Ricardo dos Santos Moreira

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Quadrotor Simulator for Control Development – Application to Autonomous Landing

Dissertação para Obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador:	Professor Doutor Fernando Coito, Professor Auxiliar, FCT-UNL

	Júri
Presidente:	[Nome do orientador], [Cargo], [Instituição]
Arguentes:	[Nome do co-orientador 1], [Cargo], [Instituição]
Vogais:	[Nome do Vogal 1]

Setembro de 2017

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Quadrotor Simulator for Control Development – Application to Autonomous Landing

Copyright © Ricardo dos Santos Moreira, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

To my parents,

Acknowledgments

In the first place, I would like to thank my advisor, Professor Doctor Fernando Coito, for the guidance and patience when silly questions were asked. The knowledge I gain from this work and from his advices are certainly going to help me as an engineer. For that, I thank you.

Thanks to Vasco Brito for helping me to understand some of his work, which contributed to the development of this thesis.

For the best persons I know, a deeply thanks and acknowledgment for my parents, who always gave me support and stood beside me in the good and bad moments. Without them, I would not be where I am now.

To conclude, thanks to Faculdade de Ciências e Tecnologia of Universidade Nova de Lisboa and to Electrical Engineering department, for gaving me the opportunity to meet good collegues and excellent and caring professors. All contributed to my graduation.

Abstract

In this thesis is studied the landing problem of a VTOL UAV and a 3D simulation environment is built to safely develop control for a quadrotor, resorting to 3D modelling and simulation software.

In a time where the development of unmanned vehicles is a trend and it is technologically in growth, the emergent difficulties are challenging when it comes to aviation. In this field, it is useful a tool for researchers to have at their disposal to conduct experiments without putting their real systems to real threat. Also, the landing of UAV's is currently one of the most serious cases of study with a lot of investigation going on to solve the problems associated with it. In this sense, some problematics are contemplated.

Based on a quadrotor in a X8 configuration – 4 frames and 8 propellers –, are applied linear and nonlinear control design techniques with the intent to stabilize and control the quadrotor and a 3D simulator is developed.

Keywords: 3D simulation; landing considerations; quadrotor; unmanned aircraft vehicle; Unreal Engine.

Resumo

São abordados, nesta tese, alguns problemas associados com a aterragem de um veículo do tipo VTOL UAV. Em adição, um ambiente de simulação 3D é construído com o intento de, em segurança, desenvolver controlo a aplicar em quadrotores, recorrendo, assim, a ambientes de modelação e simulação 3D.

Numa época em que o desenvolvimento de veículos não-tripulados é uma tendência e está tecnologicamente em crescimento, as dificuldades emergentes são desafiantes no que concerne à aviação. Nesta área, é como uma mais-valia uma ferramenta ao alcance de investigadores, de forma a que estes possam conduzir as suas experiências sem colocar em risco qualquer instalação física. Em adição, a aterragem de veículos aéreos não-tripulados apresenta-se como um sério caso de estudo, existindo, ainda, bastante investigação a ser conduzida de forma a resolver os problemas associados à mesma. Neste sentido, algumas problemáticas são contempladas.

Baseado num quadrotor em configuração X8 – 4 braços e 8 hélices –, são aplicadas técnicas de controlo linear e não-linear com o intento de estabilizar e controlar o quadrotor. Em adição, um simulador 3D é desenvolvido.

Palavras-chave: simulação 3D; aterragem; quadrotor; veículo aéreo não-tripulado; Unreal Engine.

Nomenclature

Acronyms

2D – Plane representation

3D – Space representation

BLDC – Brushless Direct Current

CPU – Central Processing Unit

GE – Ground Effect

IGE – In Ground Effect

IMU – Inertial Measurement Unit

OGE – Out of Ground Effect

PD – Proportional-Derivative

PI – Proportional-Integral

PID – Proportional-Integral-Derivative

PSO – Particle Swarm Optimization

PWM – Pulse Width Modulation

RC – Remote Control

TF – Transfer Function

UAV – Unmanned Aircraft Vehicle

VTOL – Vertical Take-off and Landing

Symbols

Symbol	Description	Unit
C_d	drag coefficient	adimensional
C_l	lift coefficient	adimensional
F, f	force	N
g	gravity	$m.s^{-2}$
h	height	m
I	body inertia	$kg.m^2$
k	discrete-time variable	$samples.s^{-1}$
L, l	length	m
M	total body mass	kg
r	radius	m
t	continuous-time variable	s
T	thrust	N
T_τ	torque induced by thrust	N.m
T_{PWM}	PWM period	s
x, y, z	translational position	m
ϕ, θ, ψ	Euler angles	rad
α, β, γ	body angles	rad

p, q, r	body axes rate	rad.s^{-1}
τ	Torque	N.m
v	linear speed	m.s^{-1}
ω	angular speed	rad.s^{-1}

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

One of the main reasons why technology has accomplished many and great deeds has to do with wars. Every challenge that stepped in the way of science, were keenly overcame in order for groups of individuals to compete and surpass their foes. Or, in other words, to do harm. Nowadays, and fortunately, the thoughts are settled elsewhere. People are encouraged to thrive so others can benefit from the results and learn more about the planet and universe we live in. And even if the results are not that appealing, at least a piece of knowledge can be extracted and can motivate others to do better.

Aircrafts are one of the wonders born of war. Manned, or even RC ones, can be rather difficult to manoeuvre, even though the CPU is our brain, the actuators our hands and the sensors our eyes. It takes an enormous amount of skill to fly one of these machines and that is why a pilot need so many hours of flight experience before heading to pilot a commercial one, for example. It is even heard, from time to time, that a plane, helicopter or the emergent RC quadrotors have crashed somewhere. But these difficulties are nothing compared to the challenges of letting one aircraft to take-off from a specific location and landing on another, in complete autonomy. These are called UAV and are most likely the future of private and commercial flights.

The UAV's are currently used for scouting, mapping, leisure, professional photography and others, but can have a key role in the near future, as such in rescue missions, environmental monitoring, terrain analysis, infrastructures inspection, and much more.

Despite some important achievements concerning attitude and trajectory control, the landing control is still an issue and is one of the most vital systems in an UAV. Also, to prevent any type of accidents, which can be disastrous, a 3D simulation environment for a quadrotor is developed, which is the main contribution of this work.

1.2 Research objective and main contributions

This research purpose is to discuss the landing problems of UAV's and develop a 3D simulation environment where this and other flight problems may be tested in safety.

The main goal of this thesis is to design a simulation environment with the intent to perform several tests concerning the landing of a UAV. For this purpose, a kinematics model of the aircraft has to be considered. As such, a specific quadrotor is studied and controllers are developed on Matlab software, with the respective results presented at Chapter 4. The simulator will only include the quadrotor model, but it will be possible to manually control it, with the prospective to receive the input commands from an external controller or one to be built-in.

The main contributions are the 3D simulation environment, which is expected to recreate a physical one, where the quadcopter 3d model and the surrounded environment are furnished with a set of features as similar as possible to real physics. Furthermore, a quadrotor model is proposed and determined if the designed controllers are able to provide the necessary stability and performance to execute a successful flight.

1.3 Thesis Structure

This thesis is structured in 6 chapters, which are organized as follows:

Introduction: In this chapter, are presented the motivation, the objectives and the main contributions to unmanned aviation. Also, the thesis structure is described.

State of the Art: In this chapter, the most relevant background information that supports this thesis is provided. Are referred the landing problems, the quadrotor for which the control is to be developed, an analysis on computer graphics software, control design techniques and finally the related work.

Quadrotor Dynamics and Control: Here is presented the quadrotor model used as starting point for this project, followed by improvements for a better navigation. Additionally, each motor TF is considered, problems associated with landing are considered and, to complete this chapter, are designed the controllers to be implemented on the closed-loop system.

Simulation and Results: This chapter intent is to validate the efficiency of the control architecture in the proposed model. Also, tests are conducted on certain physics aspects to understand their influence in a real landing manoeuvre.

Virtual Environment: In this chapter, an explanation about the choice of the 3D modelling and game development tools is given, along with a brief discussion of the steps to follow, so anyone can use these tools.

2 State of the Art

2.1 Introduction

In this chapter, it is presented the literature and scientific surveys of the areas of study used as milestones for this work. A briefly insight into some main problems in the aeronautics field is given, as well as their influence on landing phase of flight.

Following the problems associated with landing, the quadrotor kinematics model in which this thesis is grounded is presented. This model is a simplified version and does not considers several physics phenomena that has impact on landing approach.

The next subject consists of a 3D software assessment. The goal is to develop a simulation environment where tests can be conducted without damaging the physical structure and to be an interactive way to learn by playing. Firstly, are analysed software responsible for 3D modelling, so a 3D representation of the quadrotor can be constructed. Secondly, game engines to animate and describe the physics of the quadrotor, apply disturbances and create a scenery.

Afterwards, are specified the techniques to be applied, so a satisfactory control over the aircraft can be provided. The main steps to land a VTOL are to acquire the target, stabilize the aircraft, analyse the target surface and execute the right control actions for a perfect landing.

To end this chapter, it is presented the ground work for this thesis constituted by a few researches conducted in past recent years.

2.2 The Landing Conundrum

Many are the effects that can cause a quadrotor to experience some difficulties. Since the first flight attempts that inventors and researchers are trying to nullify these. The most relevant ones are explained below.

2.2.1 Weather effect

Weather plays a significant role in every step of an aircraft navigation, from take-off to landing. For instance, changes in temperature lead to a variation of air density, which in turn leads to changes of air pressure. Ultimately, due to these events, air currents are formed. Altitude is also relevant, because the higher the aircraft, lesser the pressure and thinner the air, causing the aircraft to experience stability issues. These phenomena alters both the stall speed and minimum flying speed necessary for any aircraft to take-off or land, respectively (Federal Aviation Administration 2016).

Air flows, also known as winds, can interfere greatly on flight control. It can be a major setback on the quadrotor normal operation mode, because not only the quadrotor tends to deviate in a randomly way, but without a sufficiently robust controller it might crash.

Wind's pattern and formation are primarily due to changes of pressures. These variations are driven by distinct types of events. Three of which are the atmospheric pressure, the Coriolis effect and even the topography. As a result, it is of an utmost difficulty to write down a mathematical notation to characterize this phenomenon. However, the impact of it over an object can be measured. A way to represent this is through eq. (2.2.1) and eq. (2.2.2).

$$F = P \cdot A \cdot C_d \tag{2.2.1}$$

$$F = \frac{1}{2} \cdot \rho \cdot V^2 \cdot A \cdot C_d \tag{2.2.2}$$

Where F is the drag force, or wind load, P [Pa] is the wind pressure, A [m^2] is the area section of the object where the force is being exerted, ρ [kg·m^{-3}] is the air density, V [m·s^{-1}] is the speed of the body relative to the air flow and C_d is the drag coefficient.

2.2.2 Effect of Obstructions on Wind

As denoted in the previous subsection, wind is very unpredictable in each time instant, if taking solely into account natural causes. This condition might be aggravated if structures are near and on wind's side, wherein forms more turbulence and the changes of wind direction become even more random.

Manmade constructions, e.g., buildings, ships superstructure, bridges and amongst others, can create air pressures that burst in several directions and differently from natural deformations. Buildings, for instance, cut the wind and on the opposite side turbulence appears with air currents flowing in many indistinct directions. In contrast, when the structure is a mountain, for example, and the wind is flowing up the windward side of the mountain, the currents tend to point downwards on the leeward side of the mountain (Federal Aviation Administration 2016).

2.2.3 Ground Effect

When a VTOL aircraft is about to land, it experiences some undesirable phenomena due to ground effect. This happens when the air mass generated by the rotor blades is reflected by the surface, thus creating an air cushion, which is an air pressure on the lower side of the aircraft. This airflow can provide more thrust, leading to an increase of efficiency of the rotors. The main consequences of this principle includes vibration, which can lead to irreversible instability of the aircraft, altitude fluctuations (Davis and Pounds 2016; Sharf et al. 2014; Aich et al. 2014), and possible bounce after touching a rigid surface (ArduPilot Dev Team 2016).

A work conducted by Cheeseman and Beckett produced a first mathematical description of this effect on the lift of a helicopter rotor at different forward speeds. The simplest situation, and the most important one to consider in this thesis, occurs when the rotor is rotating at constant power, zero air speed and zero forward speed. Thus, the thrust ratio between the thrust IGE and OGE is dependent of the rotor radius and the propellers distance away from the ground. The mathematical expression is given by eq. (2.2.3) (Cheeseman and Bennett 1957).

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-\frac{r^2}{16z^2}} \tag{2.2.3}$$

T_{IGE} [N] is the rotor thrust under the influence of the air cushion, T_{OGE} [N] is the rotor thrust away from the ground, r is the rotor radius and z [m] is the distance of the propeller from the ground.

Another situation occurs when the aircraft is moving parallel to a surface and at constant power. In this case, both aircraft speed v_A and induced speed v_i at the rotor appears in the eq. (2.2.4).

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-\frac{r^2}{16z^2\left(1+\left(\frac{v_A}{v_i}\right)^2\right)}} \tag{2.2.4}$$

2.3 Quadrotor

Back in 2016, Vasco Silva developed a quadrotor with 4 frames and 8 thrusters, 2 in each frame, whose mathematical model contemplates several physical properties like gravity, gyroscopic effect and the overall force produced by the motors. In addition, it has a control scheme for failure detection. (Brito 2016).

The real structure is based on the DJI Flamewheel 450 and some components were especially made in a 3D printer in order to accommodate the eight motors. The control unit comprises an Arduino Due, wherein lies all the programmable logic to control the quadrotor (Brito 2016). Amongst other features, it is equipped with GPS, absolute orientation and altiMU sensors, allowing a good data acquisition for reliable information about the quadrotor positioning, attitude, and others.

For these reasons and since it is in a very early stage, it is an interesting challenge to continue the legacy of a former colleague. Figure 2.1 represents the quadrotor assembled by Brito.

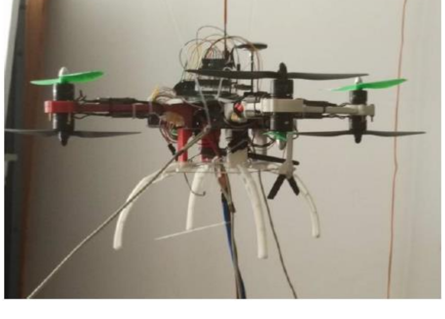


Figure 2.1: Vasco Brito's quadrotor. Retrieved from (Brito 2016).

2.4 3D Modelling Software

In this section, are presented three 3D modelling software, 3ds Max, Maya and Blender, and the respective attributes for further review.

2.4.1 3ds Max

This software property of Autodesk is commonly used in the industry for 3D computer graphics development. It provides the necessary tools for modelling, animation, simulation and rendering, supporting the creation of films and games.

Autodesk has several programs prone to similar purposes, like Maya, which is the next program approached, with little differences between these and 3ds Max (Autodesk 2016). In a more intrinsic view, 3ds Max is considered not very user friendly as the user interface is not quite intuitive, with a learning curve a bit steep, especially for new developers in the field (Tay 2014). The simulation tools are slightly complex even for people with experience and its own scripting language, MAXScript, is not straightforward. Despite these not positive features, it offers an ease of use Material Editor and a rich set of tools essential for modelling. It is possible to import or export FBX files, which are widely used and therefore is compatible with many other 3D software (Yang 2016). It also has many plugins at disposal and, as well as other software from Autodesk, the full version is paid. Nevertheless, a student version is available and is free for three years with almost the same features as the paid one (Yang 2016).

2.4.2 Maya

Here is another program from Autodesk with similar features as the previous 3ds Max. Maya stands out for its animation and effects tools, but lacks in what concerns to modelling, contrasting with 3ds Max. It is not very intuitive and the creation and handling of materials along with professional computer animation and simulation is generally complex to develop, in part due to the need to use some programming (Carrasquinho 2015). In this matter, Maya scripts can be written in Python or in its own programming language, MEL. Like 3ds Max, there are many plugins and add-ins to support the 3D development. The software is paid but it can be acquired, with less features, with a student licence, free for three years.

2.4.3 Blender

Blender is a 3D computer graphics software from Blender Foundation, mostly used by artists and small companies in this area (blender.org 2015). It supports the tools for modelling, animation, simulation and rendering, as so does the previous software mentioned, but can also be used as a game engine, although this is not its best feature. The major advantage of using Blender lies in being an open source software, therefore it is free. There are also several tutorials and documentation and occasionally it is updated with the help of contributions provided by the community. The most significant setback using this software are software faults (commonly referred as "bugs") that appear each time an update happens to fix other bugs. Additionally, the tutorials, documentation and other types of support are not up-to-date, even though the software is (Carrasquinho 2015; Supernat 2012).

Regarding the technical aspects, it has all features available and the only weak element is the user interface, which is little intuitive with a relatively harsh learning curve (Carrasquinho 2015).

2.5 Game Engines

After a 3D model is complete, it must be imported to a simulation software in order to conduct tests or simply to play and enjoy. Ahead, is available the description of two game engines commonly used in the game industry.

2.5.1 Unity

Unity, also known as Unity 3D, is a software property of Unity Technologies and is one of the most popular game engines, commonly known for being intuitive and proper for beginners. It has a vast set of tools and a very complete asset store to help in the projects development. It is compatible with many 3D simulation and modelling software as it can read several file formats. The projects can be developed in a node editor written mainly in C-sharp or Javascript and exported into several file formats as well. Unity is a paid software but a free version is available for the common user. However, the free version has a lot less features in comparison with the paid one.

2.5.2 Unreal

Unreal Engine is a software developed by Epic Games and is considered one of the best game engines in the market due to its remarkable graphical capabilities (Mayden 2014). In comparison with Unity engine, it has also a lot of powerful features and tools but perhaps the most differentiation aspect is the Blueprint visual scripting. This feature is a node-based scripting editor, providing the ability to create equations using blocks diagram. Notwithstanding, one can also write code in C++ (Epic Games 2017).

Unity has a larger store asset than Unreal, with the Unreal one to be reasonable regardless. In Unity, is also possible to import / export projects within a vast range of file formats, whilst Unreal supports only FBX format. A major factor is the pricing issue and Unreal Engine 4 leads in this subject as it is free with all its features accessible by anyone, if not used for commercial activities. Otherwise, royalties must be paid to Epic Games.

2.6 Control Design Techniques

In this chapter are presented the control techniques used in this thesis with the purpose to incorporate them in a quadrotor model. The first method described is the classical PID controller.

2.6.1 PID control

One of the most commonly used control techniques is the classical Proportional-Integrative-Derivative algorithm. This method appeared in the 1920's by Minorsky while observing the way a helmsman steered a ship. It was then improved and applied during the following decade in pneumatic industry and in 1942 John G. Ziegler and Nathaniel B. Nichols developed the well-known tuning rules to find the optimum parameters of a PID controller, given certain constraints (Bennett 1996). Presently, this type of controller is commonly used in industry, offering reliable results for most industrial processes.

The generic PID control algorithm assumes the following form:

$$u(t) = K \cdot \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \cdot \frac{de(t)}{dt} \right) \qquad (2.6.1)$$

In the previous eq. (2.6.1), u is the control action, e the error between the reference and output signals of the process, K the proportional gain, T_i [s] the integral time and T_d [s] the derivative time.

2.6.1.1 Proportional Action

The simplest form of a PID controller occurs when the integral and derivative actions of eq. (2.6.1) are cancelled, thus leaving a pure proportional controller given by eq. (2.6.2).

$$u_p(t) = K \cdot e(t) + b \tag{2.6.2}$$

Here appears a new variable b, which stands for the reset or bias. When control error equals zero, $u_p = b$. This factor acts like a disturbance and can be manually adjusted so that the stationary control error equals zero at a specific condition of operation (K. Astrom 1995).

A high value of the proportional gain can lead to accentuated oscillations of the process output without cancelling the stationary control error. Therefore, the integral action is introduced (K. Astrom 1995).

2.6.1.2 Integral Action

The integral action main purpose is to nullify the control error in stationary state, when a variation of the proportional gain by itself is not enough. This integral effect is represented by eq. (2.6.3).

$$u_i(t) = K \cdot \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau \tag{2.6.3}$$

When the control error is positive, the control action increases to compensate the low value of the process output. When it is negative, the control action decreases so the process output decreases as well and follows the reference value.

A PI controller type is able to effectively nullify the control error in steady state, but it might need more time than the desirable to do so due to present and long-lasting oscillations (K. Astrom 1995).

2.6.1.3 Derivative Action

An integral effect provides a prior knowledge of the system past states, but it can't predict how it is going to behave, leading to possible underdamping. A derivative action is able to predict the next process outputs through the tangent to the error curve, decreasing the oscillations and thus increase the stability of the closed-loop system (K. Astrom 1995). The eq. (2.6.4) represents this action.

$$u_d(t) = K \cdot T_d \cdot \frac{de(t)}{dt} \tag{2.6.4}$$

The combination of the three aforementioned actions constitutes a classical PID controller.

