TÉCNICO LISBOA

Design and Control of a Bimodal aerial robot for locomotion in flat and inclined surfaces

Miguel Vicente Pimentel

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor: Dr. Meysam Basiri

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Dr. Meysam Basiri
Member of the Committee: Prof. Bruno João Nogueira Guerreiro

April 2021

To the ones who are and the ones who were.

Acknowledgments

I would like to start by thanking my parents, who always allowed me to dream and provided me with the best education I could hope for. Thank you, without your support this wouldn't be possible.

To Carolina, Guilherme and Francisco, who were there since I was a baby, thank you for accompanying me throughout all the steps of my life.

To Beatriz and Adriana, for helping me become who I am, thank you. To Duarte, thank you for putting up with all the stress and helping me along the way.

I'm fortunate enough to be surrounded by amazing people. In no particular order, Laura, Tiago, Sofia, Madalena, thank you for allowing me to disappear for weeks berried in work but still being there for me whenever I need. To my most recent friends, who passed from colleagues to very good friends Paulino, Mim, Tomas, Afonso, Edgar, my deepest thank you for always challenging me to do more and be better.

A special thanks to all my course colleagues, friends or not, it was quite a journey! To my IST, *Padova* and *Sapienza* professors, tutors, ISR researchers, thank you for inspiring and teaching me. To the whole team at CoLAB+Atlantic, thank you for being a beacon during this terrible year - I'm counting the days for this pandemic to be over so we can be back at the office.

To Inês, my godmother, thank you for all the advice you gave me over the past 5 years. I owe you a debt of gratitude greater than you can ever imagine.

To Malin, Josep, Nicole, Tim and Sofia, my dearest Erasmus friends, I miss you so much everyday and I hope we can be reunited soon enough.

To my family, thank you for the Sunday lunch's, the celebrations, the holidays and all the good times we spend as a family. We are very lucky to have each other! To everyone else, even if you're not mentioned here, you know you have a place in my heart. A deep deep thank you!

Last but not least, to my supervisor Meysam, thank you for pushing me to do my best and helping me in every step of the way.

Resumo

O uso de robôs aéreos não tripulados é extremamente útil para missões que envolvem inspeção e manutenção de estruturas. Devido às restrições impostas pelas plataformas aéreas comumente disponíveis, grande parte destas missões estão limitadas à percepção distante dos alvos, sem possibilidade de se aproximarem ou entrarem em contato físico com as estruturas. Este trabalho descreve um robô aéreo Bimodal, que consiste num *quadrotor* equipado com duas rodas sem transmissões, o que permite ao robô voar, aproximar-se, aterrar e mover-se em superfícies planas e inclinadas, que pode ser usado para inspeção enquanto em contacto com a superfície. Esta tese descreve em detalhe o modelo matemático do robô para os diferentes modos de navegação: voo, solo e superfícies inclinadas. Além disso, são propostos dois controladores automáticos diferentes, para os três modos de locomoção, sendo que o seu desempenho é avaliado através de uma série de experiências em ambiente de simulação, onde se testam as características híbridas do veículo.

Palavras-chave: Robôs Aéreos, Locomoção Híbrida, Controlo Automático, Multi-modalidade

Abstract

Employing aerial robots, acting as mobile airborne sensors, is extremely useful for many surveillance and inspection missions. Due to the strict constraints of commonly available aerial platforms, most aerial inspection missions are limited to distant perception of targets, without the possibility of coming into centimeter-range proximity or even in physical contact with them. This thesis describes a Bimodal aerial robot, consisting of a common quadrotor equipped with two passive wheels, that allows flying, approaching, landing and moving on planar and inclined surfaces, suitable for micro-level inspection of large areas. This work describes in detail the mathematical model of the passive two-wheeled quadrotor for its different modes of operation: flight, ground and inclined surfaces. Furthermore, two different motion controllers are proposed, for all three modes of locomotion, and their performance is evaluated through a series of simulated experiments that make use of the hybrid characteristics of the vehicle.

Keywords: Aerial Robots, Hybrid Locomotion, Motion Control, Multi-Modality

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α, β	Dynamic Compensator.
Δ	Variation.
η	Output.
γ	Inclination angle.
μ	Friction coefficient.
ω	Angular Velocity vector.
ψ	Yaw.
τ	Torque produced by thrust.
θ	Pitch.
φ	Roll.
ϑ	Virtual control input.
ξ	State vector.
$\mathbf{\Omega}$	Motors Rotational rate.
$\mathbf{\Phi}$	Rotation (Euler Angles) vector.

