.eu/it/.

[3] URL: https://pixhawk.org/.

[4] URL: http://ardupilot.org/copter/docs/common-thecube-overview.html.

[5] URL: https://www.rcbenchmark.com/pages/series-1520 (visited on 12/09/2019).

[6] Kanaiya Agrawal and Punit Shrivastav. "Multi-rotors: A revolution in unmanned aerial vehicle". In: ().

[7] Hossein Bolandi et al. "Attitude control of a quadrotor with optimized PID controller". In: *Intelligent Control and Automation* 4.03 (2013), p. 335.

[8] Samir Bouabdallah, Andre Noth, and Roland Siegwart. "PID vs LQ control techniques applied to an indoor micro quadrotor". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2451–2456.

[9] Elisa Capello, Giorgio Guglieri, and Gianluca Ristorto. "Guidance and control algorithms for mini uav autopilots". In: *Aircraft Engineering and Aerospace Technology* 89.1 (2017), pp. 133–144.

[10] Ceren Cömert and Coşku Kasnakoğlu. "Comparing and developing PID and sliding mode controllers for quadrotor". In: *International Journal of Mechanical Engineering and Robotics Research* 6.3 (2017), pp. 194–199.

[11] Nabil Derbel, Jawhar Ghommam, and Quanmin Zhu. *Applications of sliding mode control*. Vol. 79. Springer, 2017, pp. 18–25.

[12] Davide Falanga et al. "The foldable drone: A morphing quadrotor that can squeeze and fly". In: *IEEE Robotics and Automation Letters* 4.2 (2018), pp. 209–216.

[13] Wei Zhong Fum. *Implementation of Simulink controller design on Iris+ quadrotor*. Tech. rep. Naval Postgraduate School Monterey United States, 2015.

[14] Marco Herrera et al. “Sliding mode control: An approach to control a quadrotor”. In: *2015 Asia-Pacific Conference on Computer Aided System Engineering*. IEEE. 2015, pp. 314–319.

[15] Alex Kushleyev et al. “Towards a swarm of agile micro quadrotors”. In: *Autonomous Robots* 35.4 (2013), pp. 287–300.

[16] Gyula Mester. “Cloud robotics model”. In: *Interdisciplinary Description of Complex Systems: INDECS* 13.1 (2015), pp. 1–8.

[17] Ashfaq Ahmad Mian and Wang Daobo. “Modeling and backstepping-based nonlinear control strategy for a 6 DOF quadrotor helicopter”. In: *Chinese Journal of Aeronautics* 21.3 (2008), pp. 261–268.

[18] Simone Panza et al. “Tilt-rotor multivariable attitude control with rotor state feedback”. In: *IFAC-PapersOnLine* 49.17 (2016), pp. 100–105.

[19] Elias Reyes-Valeria et al. “LQR control for a quadrotor using unit quaternions: Modeling and simulation”. In: *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*. IEEE. 2013, pp. 172–178.

[20] Iman Sadeghzadeh et al. “Active fault tolerant control of a quadrotor uav based on gainscheduled pid control”. In: *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE. 2012, pp. 1–4.

[21] Atheer L Salih et al. “Flight PID controller design for a UAV quadrotor”. In: *Scientific research and essays* 5.23 (2010), pp. 3660–3667.

[22] Zhao Shulong et al. “A new feedback linearization LQR control for attitude of quadrotor”. In: *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE. 2014, pp. 1593–1597.

[23] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 199. 1. Prentice hall Englewood Cliffs, NJ, 1991.

[24] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[25] Grzegorz Szafranski and Roman Czyba. “Different approaches of PID control UAV type quadrotor”. In: (2011).

[26] Vadim Utkin. “Variable structure systems with sliding modes”. In: *IEEE Transactions on Automatic control* 22.2 (1977), pp. 212–222.

[27] Holger Voos. “Nonlinear control of a quadrotor micro-UAV using feedback-linearization”. In: *2009 IEEE International Conference on Mechatronics*. IEEE. 2009, pp. 1–6.

[28] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi. "Linear vs. nonlinear control techniques for a quadrotor vehicle". In: *7th International Symposium on Mechatronics and its Applications*. Apr. 2010, pp. 1–10.