2.6.2 Adaptive Control

The PID algorithm proves to be a good and practical method to solve many cases of industrial processes. However, some processes parameters are unknown or vary unpredictably in time, which can pose a threat to system stability. To counter that, the controller parameters should be adjusted dynamically, which is the main focus behind adaptive control theory (Landau et al. 2011).

An adaptive control consists in the capture of a system's dynamics and specification of the control-design algorithm, along with a fit controller design method for an estimation on-line of the controller's parameters. This type of control is therefore inherently nonlinear and has several applications regarding both linear and nonlinear systems (Landau et al. 2011; K. J. Astrom and Wittenmark 1996).

Applications for this control technique are found on multirotors for attitude stabilization (Zairi and Hazry 2011), trajectory control (Santos et al. 2017) or general control (Buyukkabasakal et al. 2015). On the first two works, artificial neural networks are included to improve precision and minimize control errors.

2.6.3 Optimal Control

Another control method with particular interest is the optimal control. The Optimal Control Theory is an extension of the calculus of variations which intends to minimize or maximize a given functional subject to constraints. In control problems, the functional is usually a cost function, or minimization function, subject to constraints, which is intended to be minimized. The founders of this theory are Bellman and Pontryagin, providing solutions to stochastic and deterministic problems, respectively (Todorov 2006).

When the goal is to achieve optimal solutions, optimal estimation is commonly used due to sensor noises and delays and because the two problems are dual, meaning that one is closely related to another via control and filtering equations, respectively (Todorov 2006).

2.6.3.1 PSO

The Particle Swarm Optimization is a method based on the collective intelligence and first proposed by Russel Eberhart and James Kennedy in 1995. This method intent is to optimize continuous nonlinear functions (Kennedy and Eberhart 1995). It has several applications, one of which to obtain a controller's parameters for a given nonlinear system.

One version of the PSO method consists in the achievement of the best value considering all the particles in the swarm, where each particle is a solution of the system and can be represented in a Cartesian system with any dimensions. The respective algorithm for one-dimension solution is hence described.

The first step is to create particles in random positions and with random velocities. Secondly, with these particles, apply the desired minimization function and calculate its value. This value is thus compared with the current particle's best value (p_{best}). If the result is positive, this value is now considered the particle's best value and is compared with the group's best value (g_{best}). Again, if it is true, this *pbest* is now equal to *gbest*. The change of speed and position of each particle on each axis is given by eq. (2.6.5) and eq. (2.6.6), calculated in this specific order. This process restarts on the particle evaluation and the loop is repeated. (Eberhart and Kennedy 1995).

$$s_i = s_i + c_1 \cdot rand \cdot (p_{best} - p_i) + \\ + c_2 \cdot rand \cdot (g_{best} - p_i) \tag{2.6.5}$$

$$p_i = p_i + s_i \tag{2.6.6}$$

On the previous equations, s_i and p_i stands for speed and position of the i--th particle, respectively, c_1 and c_2 are the adjustment coefficients and $rand$ is a random value in the range [0;1].

2.7 Related Work

To understand in what way this thesis could contribute to science, a survey was conducted and some researches related to autonomous landing of multi-rotors were found. Some of the most significant are referred below.

There are several studies conducted in this field and most uses image processing or optical flow to determine the relative position of a target. A work developed by Lee, Ryan and Kim in 2012 consisted in using image-based visual servoing (IBVS) algorithm to locate the target and get its velocity relatively to the quadrotor. Their main contributions are image processing in a 2D instead of a 3D representation, thus decreasing computational calculations and complexity, and an adaptive SMC – Sliding Mode Control – control regarding landing step for precision control as well for GE compensation. An IMU is also used to provide information about the quadrotor attitude and Lyapunov Stability Theory to stabilize the quadrotor during landing procedure (Lee, Ryan, and Kim 2012). The Figure 2.2 shows the quadrotor and target used for the experiments.

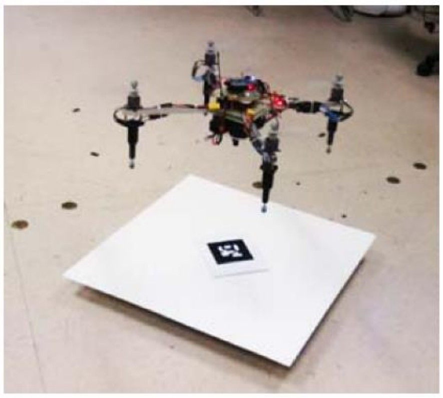


Figure 2.2: Quadrotor and moving target (Lee, Ryan, and Kim 2012).

In the same year, Hérissé, Hamel, Mahony, and Russotto improved their earlier work from 2010 on the use of optical flow as hovering and landing control on a moving platform. This system is composed by a camera, to capture the visual motion, or optical flow, and by a IMU, to determine the attitude and linear position of the UAV (Herissé et al. 2012).

The control schemes used in this project includes nonlinear PI control, optical-flow based control, time-to-contact based control, Lyapunov Stability Theory and a guidance and control approach (Herissé et al. 2012). The quadrotor used for testing is shown in Figure 2.3.

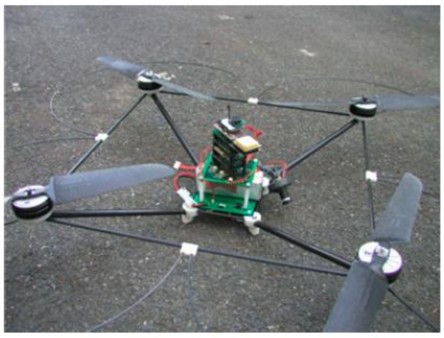


Figure 2.3: Quadrotor used for experimental results (Herissé et al. 2012).

A more recent project dates from 2016 and it was written by Serra, Cunha, Hamel, Cabecinhas and Silvestre. Their main contribution to the landing problematics over a moving surface is the use of a dynamic IBVS control to detect the target amongst noise and an optical flow measurement to detect surface movement (Serra et al. 2016).

In their paper, an innovative IBVS control is proposed, and other control techniques are approached like optical flow control, Lyapunov Stability Theory and a cascade control architecture. An IMU is also used. The setup is shown in Figure 2.4.

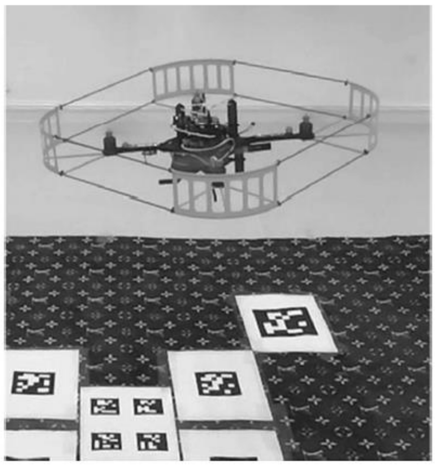


Figure 2.4: Quadrotor used in experimental setup (Serra et al. 2016)

3 Quadrotor Dynamics and Control

In this chapter are presented both the kinematics model used by Brito's work and an extended version, respectively. Some experiments conducted on Brito's work are considered and physics intrinsically related to landing are characterized. At the end, control architectures are presented along with techniques for the estimation of the controllers' parameters.

3.1 Simplified Model

The core of Vasco Brito work was the development of a system tolerant against motor failures, thus some considerations being wittingly neglected in the model proposed. From Brito's work, the focus is entirely on the model and aircraft dimensions, improving the first and assuming values for unknown parameters to face several physics properties regarding the aircraft landing.

3.1.1 Model Parameters

In this subsection, the model parameters are described, with a similar notation from the one adopted by Brito in his thesis.

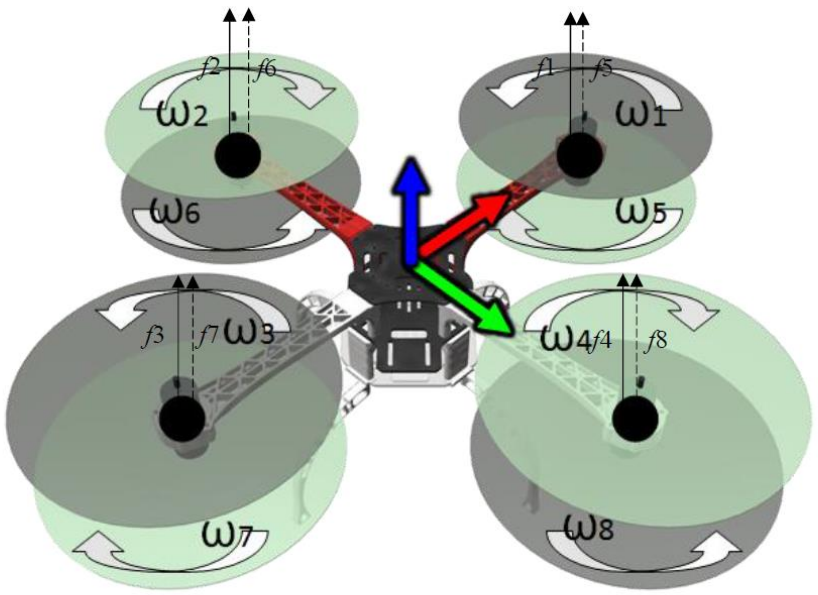


Figure 3.1: Representation of the quadrotor steady hovering. XYZ axes are represented by RGB arrows, respectively. Retrieved from (Brito 2016).

On the Figure 3.1 illustrated above, are identified the angular speed ω_n and the respective force f_n produced by each rotor n. Particularly, this image illustrates a hovering flight, since all forces have equal magnitude. In addition, controlling each rotor independently enables the control of each of the three fundamental rotations: roll, pitch and yaw. The altitude and angular position control actuations are, therefore, mathematically represented by eq. (3.1.1).

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{8} f_n \\ f_4 + f_8 - f_2 - f_6 \\ f_3 + f_7 - f_1 - f_5 \\ f_1 + f_3 + f_6 + f_8 - f_2 - f_4 - f_5 - f_7 \end{bmatrix} \tag{3.1.1}$$

The roll, pitch and yaw are rotations about the longitudinal, transverse and vertical axes, respectively. Each rotation represents a rate of change of angular position. The combined three-dimensional angular positions are henceforth named attitude. The attitude and the translational position matrices are represented by eq. (3.1.2) and (3.1.3), correspondingly.

$$\mathbf{\Theta}_{\boldsymbol{E}} = [\phi \quad \theta \quad \psi]^T \tag{3.1.2}$$

$$\mathbf{P} = [x \quad y \quad z]^T \tag{3.1.3}$$

The first matrix refers to Euler angles and the second to the position seen from an inertial observer.

On Brito's work, some of the aircraft parameters are explicitly referred, but others must be assumed to develop the control system. For detailed documentation on the quadrotor inertial and geometric parameters see Attachment A.

3.1.2 Open-Loop system

Finally, it is presented the kinematics model represented by eq. (3.1.4), whose mathematical deduction and the model itself are found on Brito's thesis (Brito 2016).

$$\begin{cases} \ddot{x} = (\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta))\frac{U_1}{M} \\ \ddot{y} = (\cos(\psi)\sin(\phi) - \cos(\phi)\sin(\psi)\sin(\theta))\frac{U_1}{M} \\ \ddot{z} = -g + \cos(\phi)\cos(\theta)\frac{U_1}{M} \\ \ddot{\phi} = \left((I_{yy} - I_{zz})\dot{\theta}\dot{\psi} - J_r\dot{\theta}\Omega_r + LU_2\right)\frac{1}{I_{xx}} \\ \ddot{\theta} = ((I_{zz} - I_{xx})\dot{\phi}\dot{\psi} - J_r\dot{\phi}\Omega_r + LU_3)\frac{1}{I_{yy}} \\ \ddot{\psi} = \left((I_{xx} - I_{yy})\dot{\phi}\dot{\theta} + LU_4\right)\frac{1}{I_{zz}} \end{cases} \tag{3.1.4}$$

In the previous system of equations, g represents the gravity action, J_r $[Kg.m^2]$ each rotor inertia – eq. (3.1.5) – and Ω_r $[rad.s^{-1}]$ the sum of angular velocities produced by each rotor – eq. (3.1.6). These last two parameters influence the gyroscopic effect on the aircraft, occurring due to an unbalance of the sum of the angular velocities.

$$J_r = \frac{M \, . \, r_{rotor}^2}{2} \tag{3.1.5}$$

$$\Omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4 - \omega_5 + \omega_6 - \omega_7 + \omega_8 \tag{3.1.6}$$

The established relation between body axes rate and Euler angles rate implies two important limitations. First, it is only precise for one rotation at a time. If a second rotation is desired, the aircraft attitude must be carried to the origin state, assuming this state as $(0,0,0)$. Second, and although an inequality is formed between the two reference frames, the model can be considered valid for narrow changes in attitude.

3.2 Extended Model

The issue with the simplified version is that the hovering fluctuations must not be neglectable, as they are relevant when the aircraft needs to perform rotations, e.g., on take-off and landing approach or in the presence of crosswinds. Another consideration is the presence of drag, associated to air resistance. Therefore, a distinction must be made between the body axes rate and the Euler angles rate. The deduction for the extended model version is presented throughout the following subsections.

3.2.1 Rotation Matrix

The root step of a rotating body modelling is to formulate, through mathematics, its own rotational dynamics. To accomplish this mathematical relation, the right-hand rule is applied to each axis of the Euclidean space system.

The rotation vectors defined by eq. (3.2.1), (3.2.2) and (3.2.3) respect the right-hand rule and describe the angular displacement of the rotating object in relation to the inertial reference frame.

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.2.1}$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.2.2}$$

$$\mathbf{R}_\psi = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2.3}$$

$\boldsymbol{R}_\phi$ identifies roll rotation, $\boldsymbol{R}_\theta$ pitch rotation and $\boldsymbol{R}_\psi$ yaw rotation. With these three vectors and applying the three elemental rotations in a given order, it is obtained a specific rotation matrix. For this work, it is chosen a XYZ intrinsic rotation convention. Multiplying the three elemental rotations, as $\mathbf{R}_\Theta = \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi$, results in the rotation matrix described by eq. (3.2.4).

$$\mathbf{R}_\Theta = \tag{3.2.4}$$

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

Important is to notice that $\mathbf{R}_\Theta$ is an orthogonal matrix, meaning that ${\mathbf{R}_\Theta}^{-1} = {\mathbf{R}_\Theta}^{T}$. This relation is relevant for applications seen ahead.

3.2.2 Newton-Euler equations of motion

A quadrotor is a specific type of aircraft. Assuming zero drag force of any nature, it only acquires linear motion if its attitude vector is non-null. For this case, a mathematical relationship between translational and rotational dynamics is needed and it is obtained through Newton-Euler equations.

$$\begin{cases} \mathbf{F} = M\mathbf{a} + \boldsymbol{\Omega} \times (M\mathbf{V}) \\ \boldsymbol{\tau} = \mathbf{I}\boldsymbol{\alpha}_r + \boldsymbol{\Omega} \times (\mathbf{I}\boldsymbol{\Omega}) \end{cases} \tag{3.2.5}$$

$$\begin{cases} \mathbf{F} = \mathbf{T} - \mathbf{F}_g - \mathbf{F}_D + \boldsymbol{\Omega} \times (M\mathbf{V}) \\ \boldsymbol{\tau} = \mathbf{T}_\tau + \boldsymbol{\tau}_{Gyro} + \boldsymbol{\Omega} \times (\mathbf{I}\boldsymbol{\Omega}) \end{cases} \tag{3.2.6}$$

The eq. (3.2.6) is an extended expression of eq. (3.2.5). In these equations, $\mathbf{F} = [F_x \quad F_y \quad F_z]^T$ represents the force, $\boldsymbol{\tau} = [\tau_x \quad \tau_y \quad \tau_z]^T$ the momentum, **a** $[m.s^{-2}]$ the acceleration, $\boldsymbol{\alpha_r}$ $[rad.s^{-2}]$ the angular acceleration, **v** the velocity, **Ω** $[rad.s^{-1}]$ the angular velocity, **T** is the thrust, $\mathbf{F}_g$ the gravitational force, $\mathbf{F}_D$ the drag force caused by air resistance and $\boldsymbol{\tau}_{Gyro}$ the momentum generated due to the gyroscopic effect. The external products that appears in the equations are relative to the centrifugal and centripetal forces, respectively. Because this type of aircraft is assumed to be symmetric, the inertial moment matrix **I** is defined as shown in eq. (3.2.7).

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.2.7}$$

3.2.3 Open-Loop System

First, we take in consideration only the aircraft reference frame. By this light, both linear and angular velocity vectors are given by eq. (3.2.8) and (3.2.9), respectively.

$$\mathbf{V} = [u \quad v \quad w]^T \tag{3.2.8}$$

$$\boldsymbol{\Omega} = [p \quad q \quad r]^T \tag{3.2.9}$$

These vectors are decomposed in three axes each, fulfilling the six degrees of freedom of the overall open-loop system, represented by eq. (3.2.10).

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} - \mathbf{R}_\Theta \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix} - K_v \begin{bmatrix} u - W_x \\ v - W_y \\ w - W_z \end{bmatrix} - M \begin{bmatrix} qw - vr \\ -pw + ur \\ pv - uq \end{bmatrix}$$
$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} LU_2 \\ LU_3 \\ L\frac{C_d}{C_l}U_4 \end{bmatrix} + \boldsymbol{\tau}_{gyro} - \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \tag{3.2.10}$$

On eq. (3.2.10), C_l is the lift coefficient, K_v $[kg.s^{-1}]$ is any appropriate dimensioned variable associated with velocity and W_x, W_y and W_z concerns the air flow on the three axes.

Now it is possible to characterize the model attending the body inertial frame, so the gyroscopic effect momentum $\boldsymbol{\tau}_{gyro}$ is given by eq. (3.2.11).

$$\boldsymbol{\tau}_{gyro} = J_r\Omega_r \left(\boldsymbol{\Omega} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) = J_r\Omega_r \begin{bmatrix} p \\ -q \\ 0 \end{bmatrix} \tag{3.2.11}$$

Although eq. (3.2.10) works, it has a setback. This system has six outputs to control. However, of all the six degrees of freedom possible for this system, the horizontal motion along the X and Y axes are not directly controlled by any of the four command inputs, leading to an underactuated system with only four degrees of freedom. Thus, raising issues on stability level. The solution to this problem is to combine the inertial, in this case the earth, reference frame with the aircraft body axes. Therefore, and knowing that $\boldsymbol{R}_\Theta$ is an orthogonal matrix, the translational motion regarding earth is shown by eq. (3.2.12).

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \boldsymbol{R}_\Theta{}^T \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix} - K_v \begin{bmatrix} \dot{x} - W_x \\ \dot{y} - W_y \\ \dot{z} - W_z \end{bmatrix} \tag{3.2.12}$$

Combining now the translational motion about the inertial reference frame with the rotational motion about the body axes and applying Newton's second law and equivalent for the rotational motion we obtain eq. (3.2.13).

$$\begin{bmatrix} M\ddot{x} \\ M\ddot{y} \\ M\ddot{z} \end{bmatrix} = \mathbf{R}_{\Theta}{}^{T} \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix} - K_v \begin{bmatrix} \dot{x} - W_x \\ \dot{y} - W_y \\ \dot{z} - W_z \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{I}\dot{p} \\ \mathbf{I}\dot{q} \\ \mathbf{I}\dot{r} \end{bmatrix} = \begin{bmatrix} LU_2 \\ LU_3 \\ L\frac{C_d}{C_l}U_4 \end{bmatrix} + J_r\Omega_r \begin{bmatrix} p \\ -q \\ 0 \end{bmatrix} - \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \tag{3.2.13}$$

Solving the system of equations concerning linear and angular accelerations and as functions of input commands, results in the open-loop system fully actuated expressed by eq. (3.2.14).

$$\begin{cases} \ddot{x} = (\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi))\frac{U_1}{M} - K_v(\dot{x} - W_x) \\ \ddot{y} = (\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi))\frac{U_1}{M} - K_v(\dot{y} - W_y) \\ \ddot{z} = \cos(\phi)\cos(\theta)\frac{U_1}{M} - g - K_v(\dot{z} - W_z) \\ \dot{p} = ((I_{yy} - I_{zz})qr - J_r q\omega + LU_2)\frac{1}{I_{xx}} \\ \dot{q} = ((I_{zz} - I_{xx})pr - J_r p\omega + LU_3)\frac{1}{I_{yy}} \\ \dot{r} = ((I_{xx} - I_{yy})pq + LU_4)\frac{1}{I_{zz}} \end{cases} \tag{3.2.14}$$

According to the previous equation, the body angular acceleration is subject to control. This control is performed directly over its attitude. Therefore, the body attitude matrix is, in this work, represented by eq. (3.2.15).

$$\mathbf{\Theta}_A = [\alpha \quad \beta \quad \gamma]^T \tag{3.2.15}$$

The subscript A denotes the aircraft frame.

3.2.4 Euler angle and body axis rates

The aircraft dynamics are now better controllable, although one issue arises. In a real environment, the sensors are placed on the aircraft center body. Therefore, the values read by the same are relative to the aircraft reference frame, thus a relation between the aircraft and earth must exist.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{I}_{3x3} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_{\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_{\phi}\mathbf{R}_{\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{3.2.16}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.2.17}$$

Eq. (3.2.16) establishes a mapping from the inertial to the body reference frames through the Euler angles and body axes rates. This is the convention adopted in this work and commonly adopted on aeronautics, where the yaw rotation is performed first, then pitch and finally roll, after which the Euler angles rate is converted to the body axes rate (Stengel 2016). $\mathbf{I}_{3x3}$ denotes a $3x3$ identity matrix.

This conversion matrix is non-orthogonal, meaning that if $\mathbf{Q} \in \mathcal{M}_{nn}, \forall\, n \in \mathbb{N}^*$, then $\mathbf{Q}^{-1} \neq \mathbf{Q}^{T}$. If the inverse transformation is applied, we are now obtaining the Euler angles rate from the body axes rate.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.2.18}$$

Solving eq. (3.2.16) it is obtained eq. (3.2.17). Transforming this into a body to inertial frame relation, we obtain eq. (3.2.18). Analysing this last one, a singularity at $\theta = 90°$ is detected. In this region, the aircraft is intrinsically unstable. Nevertheless, as it is shown on Flight control section, this singularity is not of concern on any flight step.

3.3 Motor dynamics and configuration

Two specific experiment were conducted in (Brito 2016). One to obtain a mathematical representation of the E305 2312E BLDC motor behaviour. The other, to understand the improvement of a two-rotor configuration on each arm over a one-rotor configuration.

3.3.1 Motor dynamics

With the first experiment, the mathematical representation led to an output as a function of only one input. Any constant coefficients are already considered. Eq. (3.3.1) shows the TF, with unitary static gain, representative of the motors dynamics. Both input and output are expressed in units of force, where the first regards the desirable force and the second the one that is actually exerted by the motors.

$$F_n(s) = \frac{47.6205}{s^2+25.9384s+47.6205} \tag{3.3.1}$$

3.3.2 One-motor vs two-motor configuration

Another experiment relates with the influence of the motors on each frame. Specifically, to understand how the second motor on each arm improves the thrust. Figure 3.2 and Figure 3.3 represent a static and nonlinear third order system approximation for the one-rotor and two-rotor configuration, respectively. The corresponding functions are given by eq. (3.3.2) and eq. (3.3.3).

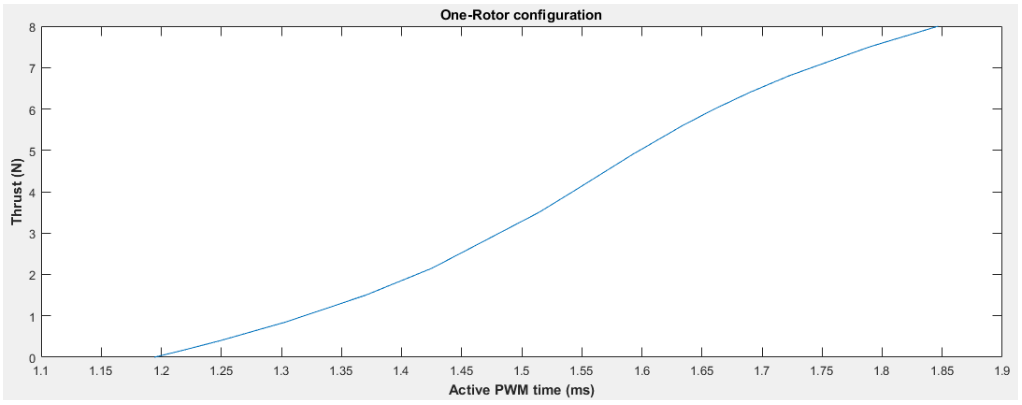


Figure 3.2: One-rotor configuration.

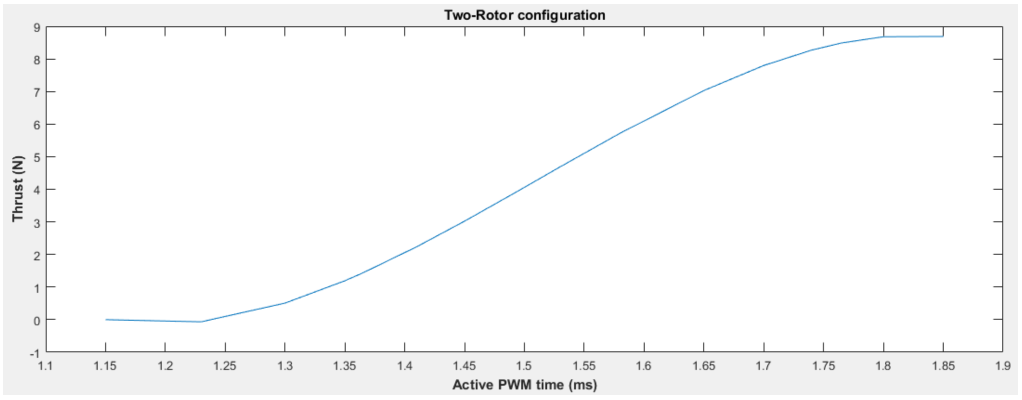


Figure 3.3: Two-rotor configuration.

$$f_{1R}(t) = -37.288994{T_{PWM}}^3 + 172.286169{T_{PWM}}^2 - \\ -249.1176057_{PWM} + 115.291879 \tag{3.3.2}$$

$$f_{2R}(t) = -\,70.427872{T_{PWM}}^3 + 319.248999{T_{PWM}}^2 - \\ -461.320062T_{PWM} + 215.423951 \tag{3.3.3}$$

On the two previous equations, f_{1R} and f_{2R} represent the force produced on each quadrotor arm in one-rotor configuration and two-rotor configuration, respectively, and T_{PWM} represents the PWM period.

These experiments were conducted by (Brito 2016) upon the mentioned motors model. The objective was that these second motors would act as redundant motors to avoid system breakdown in case of failure of the primary motor.

From Fig. 3.3, we can see that the improvement in thrust about Fig 3.2 is minimal. The way to surpass this physical threshold relies in different technical characteristics of each motor components, such as, e.g., the propellers length or pitch. By implementing a two-rotor configuration, the torque produced by each arm end may be generated with a lower power supply when compared to the one-rotor configuration. One reason is the influence of the top motor, which increases the efficiency of the bottom one.

3.4 The landing Approach

Once the model of the quadrotor is completed and equipped with the respective physical properties, the next step is to approach the aircraft landing with a mathematical description of the physics involved and with certain assumptions. In this section, only the physics are considerate.

3.4.1 Ground Effect

Firstly, let's assume the aircraft is hovering near the target and is able to acquire its relative position. If it is sufficiently closer to the target, an aerodynamic effect called Ground Effect happens to occur. This effect is previously described and the original deduction is based on an aircraft of a single propeller, specifically a helicopter, whose mathematical representation is expressed by eq. (2.2.3) and eq. (2.2.4). In this work, these equations are assumed as a valid estimate to the quadrotor case and so are tested considering the quadrotor geometry.

3.4.2 Touchdown

A focal point is the last step of the aircraft landing, where the leg meets the target. At this moment, important physics assumptions are to be made. In reality, the leg suffers deformation on impact moment. This deformation can be quantified by relying on Hooke's Law.

$$F_s = -K_s z \tag{3.4.1}$$

Of the mass-spring system expressed by eq. (3.4.1), F_s represents the force exerted on the spring, K_s $[Kg.s^{-2}]$ the stiffness constant and z [m] the deformation, or stretch, quantity of the spring. The negative sign intends to indicate the opposite direction between the spring deformation and force exerted by the spring. This is a second order system and it is marginally stable. This means that, if this system suffers a disturbance, the spring will never cease its oscillating motion. The solution is to add a damping element, hence turning into a mass-spring-damper system, as in the suspension system of an automobile.

$$m\ddot{z} = -K_s z - K_d \dot{z} \tag{3.4.2}$$

Eq. (3.4.2) is a more complete version of the previous one, where a damping factor, K_d $[Kg.s^{-1}]$, is added. It represents the friction against the spring stretch direction, causing the spring to eventually return to its resting position.

This last equation is the one to apply to the existing model. Its quality in the representation of the quadrotor legs dynamics is put to the test on the Simulation and Results Chapter. Here, the behaviour of the quadrotor is studied assuming $K_s = 10000\ Kg.s^{-2}$ and $K_d = 100\ Kg.s^{-1}$. With the first assumption, the quadrotor legs are expected to deform about two millimeters, given the overall body mass.

In this thesis, only a flat surface is considered, on which is performed a pure vertical landing by the quadrotor. For this, the four legs are mathematically characterized by Hooke's Law, previously mentioned.

3.4.3 Disturbances

The most resounding disturbance approached here is the air flow. It can have a major contribution on aircraft final approach. As the aircraft is lowering in altitude, the more critical becomes the information about its position and attitude in relation to the target. The force applied by the air resistance to aircraft motion is given by eq. (2.2.2), but approximated to $\mathbf{F}_D$, as observed on eq. (3.2.6).

3.5 Flight control

In this section, are presented the control methodologies as well as the control architectures for each of the six degrees of freedom. It is important to refer that some assumptions are made, which are described below.

All the physics quantities regarding the aircraft motion are read from ideals sensors. Clearly, this kind of sensors do not exist, but for simplification purposes they are considered. Because the rotor inertia is very small, is therefore considered zero for simulation purposes.

The motors dynamics obtained via experimentation conducted on Brito's work are considered. In his work, the control unit is an Arduino, which provides PWM signals to control each motor. Thus, this time signal is converted to a force quantity that is described by eq. (3.3.2) or eq. (3.3.3), depending on the topology applied. In the simulation environment, the output of the control system in measured in Newtons, so a conversion from force to time is necessary to preserve the characteristics of the real system. Fig. 3.4 illustrates this conversion under the form of a blocks diagram.

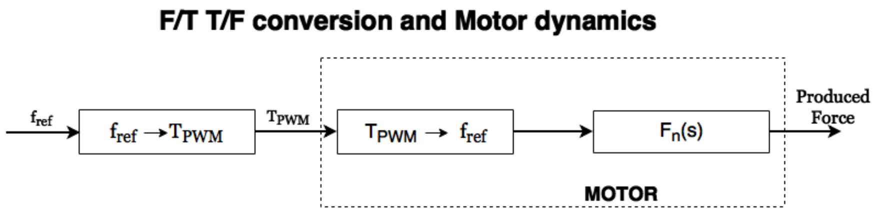

F/T T/F conversion and Motor dynamics
f_{ref}
$f_{ref} \rightarrow T_{PWM}$
T_{PWM}
$T_{PWM} \rightarrow f_{ref}$
$F_n(s)$
Produced Force
MOTOR

Figure 3.4: PWM and Force signals conditioning.

In the following control schemes, the motors' dynamics and time / force relations are integrated.

3.5.1 PID controller

In the following subsections, the control schemes are composed by PID controllers due to their simplicity of implementation. To find the parameters, the Ziegler-Nichols tuning methods and the PSO algorithm are applied. These architectures are structured in two stages: the inner loop, where the angular / linear speed is controlled; and the outer loop, where the position / attitude is locked at a desired setpoint. In this cascading control loop, the inner loop must be obviously faster than the outer loop. The setpoint is applied to the outer loop, which is not able to eliminate inner loop disturbances, like drag. Also, an obvious reason lies with the fact that position is an integration of velocity, thus slowing down the system response to a positional setpoint.

PSO algorithm can have better performance than Ziegler-Nichols method on tuning the PID controllers as observed on (Yadav and BhuriaVijay 2015; Edaris and Abdul-Rahman 2016) works. Nonetheless, it may be difficult to find a local optimum that drives the system to a stable closed-loop or even to find a local minimum in the first place (Clerc and Kennedy 2002). Each particle may be driven away from a satisfactory solution depending on its location when initialized or the location of the global best particle.

Infra are the procedures to take in order to acquire the controllers' parameters, along with tables showing the values obtained for the same parameters.

3.5.1.1 PSO method

The first algorithm put to the test is the PSO. Because this is a discrete-time process, it is necessary to discretize both the continuous model and PID algorithm. From the previous eq. (2.6.1), the PID equivalent in the Laplace domain is given by eq. (3.5.1).

$$U(s) = KE(s) \cdot \left(1 + \frac{1}{sT_i} + sT_d\right) \tag{3.5.1}$$

For any continuous-to-discrete transformation, a sampling of 10 ms and the bilinear – *Tustin* – transformation, eq. (3.5.2), are applied (K. J. Astrom and Wittenmark 1996) in order to preserve the dynamics of the continuous model. The resulting PID discrete controller is described by eq. (3.5.3).

$$s = \frac{2}{T_s}\frac{\mathfrak{z}-1}{\mathfrak{z}+1} \tag{3.5.2}$$

$$U(\mathfrak{z}) = KE(\mathfrak{z}) \cdot \left(1 + \left(\frac{T_s}{2}\frac{\mathfrak{z}+1}{\mathfrak{z}-1}\right)\frac{1}{T_i} + \left(\frac{2}{T_s}\frac{\mathfrak{z}-1}{\mathfrak{z}+1}\right)T_d\right) \tag{3.5.3}$$

Where $\mathfrak{z}$ represents the discrete domain, s the *Laplace* domain and T_s is the sampling period. The continuous system model is also discretized, but for simplicity are assumed null disturbances and the physical properties are preserved as constants.

The PSO algorithm is configured with a sampling period of 10 ms, swarm population of 20 particles where each particle is initialized with a random value comprised between 0 and an arbitrary positive value, 1.49 for both adjustmemt coefficients and a maximum of 1000 cycles if the stop condition has not yet been met. The reference signal is a unit step function, changing from zero to one in ten milliseconds at second two. The time horizon is set in the range $[0; 10]s$ and sampling frequency of 100 samples per second. Throughout the next experiments, no restrictions were superimposed on the acquisition of the solutions.

The cost function chosen is a ponderation between the mean-squared error and a mean square variation of the control action and is represented by eq. (3.5.4).

$$J(.) = \left(\alpha \sum_k^N \left(ref(k) - y(k)\right)^2 + \tag{3.5.4}$$

$$+\beta \sum_k^N (u(k) - u(k-1))^2)\frac{1}{N}$$

Where α and β are wheights with arbitrary values, ref is the reference value or setpoint and y the system output. The stop condition is based on the change of the cost function. If this change represents less than 10^{-6}, then the sequence of solutions converges, the process ends and the solution is a local minimum or is very close to one.

Table 3.1 represents the PD and PID parameters obtained through PSO algorithm to control the roll / pitch angular speed. This continuous model must be also discretized.

Table 3.1: Roll / pitch controllers' gains obtained via PSO method.

Roll / Pitch	**Kp**	**Ki**	**Kd**
PD	1.7908	-	0.4907
PID	2.1672	5	0.4418

Where $K_p = K$, $K_i = \frac{K}{T_i}$ and $K_d = KT_d$. With these parameters, a step signal is applied to the closed-loop system. The result is illustrated on Fig. 3.5.

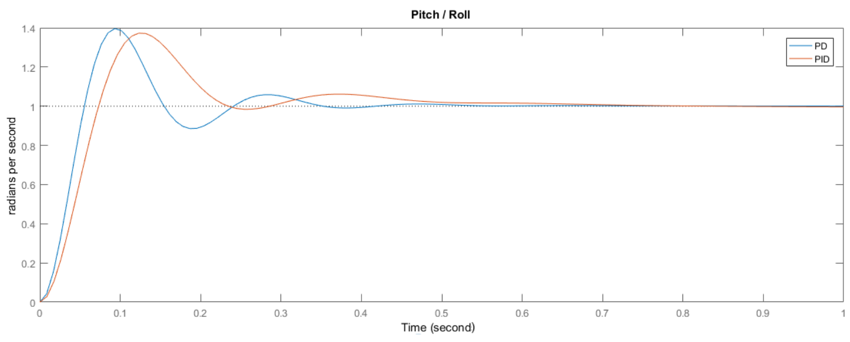


Figure 3.5: Step response to pitch / roll speed closed-loop system. Controller's gains based on PSO method.

It is possible to see that PD controller provides a reduced rising and settling time compared to PID controller.

In this work, PSO is employed for attitude control. Due to factors mentioned previously, the presumably bad PSO parameters chosen and the absent of restrictions, the solution did not converge for any local optimum that could place the horizontal motion and altitude closed-loop systems in the stability regions.

Also, the sampling period used to discretize the continuous model affects the analytical discrete model precision. Higher the sampling rate, more similar are the continuous and discrete systems' response to an input. However, as the coefficients become smaller, the more precision may be required. In the case of horizontal motion, where the respective control is placed on top of the attitude control block, this situation is more evidenced. Consequently, the controllers' parameters for the horizontal motion and altitude are ruled by Ziegler-Nichols Ultimate Sensitivity method.

3.5.1.2 Ultimate Sensitivity method

The main idea behind this heuristic method lies in increasing progressively a sensitivity gain until the system reaches the threshold of stability. At this point, the sensitivity gain is given by K_u and the oscillatory reaction of the system possesses a period given by T_u (Ziegler and Nichols 1995). Table 3.2 represents each controller's gains obtained according predefined rules.

Table 3.2: Controllers' gains obtained via Ultimate Sensitivity method.

Controller	K	Ti	Td
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
PID	$0.6K_u$	$T_u/2$	$T_u/8$

This tuning method is achieved on a close-loop system. Another method is applied on the open-loop system by analysing the system reaction to a stimulation. The decision to not going with this last is due to a relative difficulty in analysing the curve and because the step response of the overall system in particular is not so monotone as desired.

The process to determine the controllers' gains is here presented with the ascension speed control. The first step is to drive the closed-loop to the limit of stability and find K_u and T_u. The marginally stable system is illustrated on Fig. 3.6.

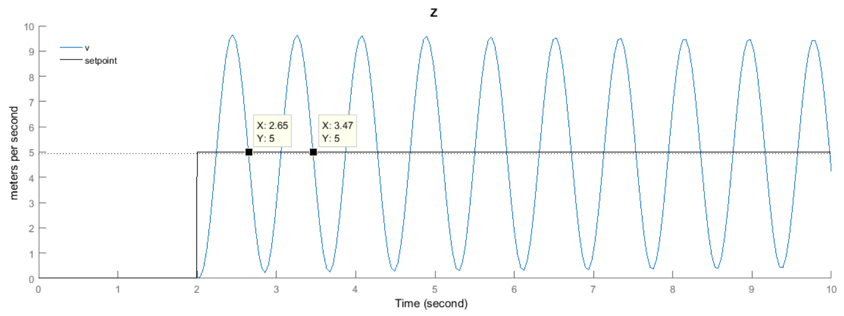


Figure 3.6: Marginally stable ascension speed closed-loop system.

With this test, it is obtained $K_u = 32.5$ and $T_u = 0.82s$. Applying the rules of Table 3.2, it is possible to design the controller. Fig. 3.7 illustrates the closed-loop system with PD and PID control over the ascension speed to understand which provides the fastest response.

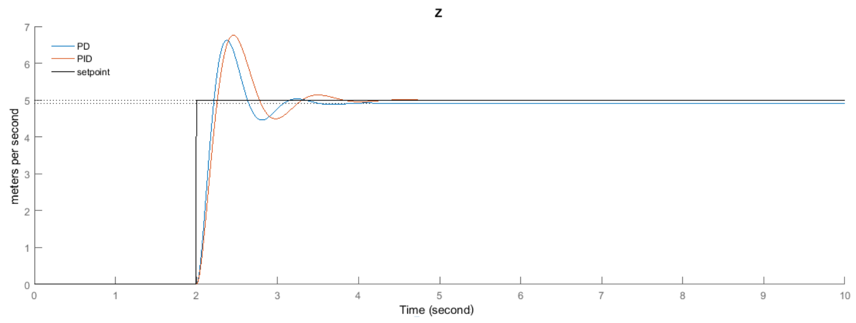


Figure 3.7: Step response to ascension speed closed-loop system. Controller's gains based on Ultimate Sensitivity method.

Analysing the figure above, PD controller delivers a reduced rising and settling time when compared to PID controller. However, this method provides slowest controllers than the PSO method, as foreseen. Also as seen in (Gibiansky 2012), a manual tuning may not be the best approach, in the sense that may produce a poorest performance by comparison with other sophisticated methods.

3.5.2 Attitude control

The first control scheme presented regards the quadrotor attitude. The closed-loop system assumes a cascading control loop with one inner loop and an outer loop. The inner loop, as mentioned, must be faster than the outer loop, as it will set the stall speed response of the system. The control loop is illustrated on Fig. 3.8.

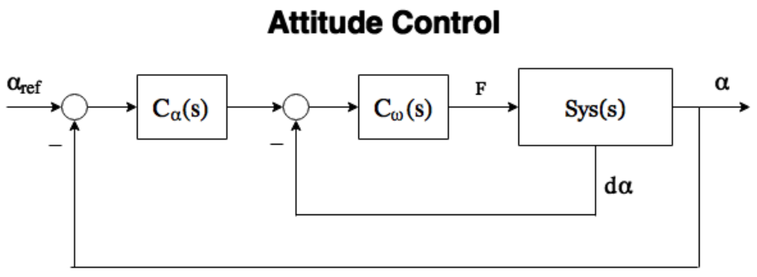


Figure 3.8: Attitude control scheme.

In order to maintain stability, a range of $[-30; 30]$ degrees is set as admissible for each angular position, thus restrained between these bounds. Under or above this range, the quadrotor stability and lift could be at risk.

The attitude controllers' gains are obtained through PSO method. On the inner loop, an angular speed controller is placed to stabilize the attitude control, benefiting from information about other state variable than angular position. The gains obtained can be seen on Table 3.3.

Table 3.3: Attitude controllers' gains obtained via PSO method.

Controller	**Kp**	**Ki**	**Kd**
Roll speed	1.7908	-	0.4907
Pitch speed	1.7908	-	0.4907
Yaw speed	2.3375	-	0.6894
$\boldsymbol{\alpha}$	1.5018	-	-
$\boldsymbol{\beta}$	1.5018	-	-
$\boldsymbol{\gamma}$	0.988	-	-

3.5.3 Position control

3.5.3.1 Altitude

The altitude is an important measure to control, as it is the core step to aircraft stabilization and motion. The closed-loop architecture adopted is illustrated by the following Fig. 3.9 and is similar to the one applied for attitude control.

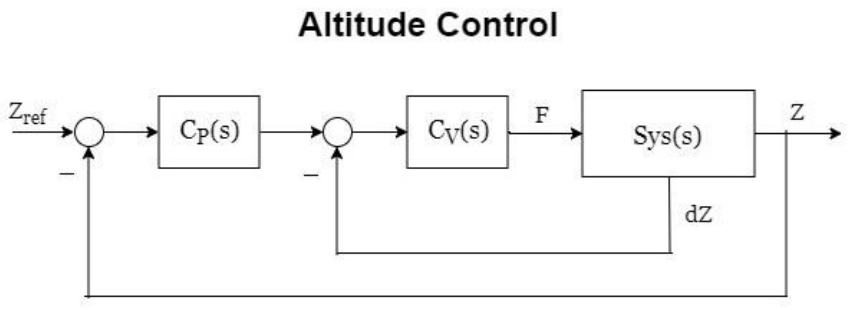


Figure 3.9: Altitude control scheme.

For both speed and position control on Z axis, the controllers' gains are acquired based on Ziegler-Nichols tuning rules. As shown before, this method does not give the best or the fastest controller, but only a good controller that can place the closed-loop system on the stability region and with a fairly good tracking control. Table 3.4 shows the gains obtained by applying the Ultimate Sensitivity method to altitude control.

Table 3.4: Altitude controllers' gains obtained via Ultimate Sensitivity method.

Controller	Kp	Ki	Kd
Speed	26	-	2.665
Position	2.065	-	-

3.5.3.2 X / Y control

The way to control longitudinal and lateral positions is naturally different from the way to control the altitude. To move along X or Y axes, the quadrotor must suffer an inclination, be it pitch or roll, respectively. Therefore, in these cases, a control over attitude is needed. On Fig. 3.9 is represented the control architecture for X and Y positions.

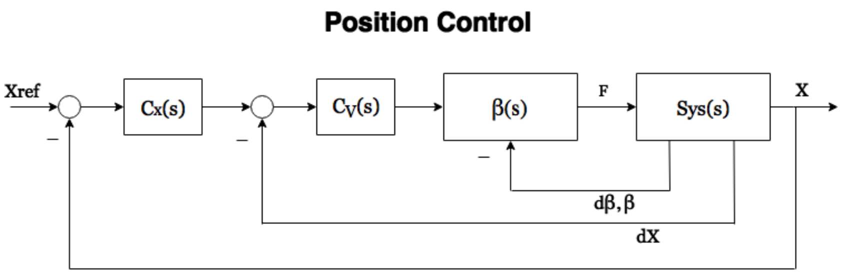


Figure 3.10: Position control scheme.

To control the motion on the horizontal plane, it is necessary to control the aircraft attitude in the first place. Hence, the attitude control is placed on the inner loop. As the block that feeds the output directly, it is more critical to the performance of the system. This is the main reason why PSO method is applied. On the middle loop, is placed the linear speed control. It provides a better stabilized angle setpoint to the inner loop. Finally, the outer loop is composed by the position control, where the position command, from a human controller or a high-level architecture, is applied.

On Table 3.5 are specified the controllers' parameters for speed and position control, obtained through Ziegler-Nichols method.

Table 3.5: Position controllers' gains obtained via Ultimate Sensitivity method.

Controller	Kp	Ki	Kd
X speed	0.7	0.392	0.098
Y speed	0.7	0.392	0.098
X	0.31	-	-
Y	0.31	-	-

3.5.4 Thrust control

One real limitation is the maximum amount of thrust that each motor can generate and provide to the body lifting. Obviously, the minimum thrust is zero, assuming air flow through the rotors is always forced down when they are powered and the blades are rotating. From the simulation point of view, this is characterized by a saturation block. However, even if the output is limited, the system integrators keep accumulating the saturated value. This can lead to faulty actuations, influencing the overall system response. To diminish this effect, an anti-windup technique is applied.

The technique applied is based on the back-calculation of the integral action. It reduces the rate at which the signal is stored. It is mainly composed by a PID controller and a coefficient to settle the discharge ratio (Bemporad 2011). Fig. 3.11 represents the architecture described.

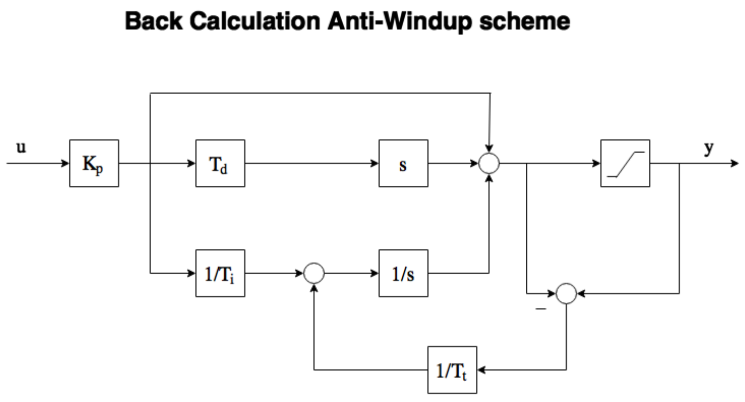


Figure 3.11: Back-calculation Anti-Windup with PID controller.

4 Simulations and Results

In this chapter are presented all the tests conducted on the overall system. It is a way to validate all the work done on Chapter 3.

Througout the next sections and for testing purposes, a variable setpoint and a time horizon in the range of $[0; 90]s$ are considered when only one rotation or translation is tested at a time. Also, each motors TF is contemplated and directly influence the results.

For a more direct analysis and easy reading of the graphs, the angular units are expressed in degrees.

Fig. 4.1 represents this hovering and the respective control action. The quadrotor remains at the origin of the reference frame regarding X and Y coordinates.

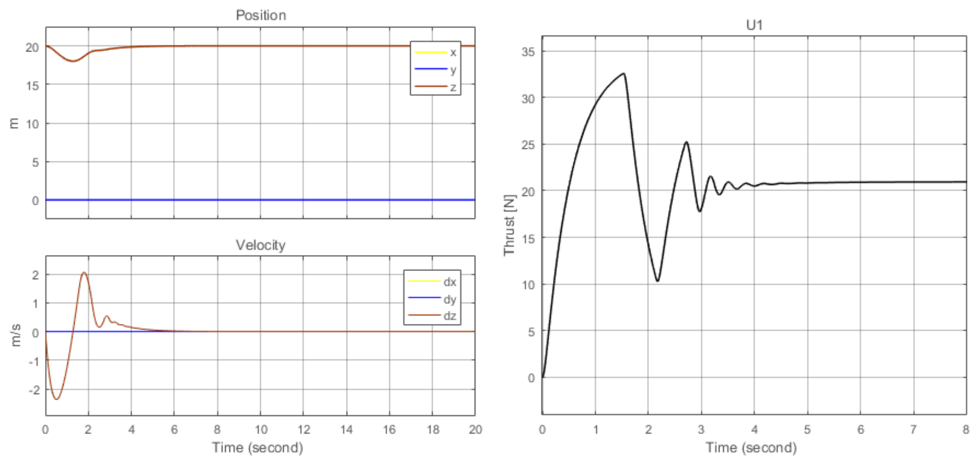


Figure 4.1: Hovering. From left to right, position and altitude control action.

Because at the start of the simulation the motors are at rest, the spikes on the right graph relate to the altitude control actuation over the motors, so the quadrotor remains at 20m setpoint, as seen on the left upper graph on the figure. The image from below refers to the speed at each time instant. In all figures where attitude or position are displayed, the respective velocities are illustrated below.

The quadrotor must be hovering sufficiently away from the ground to enable its rotation. Thus, the simulations are conducted on the aircraft while hovering at 20 m. The simulation starts with all motors off, hence the initial spikes corresponding to the transient state.

4.1 Zero drag effect

The set of graphs analysed during this sub-chapter, do not contemplate any influence of any kind of disturbance. On the next one, some tests are then executed considering the effect of air resistance to linear motion.

4.1.1 Attitude

In this section, the response of the quadrotor attitude is analysed. A separation between body and inertial frame is underlined.

4.1.1.1 Pitch rotation

The first step consisted in testing only one rotation at a time. Because the controllers for both pitch and roll are the same, are here presented only the pitch curves, as well as the control action.

Fig. 4.2 illustrates the evolution of the angular displacement and rate of change over time. Because the specified limit to pitch and roll rotations are 30 degrees, the maximum setpoint applied is 25 degrees. The respective control action and the influence on altitude control actuation are presented on Fig. 4.3.

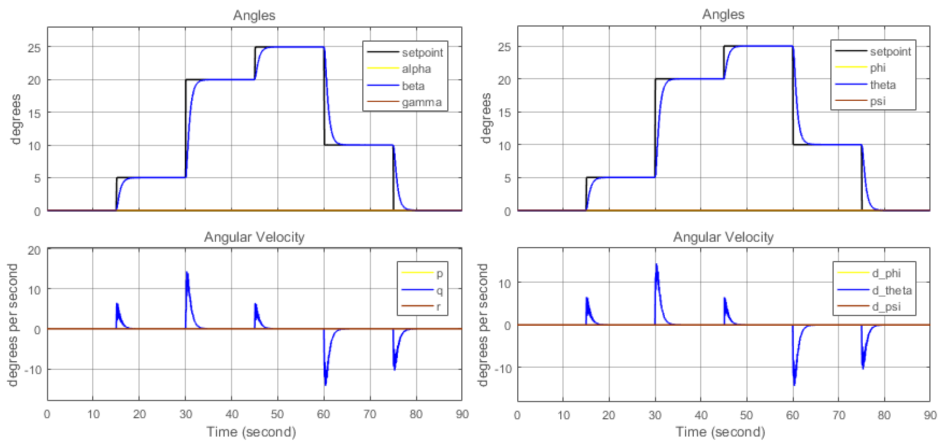


Figure 4.2: Rotation response to a variable pitch setpoint. From left to right, body and Euler angles.

We can see in the previous figures, the body and inertial curves are the same concerning both angular displacement and rate of change. The response is fairly swift, with a rising time of about four seconds on the most expressive setpoint variation. This is the simplest case where only one rotation occurs. The next figure concerns the altitude and pitch control actions.

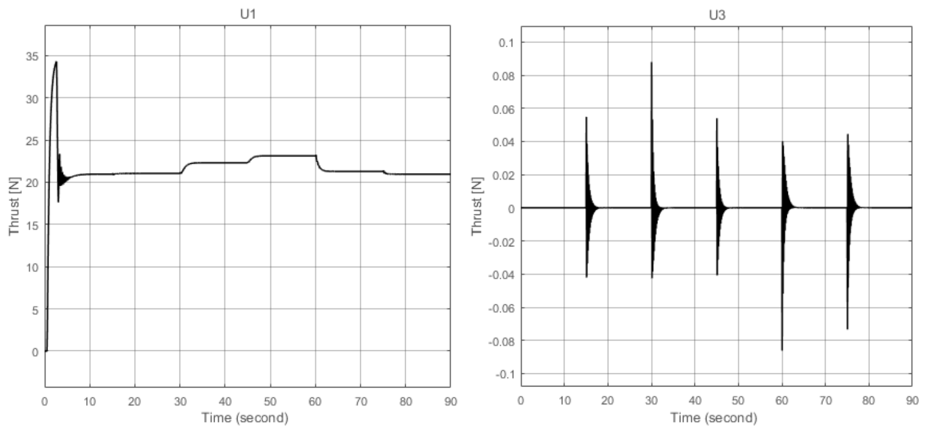


Figure 4.3: Control actions in response to pitch error. From left to right, altitude and pitch control actions.

The left image represents the fluctuations in the altitude control action U_1 due to the inclination of the quadrotor. The more emphasized this inclination, the greater must be the thrust supplied to the overall system. The right image is the control action U_3 and the spikes denounce a change of the angle setpoint. The actuation of the roll rotation is the same as pitch, only inverted. From the right-hand rule, when roll rotation is positive, the respective angle is negative. Fig. 4.4 intent is to provide further detail on the U_3 signal.

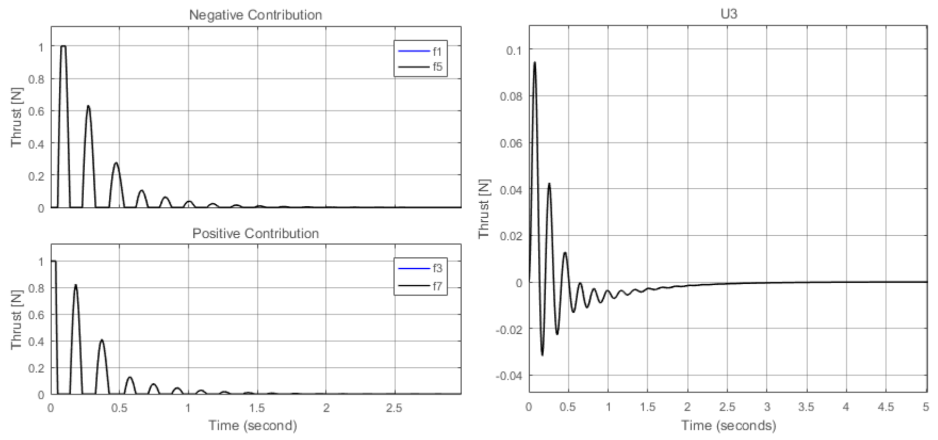


Figure 4.4: Control actions in response to pitch error. From left to right, force to be applied by each rotor and pitch control action.

A 20 degrees pitch step signal was applied to the system, resulting in the control action seen on the previous figure.

On the left side of the image, is presented the influence of each motor, with its dynamics not considered here. A saturation is set so each motor can only contribute with a thrust force in the range of [0; 1] N. When the control error is below zero, the motors 1 and 5 provide more thrust than the antagonistic motors 3 and 7. When the control error is above zero, the opposite happens.

On the right side of the image, the control action U_3 is presented. It has a smooth variation and small values are enough to cause the quadrotor to rotate.

4.1.1.2 Yaw rotation

The yaw rotation is similar to both pitch and roll rotations, but some particularities caused different controllers' gains. Fig. 4.5 describes the yaw response to a variable setpoint, with the respective control action found on Fig. 4.6.

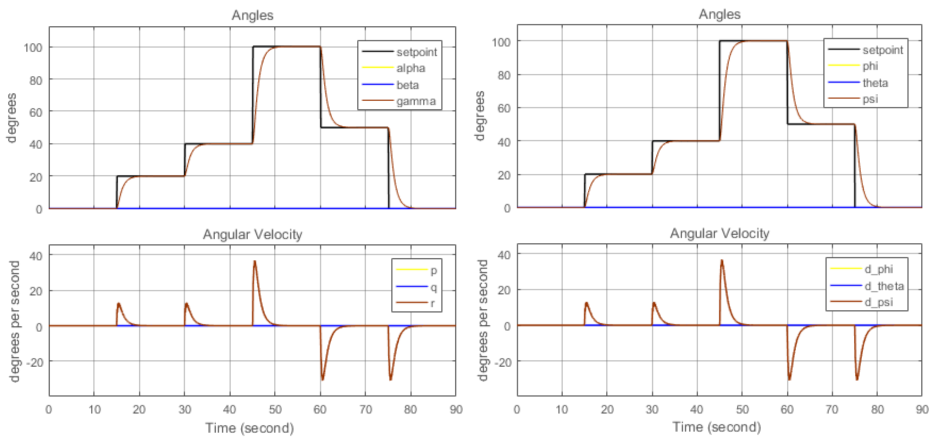


Figure 4.5: Rotation response to a variable yaw setpoint. From left to right, body and Euler angles.

The image above describes the response to a variable setpoint from the point of view of the body and the inertial frames, respectively. Similarly to the pitch / roll case, both curves are the same as only one rotation occurs and the other angular positions remains zero. The response time provided by yaw controllers is also similar to the ones applied in pitch / roll control.

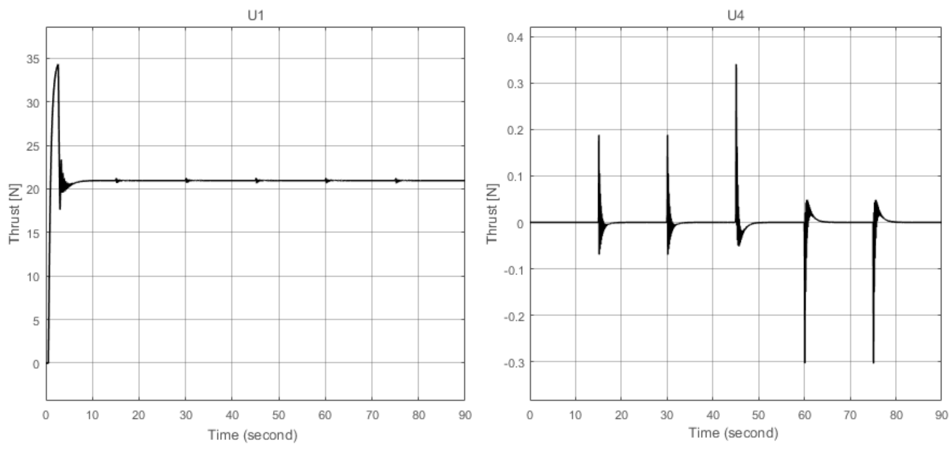


Figure 4.6: Control actions in response to yaw error. From left to right, altitude and yaw control actions.

In opposition to pitch or roll, the yaw rotation has meaningless significance on the overall system stability, still considering only one rotation applied. This assertion can be verified by superficially analyse control action U_1 on the left side of the figure above.

On the right side of the image, a graph representing the control action U_4 is shown. The spikes represent each change of the reference value, which develops a control error and the subsequent control action observed.

4.1.1.3 Pitch and Roll rotations

Another situation is now simulated. Roll and pitch rotations are now simultaneously occurring. The point of this test is to understand the difference of perception between an inertial and a rotating reference frames. The body and Euler angles and angular speeds curves are illustrated on Fig. 4.7.

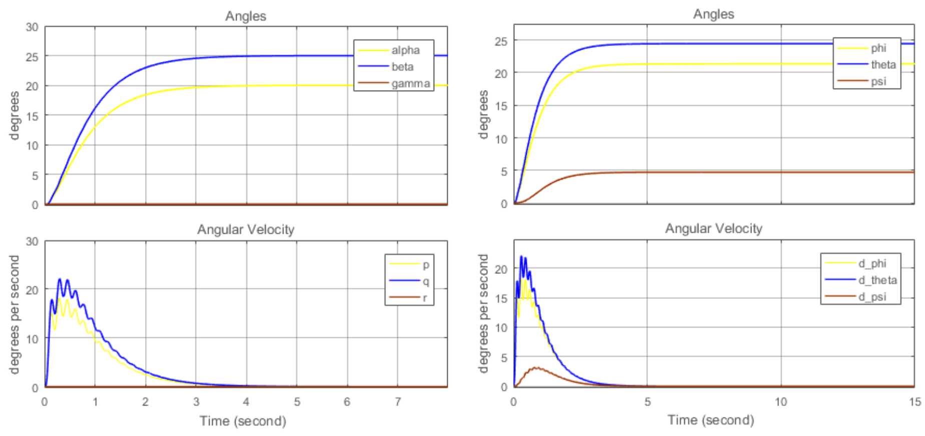


Figure 4.7: Rotation response to a step input signal applied to roll and pitch. From left to right, body and Euler angles.

For this test, step signals with static value of $(20, 25)$ degrees are applied to (α, β) angles, respectively.

In the previous graphs, the ones from the left are clearly different from the ones on the right. From the inertial reference frame perspective, the body performs a yaw rotation. By analysing the curves on the body frame and the actuation signals presented on Fig. 4.8, this fact does not happen. This effect happens when two or more rotations are executed and its confirmation comes from the following tests.

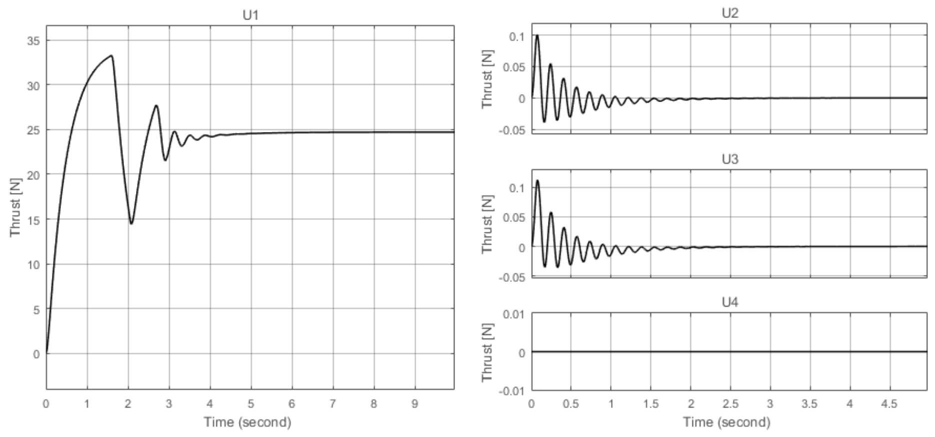


Figure 4.8: Control actions in response to roll and pitch error. From left to right, altitude and attitude control actions.

On the graph on the left, the actuation for the altitude is shown. No significant variations are noted on the curve comparing with the situation observed on Fig. 4.1.

The three graphs on the left translate the control actions U_2, U_3 and U_4, respectively. Roll and pitch actuations resemble each other as supposed, given the similarity in terms of model description and controller design.

4.1.1.4 Pitch and Yaw rotations

In close relation to the previous test, now the pitch and yaw rotations are performed at the same time. It is expected, from the inertial observer point of view, to be performed a roll rotation due to yaw. The results are present on Fig. 4.9.

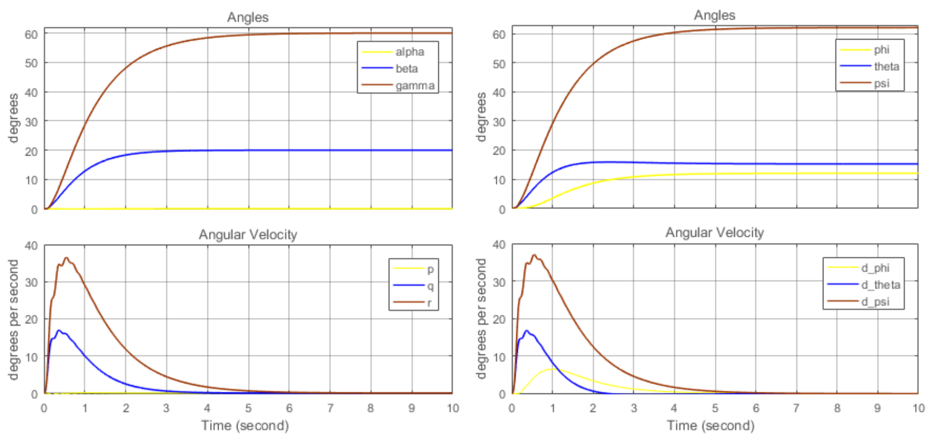


Figure 4.9: Rotation response to a step input signal applied to pitch and yaw. From left to right, body and Euler angles.

Step signals with static value of $(20, 60)$ degrees are applied to (β, γ) angles, respectively, for the time being test.

From the body frame curves, the result is closely connected to the setpoints defined. On the opposite, the inertial observer detects a third rotation. This is the same effect mentioned on the previous test.

On Fig. 4.10 are specified the control actuation graphs.

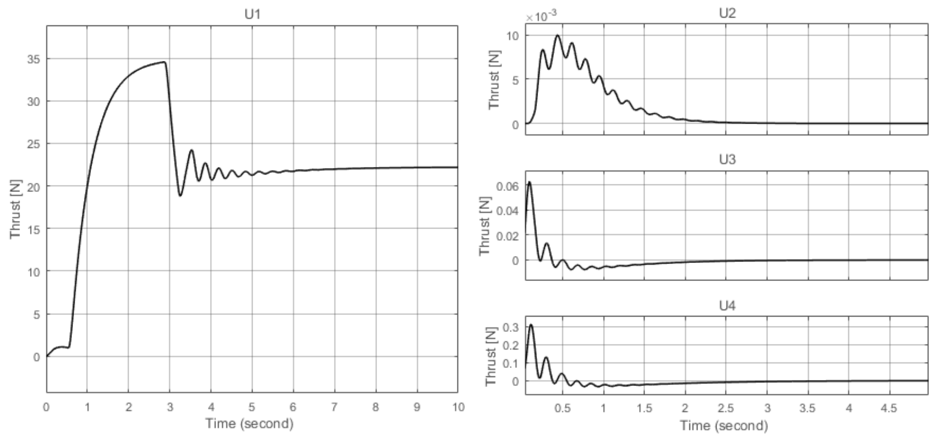


Figure 4.10: Control actions in response to pitch and yaw error. From left to right, altitude and attitude control actions.

The above figure may seem somewhat inconsistent with the previous one, at first glance. Due to the execution of pitch and yaw positive rotations, given the right-hand rule, the quadrotor will lean to its left. This means that it tends to do a negative roll. At the same time, the control action responsible for this rotation is triggered. When the pitch and yaw angular speeds decrease, the opposite effect is slightly noticed.

4.1.1.5 Pitch, Roll and Yaw rotations

For the last test involving directly the attitude control, the pitch, roll and yaw are simultaneously actuated. The results can be seen ahead on Fig. 4.11 and 4.12.

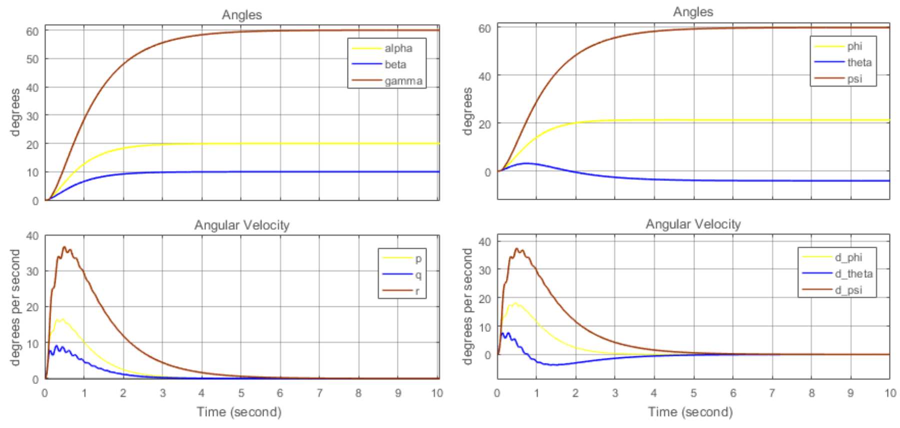


Figure 4.11: Rotation response to a step input signal applied to roll, pitch and yaw. From left to right, body and Euler angles.

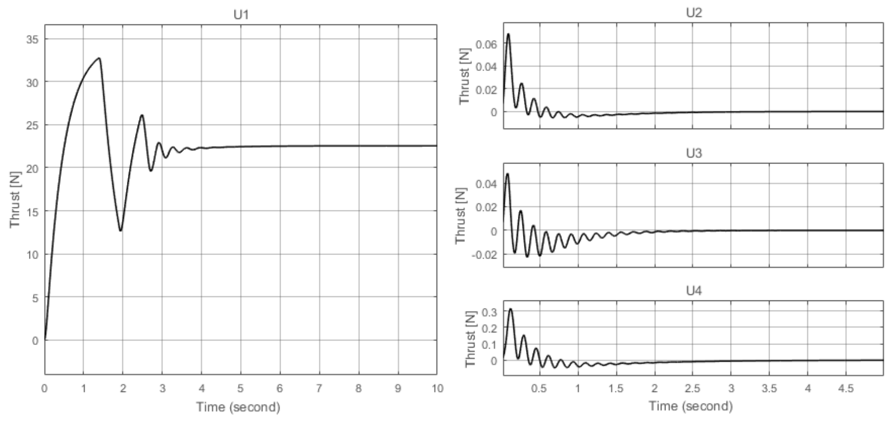


Figure 4.12: Control actions in response to roll, pitch and yaw error. From left to right, altitude and attitude control actions.

Three step signals are applied in this simulation. For pitch a reference of 10 degrees, for roll 20 degrees and for yaw rotation 60 degrees.

The information extracted of this simulation is, in all aspects, the same mentioned on the previous test conducted.

To conclude this section, the controllers designed through PSO algorithm provide general good response of the system. No evidence of overshoot and good rising time and settling time are good indicators and can provide a solid base for the position controllers, whose results obtained are described in the next section.

4.1.2 Position

In this section, are now presented the results for the position control.

4.1.2.1 Altitude displacement

The altitude control is, as declared before, the core block for the quadrotor hovering. Thus, the performance of the controllers are then analysed.

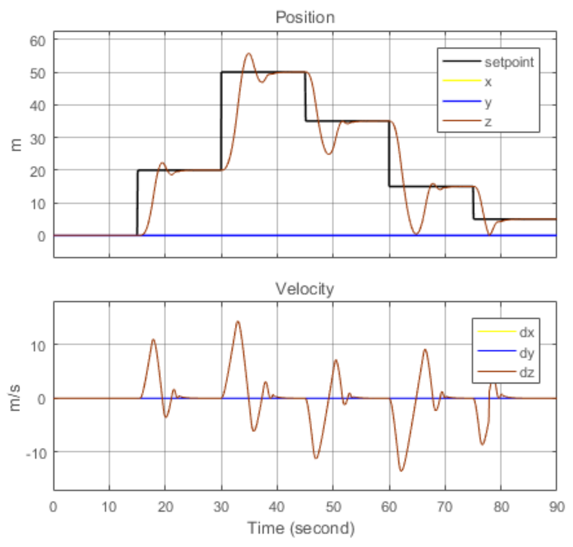


Figure 4.13: Translational response to a variable altitude setpoint.

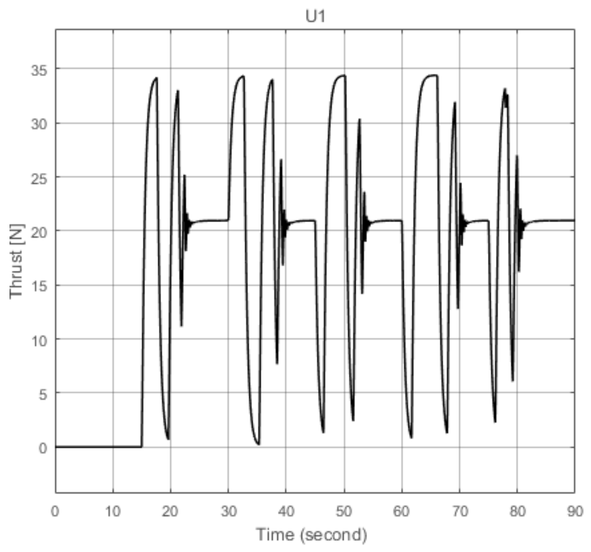


Figure 4.14: Control action in response to altitude error.

On Fig.4.13, are represented the trajectory control for the quadrotor altitude and the respective speed at each time instant. We can see that occurs overshoot at each setpoint change and both rising and settling time are slower than the ones analysed from the attitude controllers. Even though this may be considered acceptable away from the ground, when the quadrotor is too close to the target, a crash might happen. On seconds 64 and 78, roughly, this situation is noticed. Ahead, this problematic is approached.

From the Fig. 4.14, we can verify the wide set of values that U_1 assumes at each setpoint variation, which is not desirable. Particularly, because physical systems may not be able to support this actuation over prolonged periods of time.

4.1.2.2 X displacement

The control over X position is the same as the Y, so only the results, indicating the controllers' performance, for the first one are presented. On Fig. 4.15 are illustrated the system response to a variable setpoint command applied to horizontal motion. On Fig. 4.16, the altitude and pitch control actuations are denoted.

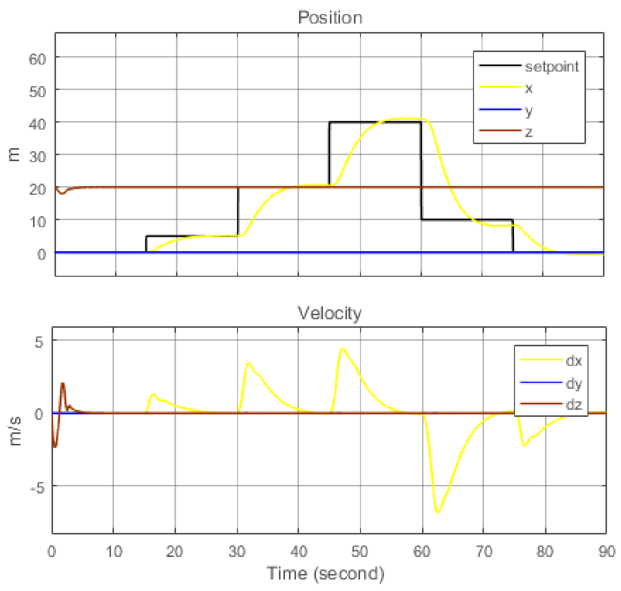


Figure 4.15: Translational response to a variable setpoint applied to X axis.

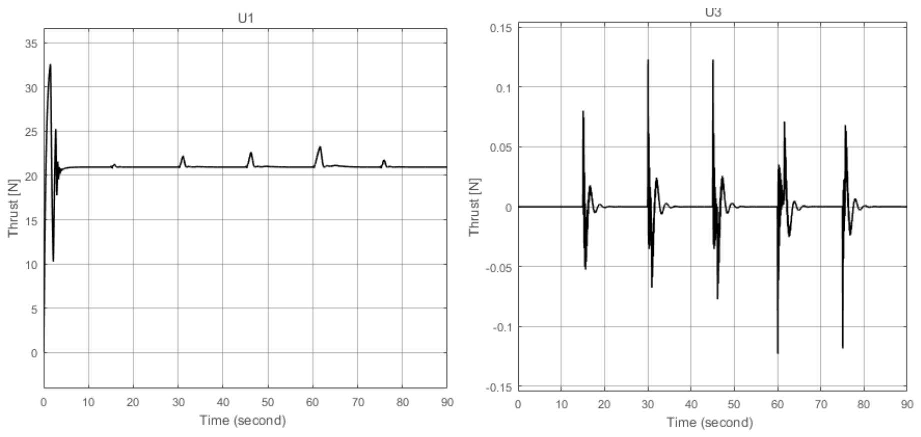


Figure 4.16: Control actions in response to X error. From left to right, altitude and pitch control actions.

On the first image, are represents the displacement and speed over the X axis. In order for the quadrotor to move along the X axis, an actuation for the motors to apply the pitch rotation must be sent. Because this is a complex system and the control techniques used have their characteristic setbacks, the results show a somewhat slow response. Nonetheless, as we can inspect from the figure below, the actuation over the altitude and pitch rotation can be considered smooth. From the graph on the right, the U_3 actuation oscillates around a narrow range of values.

4.1.2.3 X and Y displacement

On the next simulation, the horizontal motion control is reviewed. Only now, motion on the hole horizontal plane is considered. On Fig. 4.17 are shown the position and velocity response, and on Fig. 4.18, the control actuations signals are exposed.

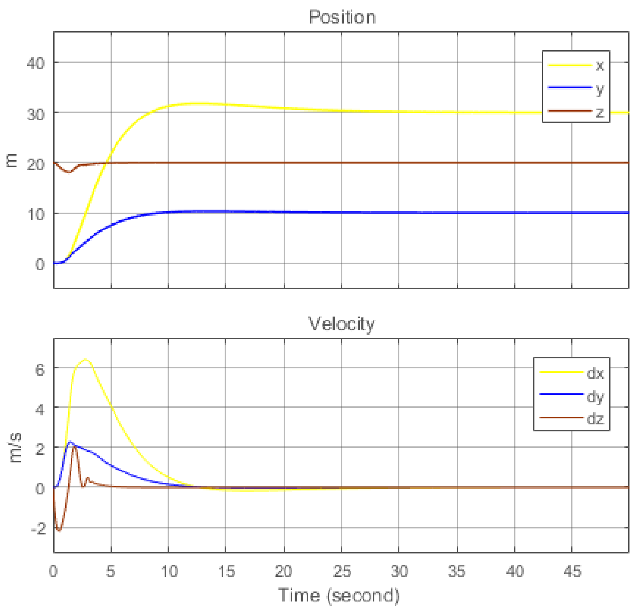


Figure 4.17: Translational response to a step input signal applied to X and Y axes.

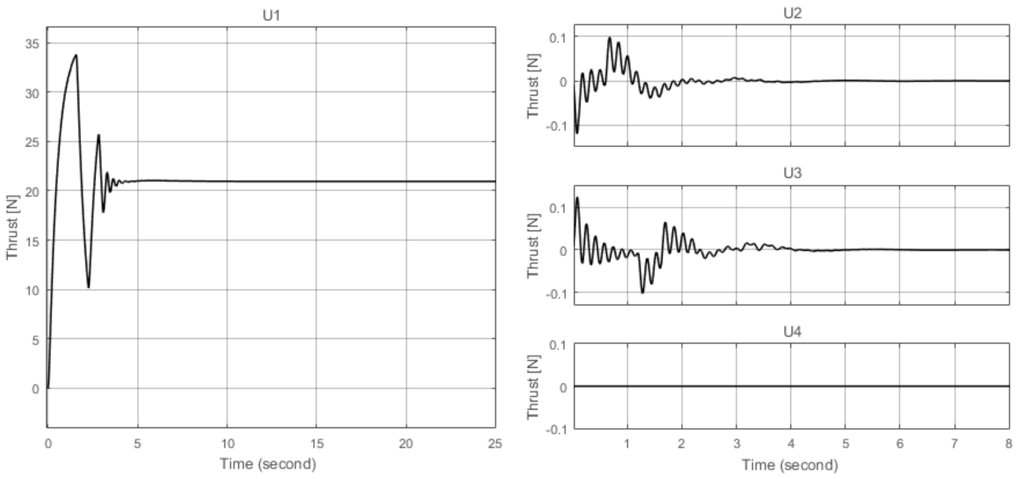


Figure 4.18: Control actions in response to X and Y error. From left to right, altitude and attitude control actions.

The setpoints chosen for this test were step signals with a static value of $(30, 10)$ meters applied to (X, Y) controllers, respectively. One can conclude that further the setpoint, the more expressive becomes the overshoot and the settling time. Notwithstanding, the quadrotor moves towards the desired setpoints and the control actuations are not too expressive.

4.1.2.4 X, Y and Z displacement

The three translational motions control are now combined altogether. This simulation enables to conclude if the displacement of the quadrotor over the three Cartesian axes simultaneously is possible. Fig. 4.19 shows the graphs relative to trajectory control.

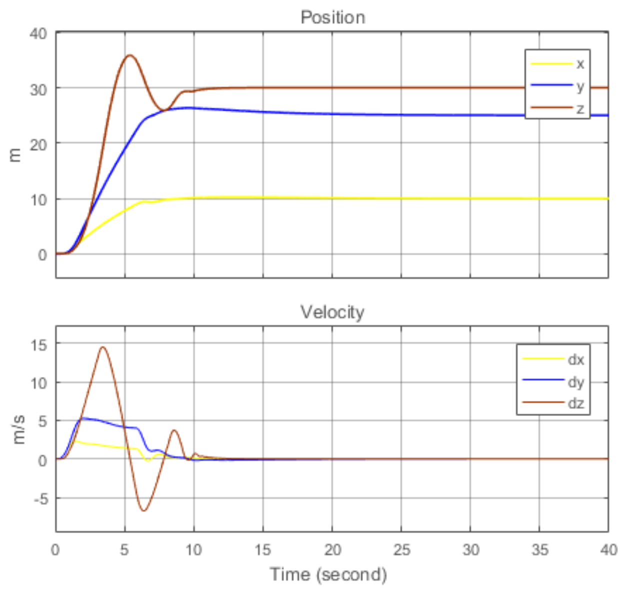


Figure 4.19: Translational response to a step input signal applied to X, Y and Z axes.

In this simulation step signals with static value of $(10, 25,30)$ meters are applied to (X, Y, Z) controllers, respectively.

The results are similar to what was expected, since the temporal evolution of the curves presents the same dynamics as in the previous tests.

Fig. 4.20 illustrates the four model inputs actuation.

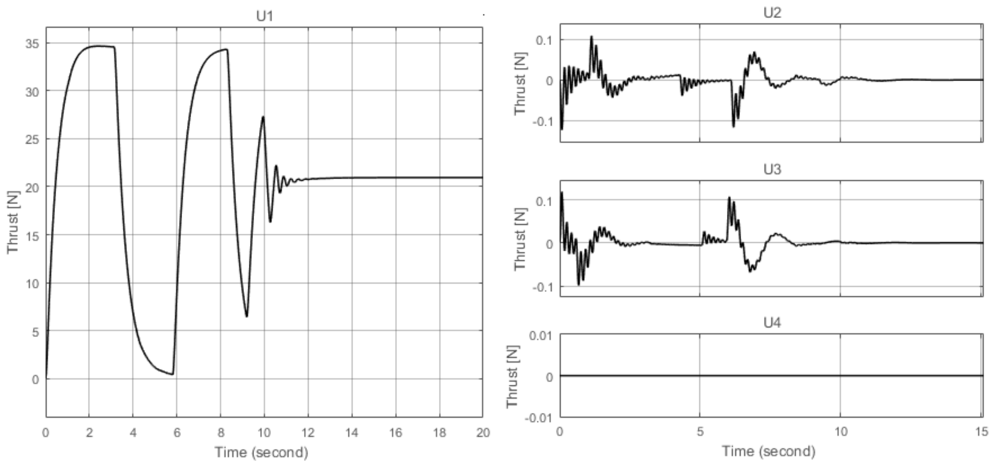


Figure 4.20: Control actions in response to X, Y and Z error. From left to right, altitude and attitude control actions.

On the figure above, the graph on the left, the result is similar to the first test conducted on this section. In what concerns to the graphs on the right, the actuation signals are similar to the ones observed on the two last tests, with a small correction at the same time. This is due to the similarity between the two kinematics models and controllers.

4.1.3 Landing

To understand the influence of the landing considerations approached on the last chapter, results are presented given the specified inertial parameters and geometry of the quadrotor. The first demonstration regards the GE.

4.1.3.1 Ground Effect

In this subsection, the relation between the thrust induced on the rotors IGE and OGE can be analysed through the following Fig. 4.21 and Fig. 4.22. The first demonstrates this relation at zero air speed, whilst in the second the aircraft is provided with forward speed. On both, the rotors are supplied with constant power.

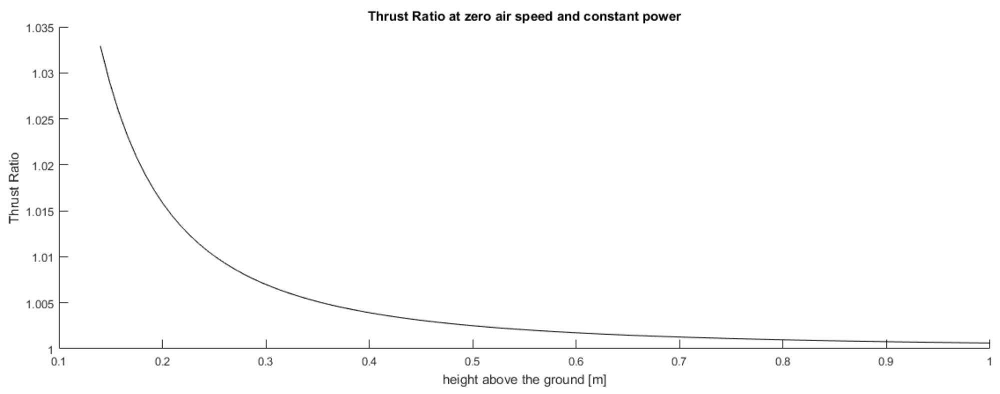


Figure 4.21: Thrust ratio at zero air speed and constant power.

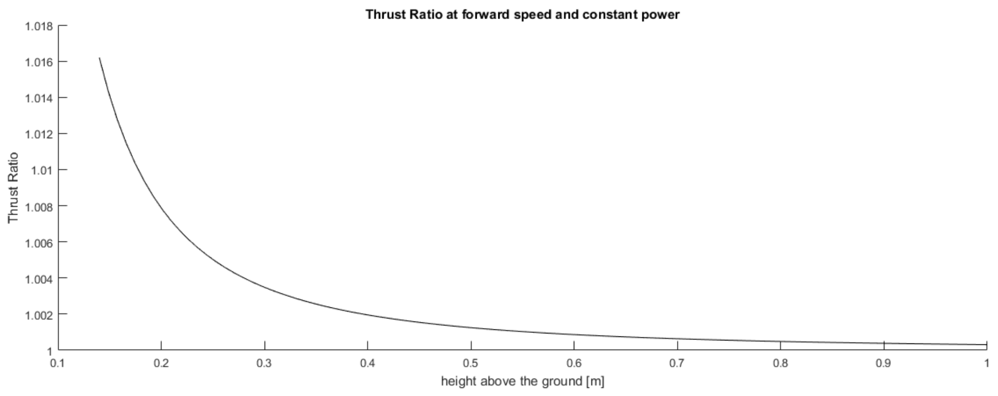


Figure 4.22: Thrust ratio at forward speed and constant power.

By inspecting the two curves above, one can assume that this effect can be neglected, as its impact on the quadrotor thrust IGE is less than 3.5% than OGE. This statement is only true given this specific quadrotor and knowing the used equations are inherently derived from a helicopter's rotor dynamics.

4.1.3.2 Touchdown

In this particular situation, the simulation consists on transport the quadrotor from a height of 20 meters to ground level and analyse its impact on the surface. Fig. 4.23 shows this effect on the quadrotor altitude, the reaction from the ground and the altitude actuation.

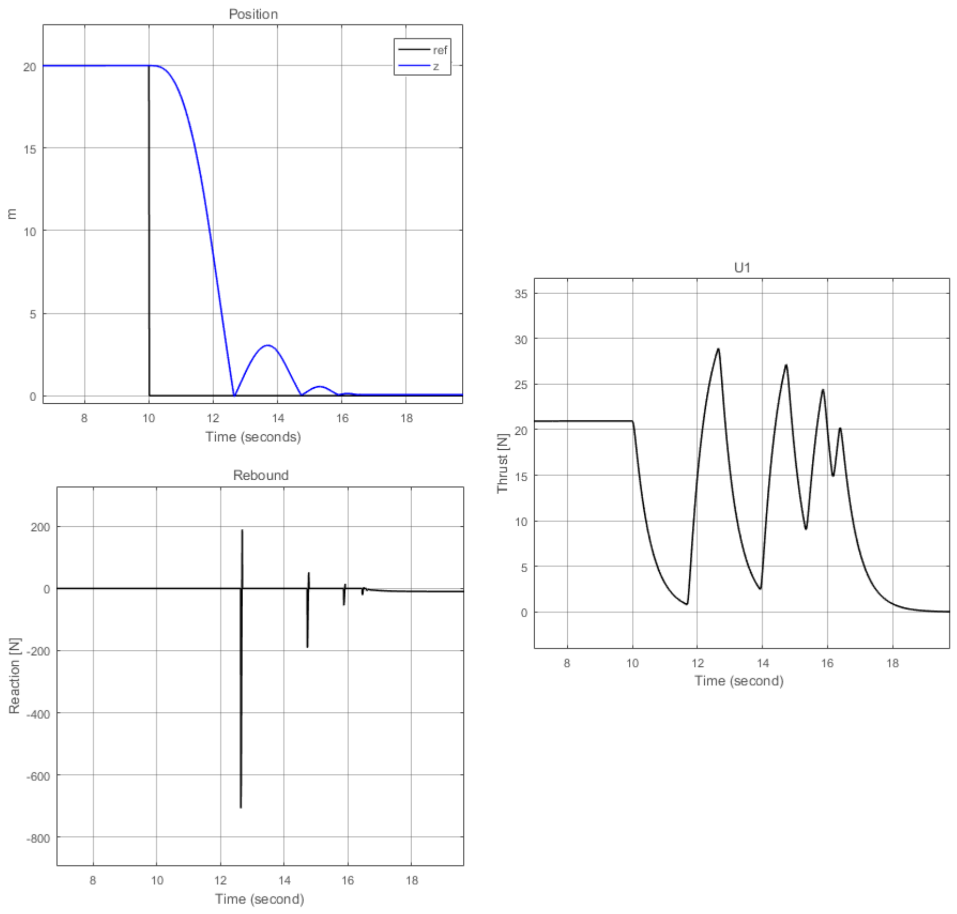


Figure 4.23: Fall, touchdown and rebound: Altitude variation, reaction to impact and control action U_1.

The control action is expected to oscillate, given the previous results acquired. As for the altitude – top left – and the reaction graphs – bottom left – the rebounds and spikes amplitudes coincide between each other.

On a further detail, the moment when the quadrotor settles on top of the target is illustrated on Fig. 4.24.

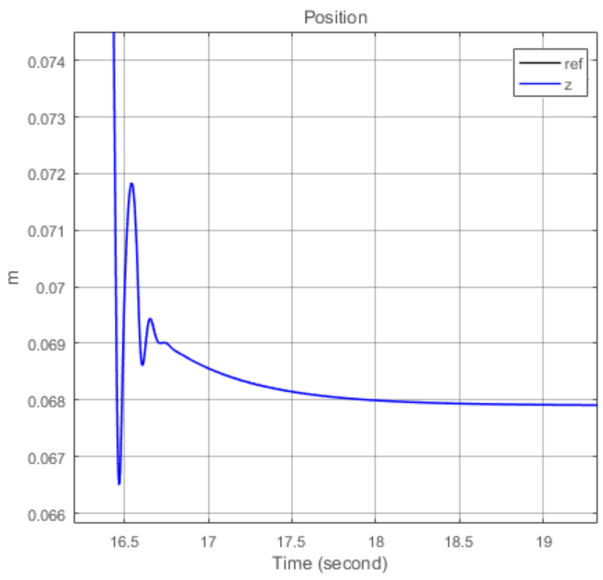


Figure 4.24: Fall, touchdown and rebound: rest.

A prolonged damping is noticed on the figure above due to the motors dynamics. After the moment they are turned off, the rotor blades have kinetic energy stored, causing them to keep rotating, thus generating thrust. When this energy dissipates, the rotors no longer generate thrust, and so the gravity force and reaction remains in the system. Because of the quadrotor's legs and the stiffness and damping coefficients chosen on Chapter 3, the quadrotor base will settle at 0.07 meters height. However, because the legs are not perfectly rigid, they will deform roughly 0.002 meters.

4.2 Disturbances – zero air speed

In this section, the air drag that opposes to linear motion is considered. For no specific reason, the coefficient K_v, that establishes a relation between the quadrotor and air velocities, is set to one. For simplicity, it is assumed null wind velocity. For a better comparison between results, the setpoints applied in this section are the same as the setpoints applied in the previous one.

4.2.1 Position

4.2.1.1 Altitude displacement

The first simulation results with disturbance regard is focused on altitude. Fig. 4.25 the position and velocity graphs are presented, followed by Fig. 4.26, where the altitude control actuation is shown.

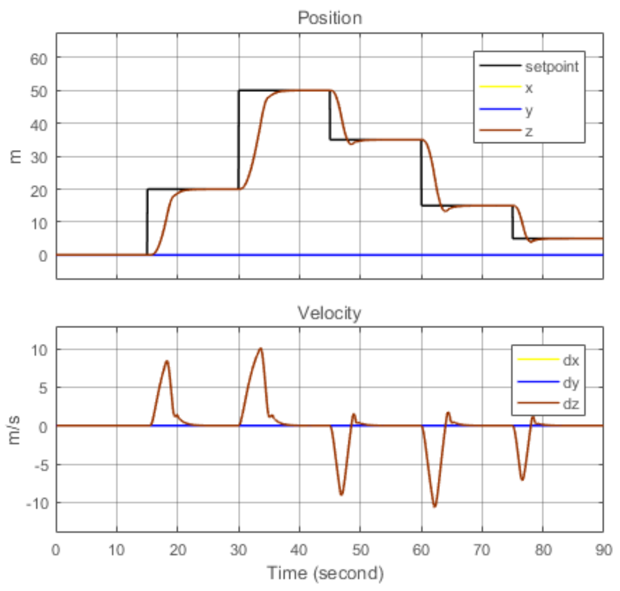


Figure 4.25: Translational response to a variable altitude setpoint. Air drag is considered.

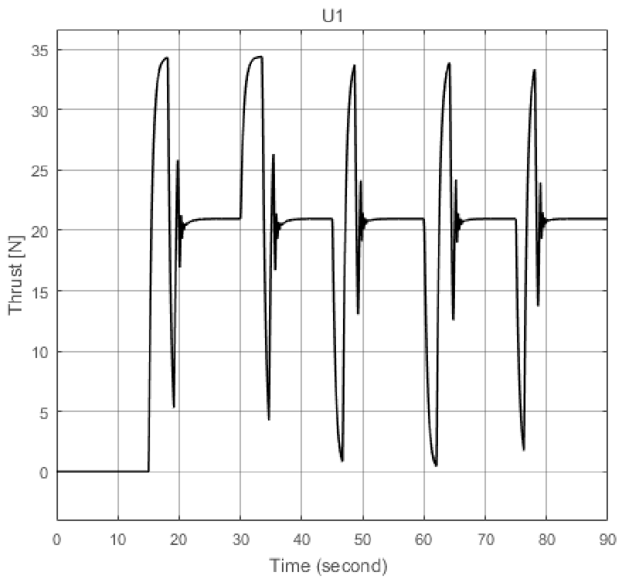


Figure 4.26: Control action in response to altitude error. Air drag is considered.

With the air resistance influence, the altitude variation is smoother than the case with disturbance rejection. This air drag helps the quadrotor to slow down, improving the trajectory control. In addition, the control action U_1 is less expressive considering both cases.

4.2.1.2 X displacement

The horizontal motion with air resistance effect is now considered. Trajectory control and actuation results are shown on Fig. 4.27 and Fig. 4.28, respectively.

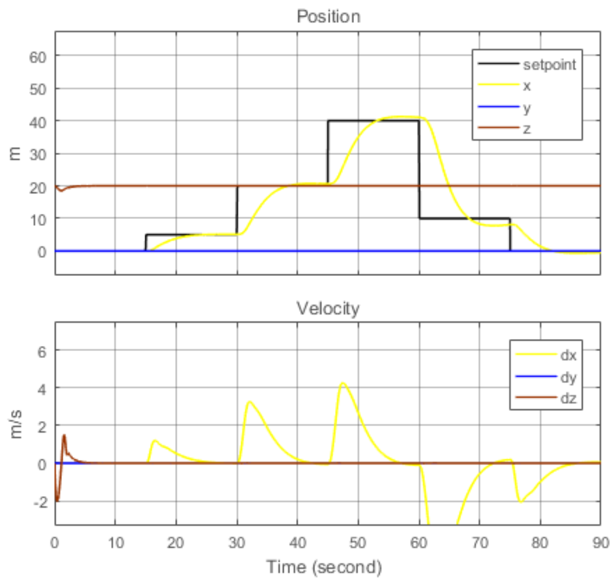


Figure 4.27: Translational response to a variable setpoint applied to X axis. Air drag is considered.

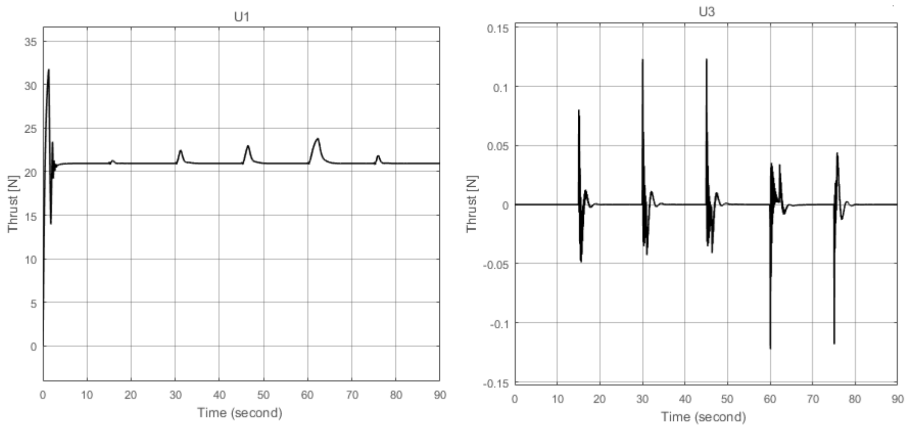


Figure 4.28: Control actions in response to X error. From left to right, altitude and pitch control actions. Air drag is considered.

Comparing the results from the previous section with graphs confronted *supra*, no significant changes are noticed. The implemented controllers are capable of rejecting small disturbances.

4.2.1.3 X, Y and Z displacement

For the final simulation performed on translational tracking control, a step signal is applied to each of the Cartesian axes. The results are implicit on the subsequent Fig. 4.29 and Fig. 4.30.

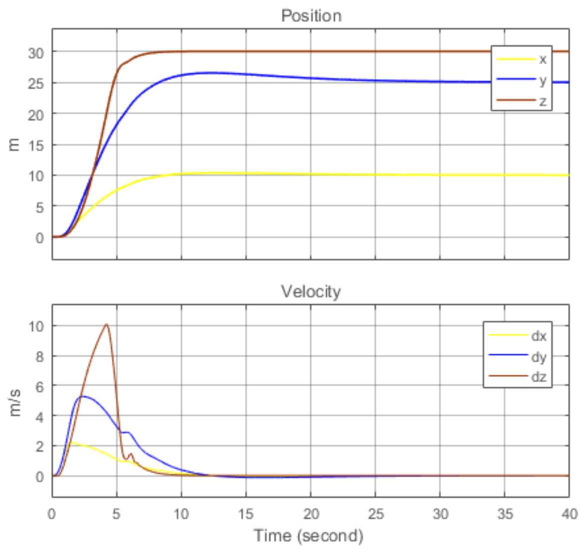


Figure 4.29: Translational response to a step input signal applied to X, Y and Z axes. Air drag is considered.

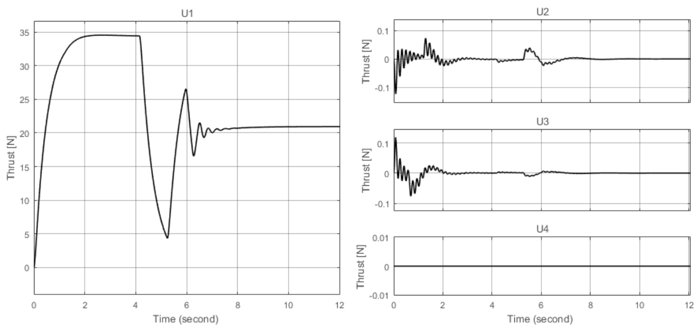


Figure 4.30: Control actions in response to X, Y and Z error. From left to right, altitude and attitude control actions. Air drag is considered.

The previous graphs denotes discrepancy, in comparison with the results from the previous section, regarding altitude response and control actions U_2 and U_3. The controllers' actuation is smoother in the presence of air drag, which is a good indicator considering that it is a situation closest to a real environment.

4.2.2 Touchdown

To conclude this sequence of simulation events and to close this chapter, the touchdown event is tested with the influence of air resistance. Fig. 4.31 describes the altitude variation, the reaction to the rebound and the control action.

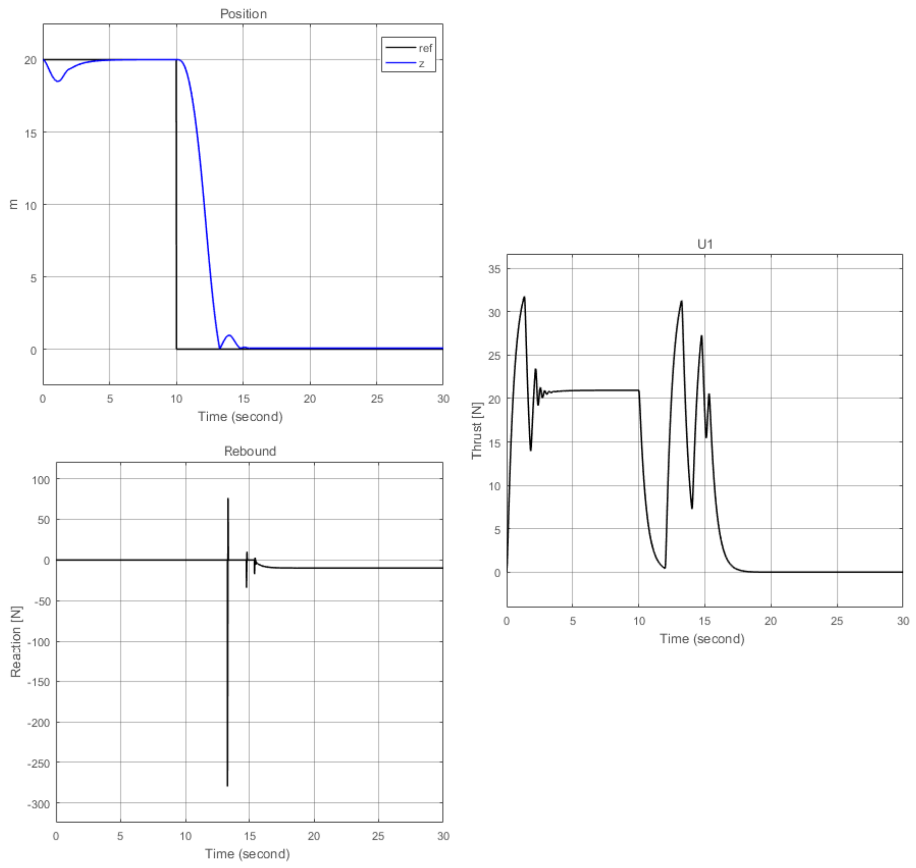


Figure 4.31: Fall, touchdown and rebound: Altitude variation, reaction to impact and control action U_1. Air drag is considered.

The graphs above simply describe a smoother rebound and altitude control actuation in the presence of air drag.

In conclusion, in any of the tests conducted on this section, the system loses its stability. In fact, in some cases, it benefits from this air resistance, providing a better tracking control and more stabilized actuation over the system.

5 Virtual environment

The next step of this work consists in the development of a virtual world, so tests can be conducted on the equations that define the quadrotor dynamics before a real experiment takes place. This environment is composed by basic features such as the quadrotor 3D model, the surface and few obstacles.

First, it is presented the comparison tables between different 3D modelling and simulation software to aid in the decision-making of each software to use. Lastly, the 3D model and the virtual environment project are exposed, respectively.

5.1 Software synthesis

To decide which software is suitable for this work, both professional and subjective opinions were considered. Tables 5.1 and 5.2 enhance the features of each software. Here, the more filled is the bar, more positive is the respective feature for this work purposes.

Table 5.1: Characteristics table of different 3D modelling tools.

	MAYA	3DS MAX	BLENDER
Learning Curve	▓▓▓▓░░░░░░	▓▓▓▓▓░░░░░	▓▓▓▓░░░░░░
Design	▓▓▓▓▓▓▓░░░	▓▓▓▓▓▓▓▓▓░	▓▓▓▓▓▓▓▓▓░
Materials Creation and Edition	▓▓▓▓▓▓▓░░░	▓▓▓▓▓▓▓░░░	▓▓▓▓▓▓▓▓▓░
Modelling	▓▓▓▓▓▓▓░░░	▓▓▓▓▓▓▓▓▓░	▓▓▓▓▓▓▓▓▓░
Simulation	▓▓▓▓▓▓▓▓▓░	▓▓▓▓▓▓▓▓▓░	▓▓▓▓▓▓▓▓▓░
Licence	Free for students	Free for students	Free

Table 5.2: Characteristics table of different Game Engines.

	UNITY	UNREAL
Learning Curve	▓▓▓▓▓▓▓░░░	▓▓▓▓▓▓▓░░░
Graphics	2D / 3D	2D / 3D
Programming Languages	C# / Javascript	C++ / Blueprint Visual Scripting
Add-ins	▓▓▓▓▓▓▓▓▓░	▓▓▓▓▓▓▓░░░
Supported Formats	▓▓▓▓▓▓▓▓▓░	.fbx only
Licence	Paid / free with less features	Free for non-commercial purposes

Based on the previous tables, showing off the software key features according to what is necessary for this work, decisions were made. Blender is the modelling tool chosen due to its general good features and it is free for any applications. Regarding game engines, the chosen one is the Unreal Engine because it offers two different programming methods and it is free, considering the purpose of this research is to provide knowledge rather than benefit monetarily from it.

5.2 Blender

In this section, the quadrotor 3D model assembled on Blender software, version 2.78, is presented. For this model, are used the 3D pieces created by Brito.

5.2.1 Skeletal Mesh

The skeletal mesh is an object with several attributes and which can be ruled by physics laws. These attributes are polygons and bones. The first ones represent the object visuals. The second allows deformations of the objects parts, enabling animations. A static mesh is only composed by the first set of attributes and cannot be deformed in any way.

5.2.2 Quadrotor Mesh

The quadrotor 3D model is, in its hole, a skeletal mesh due to its propellers rotations, which are assembled to meet this specificity.

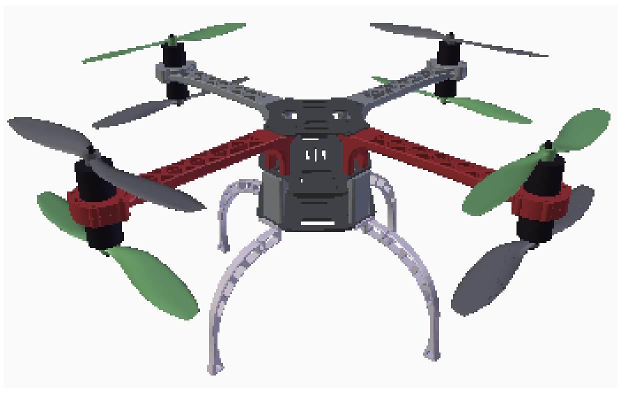


Figure 5.1: Quadrotor 3D model assembled in Blender.

The quadrotor illustrated on Figure 5.1, is visually the same as the one assembled by Brito's on Autodesk 123D. The main differences reside on the dynamics properties. This skeletal mesh contains bones: one on its center body and one on each propeller, to enable the respective rotation. It also has built-in rigid body dynamics, empowering the ability to rotate and move linearly with respect to its inertial parameters and geometry, and constraints, keeping each leg attached to the body. Because it is a skeletal mesh, it can also be used as a static mesh on a game engine, if simplicity is required.

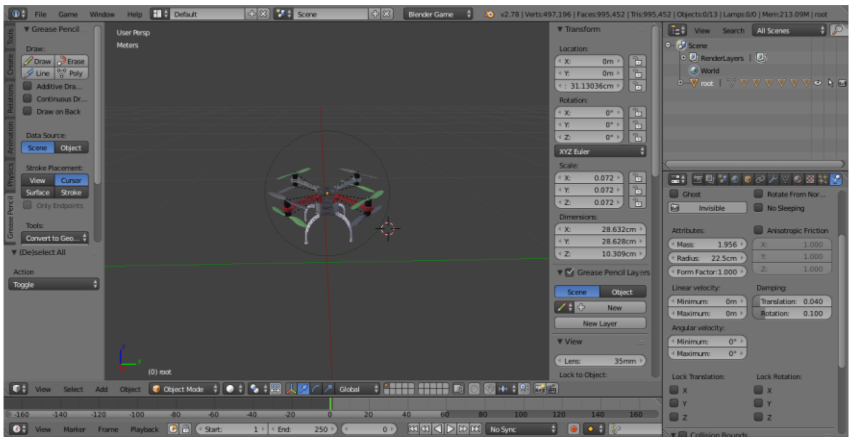


Figure 5.2: Example of Blender v2.78 workspace.

Fig. 5.2 illustrates a possible Blender workspace set. On the middle, is the scene, or viewport. On the top, is the information bar and, on the bottom, are shown the 3D view editor and the timeline and playback controls. On the right, the objects editor, the overview of scene graph and all available data-blocks are represented. Finally, on the left side, a general scene and object editor is shown.

5.3 Unreal

Once the 3D model assembly is complete, it is essential to integrate it on a simulation platform. The Unreal Engine is the software chosen to develop the virtual environment responsible for testing and simulate the quadrotor kinematics model.

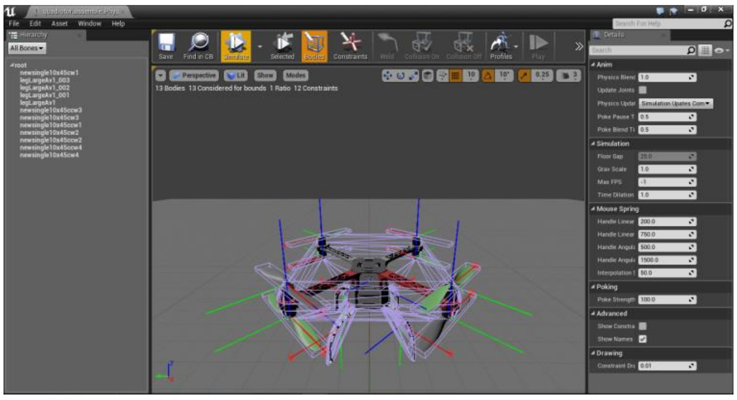
Blueprints visual scripting is an object-oriented language, such as C++, and is the language used the most in this project. But for specific functionalities, C++ code is integrated.

5.3.1 Project development

To develop a simulator, or a game as it is named in this field, in any game engine, a natural sequence of events must happen: choose the main scripting language; create the scene; import the object and specify its role in the scene; supply the object with attributes and place it on the game viewport; design the event graph; and, lastly, simulate the project on the game viewport. This sequence is the one adopted throughout the development of this project, helped by an existing template granted by the Epic Games.

Firstly, a new project is created. Templates are available to help on the initial stage of projects and so, as referred before, one is applied to this project. With this template, the scenery with the ground and the obstacles is already created.

Secondly, the quadrotor 3D model is imported. It is defined as a pawn class, meaning that the object, hereby called actor, can be controlled and receive input commands from a controller, be it an intelligent controller or a user. For this work purposes, the actor is controlled exclusively by the user. After the definition of the class, some preparation is needed. Fig. 5.3 illustrates the environment where the collision boundaries are set, constraints and gravity applied, inertial quantities manually set and other physics properties, which are not considered, can be adjusted.



root
- newsingle10x45cw1
- legLargeAv1_003
- legLargeAv1_002
- legLargeAv1_001
- legLargeAv1
- newsingle10x45ccw3
- newsingle10x45cw3
- newsingle10x45ccw1
- newsingle10x45cw2
- newsingle10x45ccw2
- newsingle10x45ccw4
- newsingle10x45cw4

13 Bodies 13 Considered for bounds 1 Ratio 12 Constraints

Anim
- Physics Blend: 1.0
- Update Joints: ☐
- Physics Updat: Simulation Upates Com
- Poke Pause T: 0.5
- Poke Blend Ti: 0.5

Simulation
- Floor Gap: 25.0
- Grav Scale: 1.0
- Max FPS: -1
- Time Dilation: 1.0

Mouse Spring
- Handle Linear: 200.0
- Handle Linear: 750.0
- Handle Angula: 500.0
- Handle Angula: 1500.0
- Interpolation S: 50.0

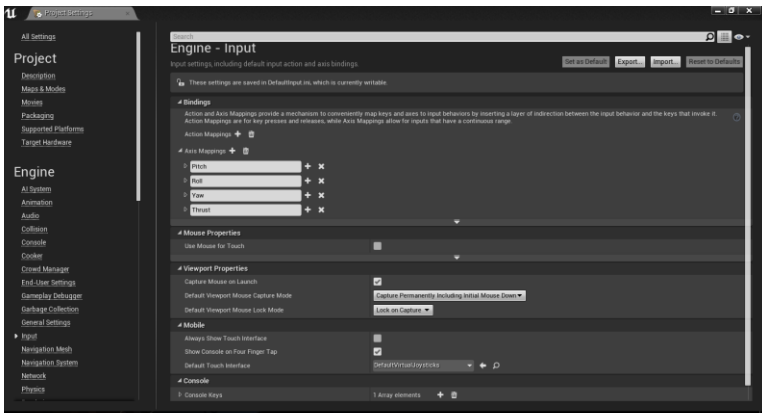
Poking
- Poke Strength: 100.0

Advanced
- Show Constra: ☐
- Show Names: ✓

Drawing
- Constraint Dra: 0.01

Figure 5.3: Quadrotor physics assembly. Collision detection and other physics considerations are embodied for simulation.

The third step regards input commands and actor dynamics. Inputs are events that are triggered if a change in state occurs or simply at every frame step, if desired. Any input event must be previously defined, but other types of events are available, e.g., events triggered only at the beginning of the simulation, when actor hits an obstacle, amongst others. Another advantage, is the possibility to manipulate files. One can read the input values from a file and write the outputs to another, opening doors interoperability with other software. With these events, a dynamic system can be built where the outputs are functions of the inputs events. On Fig. 5.4, the window where the interaction with the user is set is presented. In the project, four input commands are defined: roll, pitch, yaw and thrust. These are recognized as hardware events. The hardware used to control the action is the keyboard, as it is the most accessible.



Engine - Input

Input settings, including default input action and axis bindings.

These settings are saved in DefaultInput.ini, which is currently writable.

Bindings

Action and Axis Mappings provide a mechanism to conveniently map keys and axes to input behaviors by inserting a layer of indirection between the input behavior and the keys that invoke it. Action Mappings are for key presses and releases, while Axis Mappings allow for inputs that have a continuous range.

Action Mappings

Axis Mappings
- Pitch
- Roll
- Yaw
- Thrust

Mouse Properties

Use Mouse for Touch	☐

Viewport Properties

Capture Mouse on Launch	✓
Default Viewport Mouse Capture Mode	Capture Permanently Including Initial Mouse Down
Default Viewport Mouse Lock Mode	Lock on Capture

Mobile

Always Show Touch Interface	☐
Show Console on Four Finger Tap	✓
Default Touch Interface	DefaultVirtualJoysticks

Console

Console Keys	1 Array elements

Figure 5.4: Project inputs. Definition of the input commands and the hardware from which they are sent.

Any animation that happens in the game viewport, is preceded by a blueprints graph developed in event graph space. In this space, the system dynamics is placed and corresponds to the one represented by eq. (3.2.14). Functions and macros can be defined on the constructor graph, shown on Fig. 5.5, and be used on the event graph to simplify the blocks diagram. Additionally, values can be converted to strings and then printed on the game viewport. This way, it is easier to track minor dynamics issues, which may not be perceptible by analysing the quadrotor behaviour alone, and to monitor simple parameters, e.g., position, velocity, accelerations, and others.

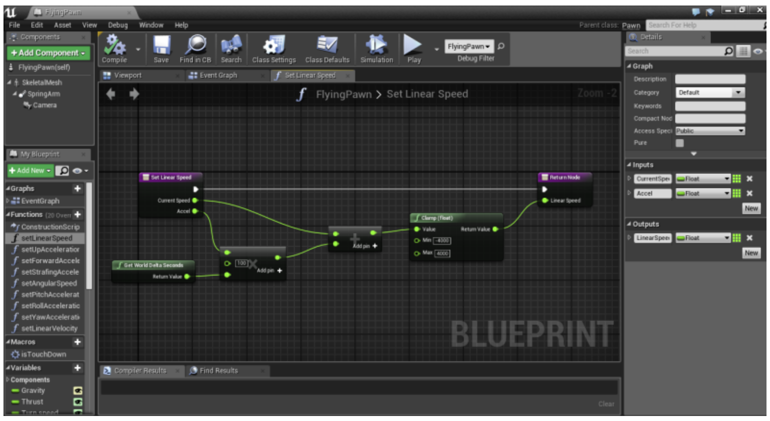


Figure 5.5: Set linear speed function on the constructor graph. Event graph has similar representation.

To conclude, the testing step is composed by a compilation of all the code designed on the graph event, followed by the test itself. All the steps aforementioned, explained in a simplified way, are part of the game development process. In the game viewport, these processes can be visualized in the form of 3D draws. An example set for the game viewport is illustrated on Figure 5.6.

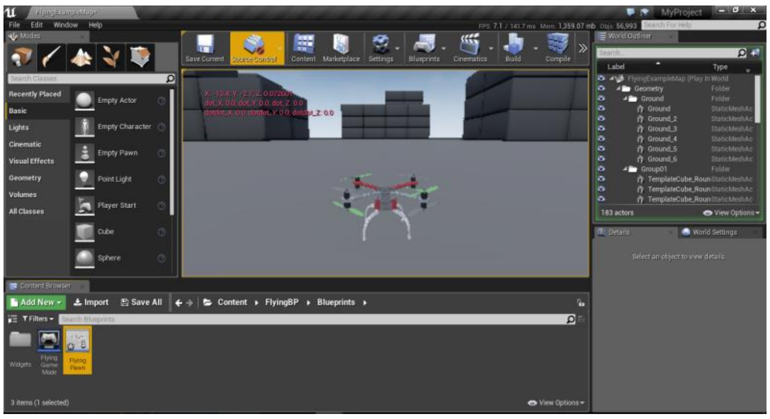


Figure 5.6: Game viewport. The quadrotor 3D model and the scene are represented.

Intrinsically, the control is produced through the hardware inputs and the rotation is simply a visual detail. Therefore, from the control point of view, only one propeller is considered. For simplicity, in this version, it is not conceived the the propellers rotation, individually. Notwithstanding, this is possible by associating to each propeller a scene component class, allowing its own rotation.

5.3.2 External actuation

The input commands are a gateway to allow an interoperability relationship between both engine and outside world, i.e., human. Is, then, important to understand how to set this relation. Fig. 5.7 illustrates the command flow from the moment the player triggers an input event through the selected hardware, to the moment when this action is perceived by the pawn component in the game.

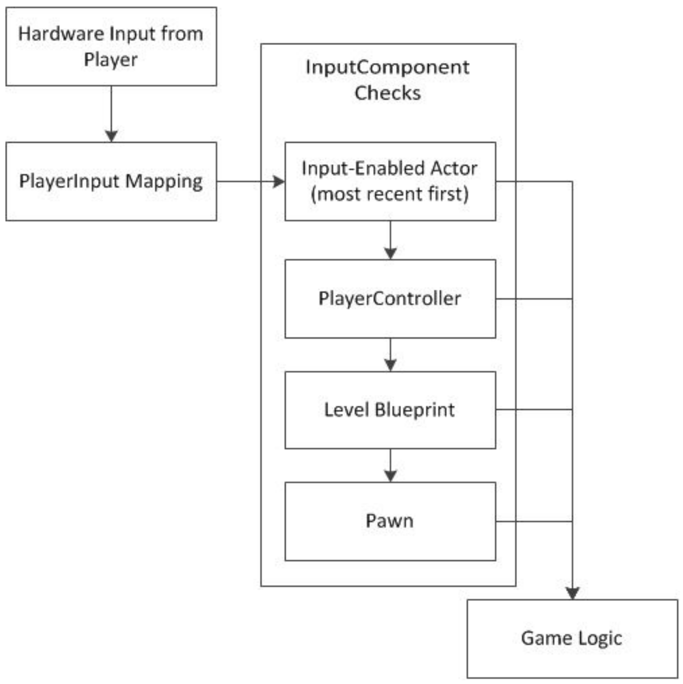


Figure 5.7: Input process flow. From the player actuation to the resultant game action. Retrieved from (Epic Games 2017).

As mentioned before, the software enables a communication to the exterior through the input settings. A panoply of hardware is compatible with the software, of which are included the keyboard, the gamepad, an Android device, amongst others. For this work, only the keyboard hardware is used to manipulate the pawn.

Inside the blueprints editor, another simple method is available. It is possible to load data from files and to store it as well. This can be done by importing a plugin, supported by Epic Games and at a cost for the user, or by integrating a C++ code, which is free for the user. In this work, a C++ project is integrated and pawn data is stored in text files, although it is possible to store it in several other file formats.

Another method to actuate the pawn and all the components in the game is through TCP/IP communication. Although it allows a wireless communication, accentuated delays may occur, which compromise the control over the scene.

6 Conclusions and Future work

In this chapter, are presented the limitations found on the control system proposed and on Unreal game engine. A conclusion of the work developed is also given, with the description of what needs to or can be improved in future developments.

6.1 Control system limitations

Two major limitations are found on the overall system. The first relates to the PWM / force conversion. The conversion has some flaws concerning the output of each conversion. On some occasions, this conversion would cause the output force quantity to exceed the by far the upper and lower limits defined. Because of this, it was not applied in this work. This problem is possibly related to precision. A better description of the functions involved is needed.

The same issue occurred with the function description of the one-rotor and two-rotors configuration functions. The solution is possibly the same as the former one mentioned.

The PSO algorithm was found difficult to implement when the system complexity increased. The solutions did not converge, or the local minimum would not guarantee the stability of the closed-loop systems. However, when a solution would converge and place the closed-loop on the region of stability, the results were satisfactory, as the simulations prove for the attitude controllers. For better selection of the PSO algorithm parameters it may be useful to see (Clerc and Kennedy 2002).

6.2 Unreal Engine blueprints limitations

One limitation found stands at monetary level. The files manipulation is possible in C++, but in blueprints scripting language this feature is paid. Although, a simple C++ code integration is possible.

A second limitation concerns the partition between world and actor rotations. Attempts were made to include the "Euler angles rate" -to- "body axes rate" conversion matrix, but to no avail. The Epic Games provides documentation about these relations, but the integration was not possible.

6.3 Work synthesis

The main purpose and contribution of this thesis was the development of a 3D virtual environment. With this tool, experiments can be conducted on the quadrotor model instead of the real structure, decreasing the chances of disabling it. A second goal was a description of some of the problems noticed when a quadrotor approaches the target. Three of them are described in this work.

Additionally, a major improvement to the simplified model is the introduction of the body axes. With this upgrade, the three fundamental rotations can be performed simultaneously without driving the system to instability. This achievement is crucial while performing landing manoeuvres, so the quadrotor may be able to tolerate any form of disturbances.

It was found, through simulations, that ground effect is not a major threat to the quadrotor stability while landing, Hooke's law proved to be a useful tool in the description of the quadrotor impact with a surface and it was found that the air resistance can lighten the controllers' actuation, which ultimately reduces energy consumption.

To conclude, the Unreal Engine has proved to be an interactive and enjoyable way to learn and to develop work experiments. It is, with no doubt, a useful tool to continue to invest in the future.

6.4 Future work

Although some work has been conducted in this thesis, some subjects were left to complete and others were not included in the project objectives, but can also be studied. The main ones are described below.

The control design was accomplished by removing the nonlinearities of the system. In future works, the nonlinear system may be described as a sum of linearities, without the need to simplify the model.

The controllers' gains could be obtained through different control techniques which could produce better controllers.

Deeper study about the influence of air resistance on the quadrotor flight.

Develop a ground effect model for the quadrotor case.

Develop control to enable the quadrotor landing over moving surfaces.

Continue the work conducted on Unreal Engine.

References

Aich, Sanjukta, Chahat Ahuja, Tushar Gupta, and P Arulmozhivarman. 2014. "Analysis of Ground Effect on Multi-Rotors." *Icecce*, 236–41. doi:10.1109/ICECCE.2014.7086619.

ArduPilot Dev Team. 2016. "Ground Effect Compensation — Copter Documentation." *Ardupilot*. http://ardupilot.org/copter/docs/ground-effect-compensation.html.

Astrom, Karl Johan;, and Bjorn Wittenmark. 1996. "Computer Control System Theory and Design." *Prentice Hall*, 555. doi:10.1002/1521-3773(20010316)40:6<9823::AID-ANIE9823>3.3.CO;2-C.

Astrom, KJ. 1995. "PID Controllers: Theory, Design and Tuning." *Instrument Society of America*. doi:1556175167.

Autodesk. 2016. "Comparison of 3ds Max and Maya | 3ds Max | Autodesk Knowledge Network." https://knowledge.autodesk.com/support/3ds-max/troubleshooting/caas/sfdcarticles/sfdcarticles/Comparison-of-3ds-Max-and-Maya.html.

Bemporad, Alberto. 2011. "Automatic Control 2." *University of Trento*.

Bennett, S. 1996. "A Brief History of Automatic Control." *Control Systems, IEEE* 16 (3): 17–25. doi:10.1109/37.506394.

blender.org. 2015. "Blender.org - Home of the Blender Project - Free and Open 3D Creation Software." *Blender.org*. https://www.blender.org/.

Brito, Vasco. 2016. "Fault Tolerant Control of a X8-VB Quadcopter." Universidade Nova de Lisboa.

Buyukkabasakal, Kemal, Baris Fidan, Aydogan Savran, and Nasrettin Koksal.

2015. “Real-Time Implementation of Mixing Adaptive Control on Quadrotor UAVs.” *2015 European Control Conference, ECC 2015*, 3597–3602. doi:10.1109/ECC.2015.7331089.

Carrasquinho, Filipe. 2015. “Ferramenta de Simulação Para Robot Industrial.” Universidade Nova de Lisboa.

Cheeseman, I. C., and W. E. Bennett. 1957. “The Effect of the Ground on a Helicopter Rotor in Forward Flight,” no. 3021: 2. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.226.6371.

Clerc, Maurice, and James Kennedy. 2002. “The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space.” *IEEE Transactions on Evolutionary Computation* 6 (1): 58–73. doi:10.1109/4235.985692.

Davis, Edwin, and Paul Pounds. 2016. “Passive Position Control of a Quadrotor with Ground Effect Interaction.” *IEEE Robotics and Automation Letters* 3766 (c): 1–1. doi:10.1109/LRA.2016.2514351.

Eberhart, R., and J. Kennedy. 1995. “A New Optimizer Using Particle Swarm Theory.” In *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39–43. Nagoya, Japan: IEEE. doi:10.1109/MHS.1995.494215.

Edaris, Z.L., and S. Abdul-Rahman. 2016. “Performance Comparison of PID Tuning by Using Ziegler-Nichols and Particle Swarm Optimization Approaches in a Water Control System.” *Journal of Information and Communication Technology* 15 (1).

Epic Games. 2017. “Unreal Engine 4 Documentation.” *Unreal Engine 4 Documentation*. https://docs.unrealengine.com/latest/INT/.

Federal Aviation Administration. 2016. *Pilot ’ S Handbook of Aeronautical Knowledge*. *Pilot’s Handbook of Aeronautical Knowledge*. Oklahoma.

Gibiansky, Andrew. 2012. “Quadcopter Dynamics , Simulation , and Control,” 1–18. http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/.

Herissé, Bruno, Tarek Hamel, Robert Mahony, and Francois-Xavier Russotto. 2012. “Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow.” *IEEE Transactions on Robotics* 28 (1): 77–89. doi:10.1109/TRO.2011.2163435.

Kennedy, J, and R Eberhart. 1995. “Particle Swarm Optimization.” *IEEE International Conference on Particle Swarm Optimization* 4: 1942–48. doi:10.1109/ICNN.1995.488968.

Landau, Ioan Doré, Rogelio Lozano, Mohammed M’Saad, and Alireza Karimi.

2011. *Adaptive Control. Analysis and Applications*. doi:10.1007/978-0-85729-664-1.

Lee, Daewon, Tyler Ryan, and H. Jin Kim. 2012. “Autonomous Landing of a VTOL UAV on a Moving Platform Using Image-Based Visual Servoing.” *Proceedings - IEEE International Conference on Robotics and Automation*, 971–76. doi:10.1109/ICRA.2012.6224828.

Mayden, Austin. 2014. “Unreal Engine 4 vs. Unity: Which Game Engine Is Best for You?” *Digital Tutors*. http://blog.digitaltutors.com/unreal-engine-4-vs-unity-game-engine-best/.

Santos, Milton, Claudio Rosales, Jorge Sarapura, and Ricardo Carelli. 2017. “Adaptive Dynamic Control for Trajectory Tracking with a Quadrotor,” 547–53.

Serra, P., R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre. 2016. “Landing of a Quadrotor on a Moving Target Using Dynamic Image-Based Visual Servo Control.” *IEEE Transactions on Robotics* 32 (6): 1524–35. doi:10.1109/TRO.2016.2604495.

Sharf, I., M. Nahon, A. Harmat, W. Khan, M. Michini, N. Speal, M. Trentini, T. Tsadok, and T. Wang. 2014. “Ground Effect Experiments and Model Validation with Draganflyer X8 Rotorcraft.” *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, 1158–66. doi:10.1109/ICUAS.2014.6842370.

Stengel, Robert. 2016. “Aircraft Equations of Motion : Translation and Rotation !” *Aircraft Flight Dynamics, ! MAE 331*, 48. https://www.princeton.edu/~stengel/MAE331Lecture8.pdf%5Cnhttp://www.princeton.edu/~stengel/FlightDynamics.htm.

Supernat. 2012. “What Is the Difference between 3ds Max and Blender?” *unity3d Forum*. https://forum.unity3d.com/threads/what-is-the-difference-between-3ds-max-and-blender.133112/.

Tay, Tiffany. 2014. “3DS Max vs Maya: A Friendly Comparison.” https://blog.udemy.com/3ds-max-vs-maya/.

Todorov, Emanuel. 2006. “Optimal Control Theory.” *Environment and Planning C Government and Policy* 4 (2): 1–28. doi:10.1068/c040121.

Yadav, Smriti, and BhuriaVijay. 2015. “Tuning of PID Controller Using Zeigler Nichols and Particle Swarm Optimization in AVR System” 4 (10): 1984–88.

Yang, Patrick Woo Ker. 2016. “What's the Difference between AutoDesk's Maya and 3DS MAX? - Quora.” https://www.quora.com/Whats-the-difference-between-AutoDesks-Maya-and-3DS-MAX.

Zairi, Shaiful, and D. Hazry. 2011. “Adaptive Neural Controller Implementation in Autonomous Mini Aircraft Quadrotor (AMAC-Q) for Attitude Control Stabilization.” *Proceedings - 2011 IEEE 7th International Colloquium on Signal Processing and Its Applications, CSPA 2011*, 84–89. doi:10.1109/CSPA.2011.5759848.

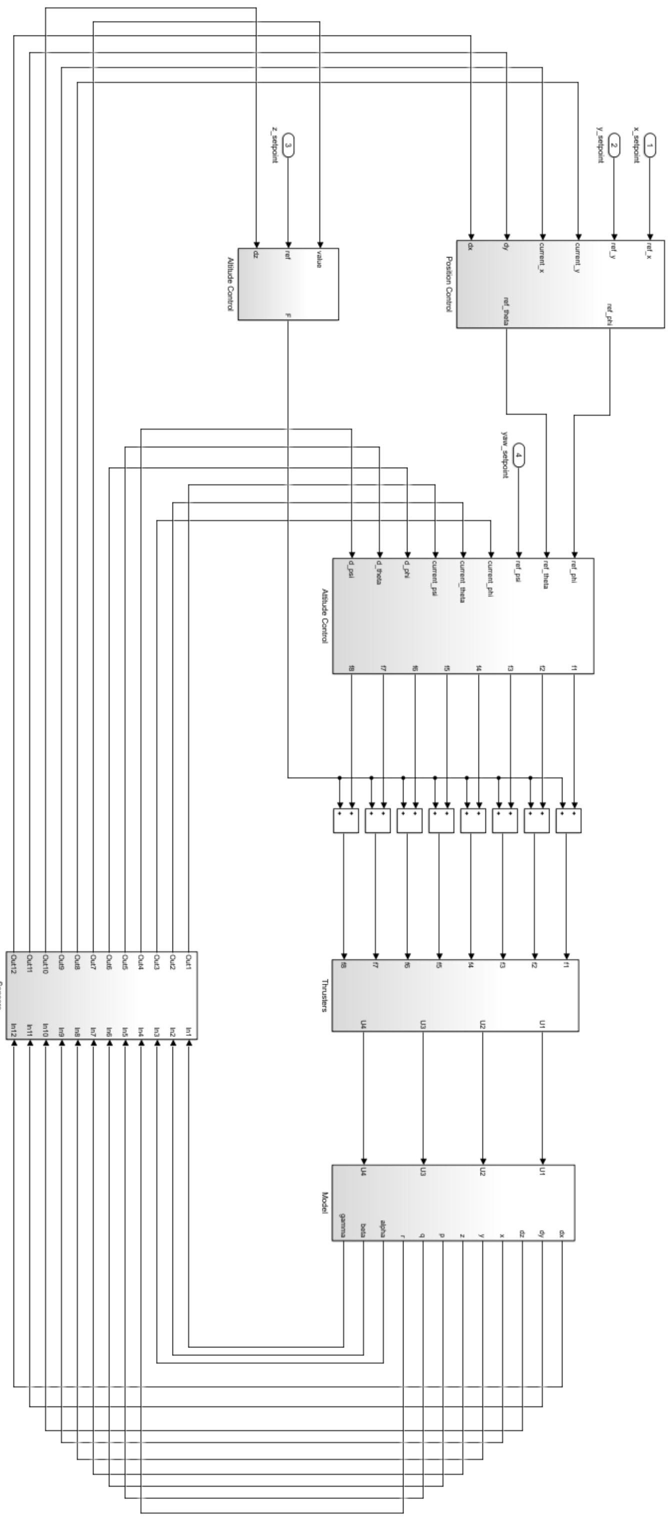
Ziegler, J. G., and N. B. Nichols. 1995. “Optimum Settings for Automatic Controllers.” *InTech* 42 (6): 94–100. doi:10.1115/1.2899060.

Attachments

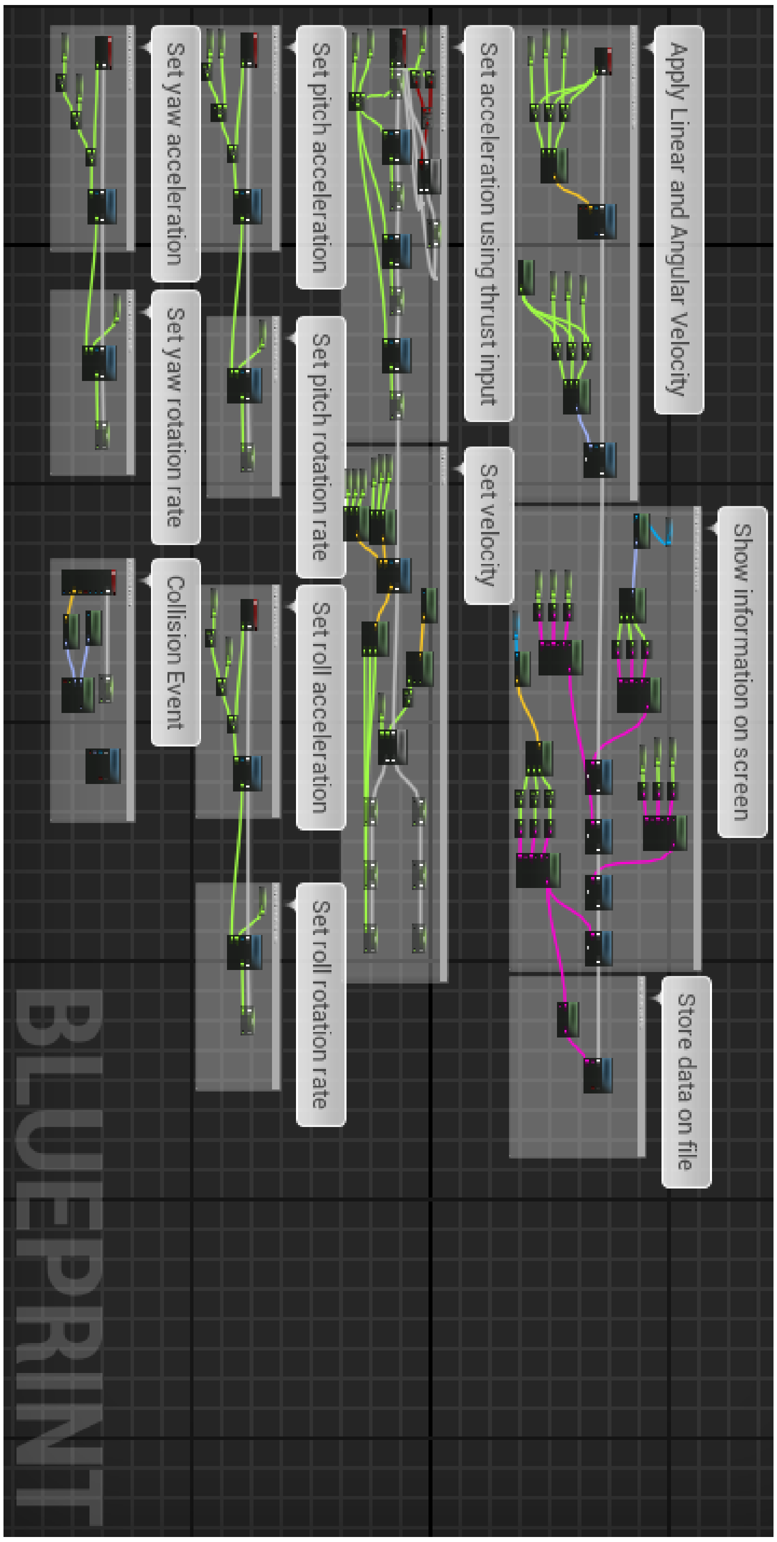
Attachment A – Quadrotor Parameters

Parameter	Value	Unit
I_{xx}, I_{yy}	0.0060	$kg.m^2$
I_{zz}	0.0166	$kg.m^2$
J_r	$4.104x0^{-6}$	$kg.m^2$
M_{body}	1.5	kg
M_{motor}	0.057	kg
$M_{propeller}$	0.015	kg
M_{leg}	0.015	kg
h_{body}	0.07	m
h_{leg}	0.07	m
l_{arm}	0.225	m
$Pitch_{propeller}$	$4.5x10^{-3}$	m
$r_{propeller}$	0.1	m

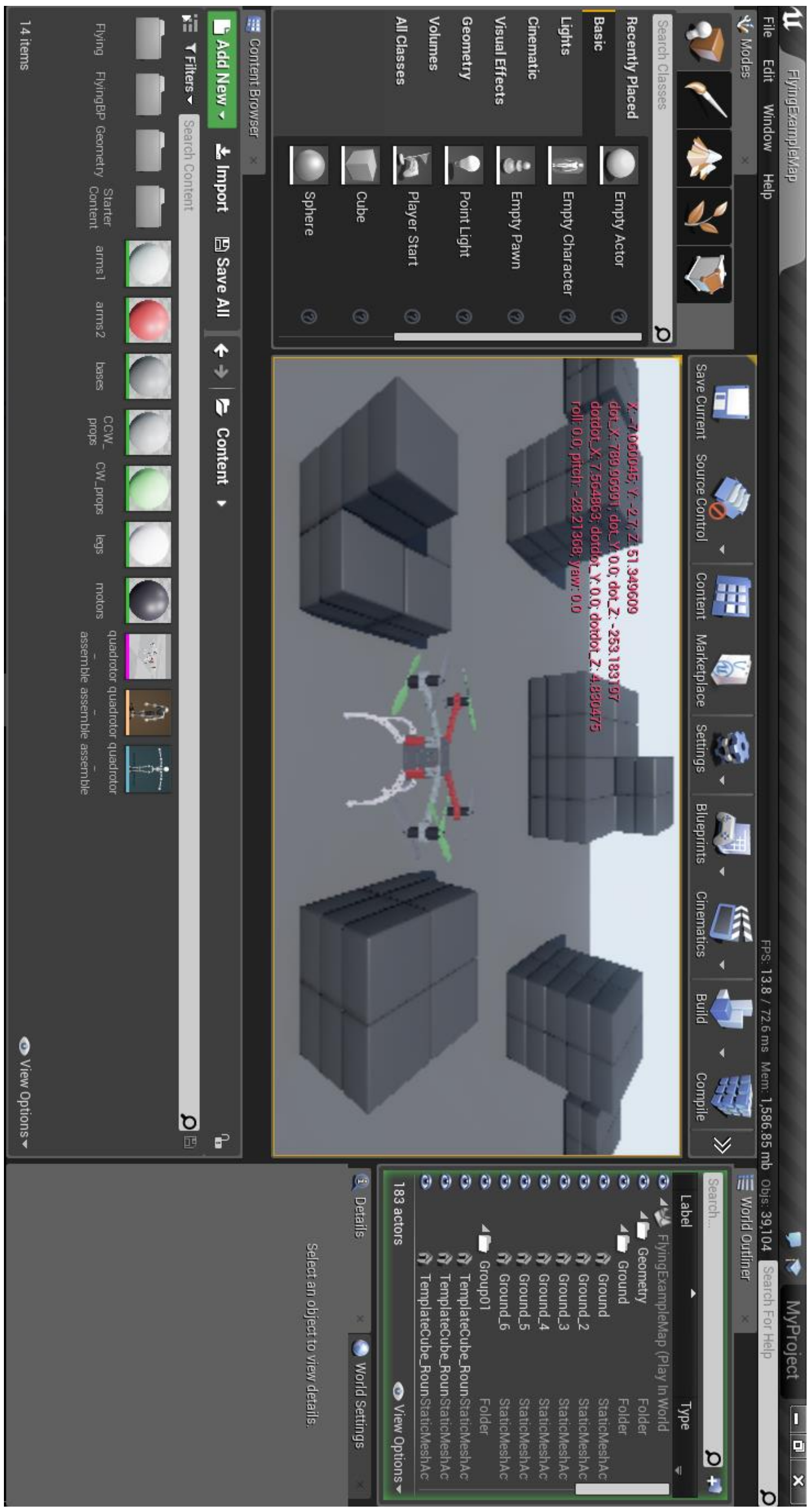
Attachment B – Simulink Project



Attachment C – Unreal Engine Project



Attachment D – Unreal Engine Gameplay



```
X: -7.060045; Y: -2.7; Z: 51.349609
dot_X: 789.96991; dot_Y: 0.0; dot_Z: -253.183197
dotdot_X: 7.564863; dotdot_Y: 0.0; dotdot_Z: 4.830475
roll: 0.0; pitch: -28.21368; yaw: 0.0
```

Label	Type
FlyingExampleMap (Play In	World
Geometry	Folder
Ground	Folder
Ground	StaticMeshAc
Ground_2	StaticMeshAc
Ground_3	StaticMeshAc
Ground_4	StaticMeshAc
Ground_5	StaticMeshAc
Ground_6	StaticMeshAc
Group01	Folder
TemplateCube_Roun	StaticMeshAc
TemplateCube_Roun	StaticMeshAc
TemplateCube_Roun	StaticMeshAc

183 actors