Roman symbols

C_D	Coefficient of drag.
C_L	Coefficient of lift.
C_M	Coefficient of moment.
F	Thrust force magnitude.
g	Gravitational acceleration.
m	Mass.

$\mathbf{A}$ Allocation Matrix.

$\mathbf{F}$ Thrust force vector.

$\mathbf{I}$ Inertia Tensor.

$\mathbf{J}$ Decoupling Matrix.

$\mathbf{u}$ Control inputs.

d Distance.

p, q, r Angular Velocity components.

v Longitudinal velocity.

v_x, v_y, v_z Velocity Cartesian components.

x, y, z Position Cartesian components.

$\mathbf{P}$ Position vector.

$\mathbf{V}$ Velocity vector.

Subscripts

0 Nominal value.

$axle$ Referring to the axle.

b Body-fixed frame.

d Desired value.

E External.

e Equilibrium.

f Flight mode.

g Ground mode.

i Inclined mode.

i, j, k Computational indexes.

r Rolling frame.

s Surface mode.

$wheel$ Referring to the wheel.

x, y, z Cartesian components.

Superscripts

’	Derivative.
-1	Inverse.
.	Derivative in time.
d	Derivative term.
I	Inertial frame.
i	Integral term.
p	Proportional term.
R	Rolling frame.
T	Transpose.

Glossary

2D	Two-Dimensional
3D	Three-Dimensional
DFL	Dynamic Feedback Linearisation
DOF	Degree-of-Freedom
IMU	Inertial Measurement Unit
ISR	Institute for Systems and Robotics
LIDAR	Light Detection and Ranging
MAV	Micro Aerial Vehicle
MR	Mixed Reality
NASA	National Aeronautics and Space Administration
O&M	Operation and Maintenance costs
PID	Proportional-Integral-Derivative
ROS	Robot Operating System
SISO	Single Input Single Output
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VR	Virtual Reality

Chapter 1

Introduction

This chapter will present an overview of the work developed in this thesis, by introducing the problem addressed and briefly explaining the proposed solution and the objectives of this endeavour. The main contributions and the structure of the thesis will also be given.

1.1 Motivation

The capability to function in a three-dimensional setting has greatly stimulated the research of Unmanned Aerial Vehicles (UAVs), for they can be used to perform tasks that other robots, which are restrained to the ground, are unable to. The latest progress has made them able to explore and map environments, manipulate objects or perform assembly tasks [1]. For this reason, multiple industries have been interested in adopting solutions based on aerial robots, tailoring them for different applications, such as autonomous inspection and surveillance.

Aerial robots can rapidly access target areas by flying over obstacles or cluttered terrain and reach regions that are inaccessible to humans or other robots. They can provide an elevated and bird's eye view sensing of the environment and approach hard-to-reach structures, while enhancing the efficiency and safety of inspection missions. Inspection of an industrial structure [2], solar panels and photo-voltaic installations [3], detecting heat leakages in buildings [4] and inspection of a vessel [5] are among many foreseen inspection applications for aerial robots.

Adapting aerial vehicles to perform inspection tasks can make maintenance and repair more cost-efficient, safer and faster [6]. In fact, equipping multi-rotors with the correct sensors, like 3D LIDAR devices, thermal or visual cameras, among others, allows autonomous navigation, the capability to detect defects (for example, surface cracks), environment mapping in locations inaccessible or dangerous for human operators, among many others [1]. In Section 2.1, some successful examples on surveillance using UAV's are analysed.

While the benefits of aerial robots for distant inspection of infrastructures have been extensively demonstrated [6], limited airborne solutions exist that can perform inspection in centimetre-range proximity to structures, or even while in physical contact with its surface. Remotely detected failures by

distant aerial robots need to be examined closely by the operators and often require additional efforts to detect the type and source of the failures. The state of the art airborne inspection solutions are subject to the strict limitations that are imposed by common aerial platforms, which are mainly unable to get close or even come in contact with structures without the risk of collision and even crashing [7]. Their flight time is limited due to the small battery size they can carry and their operation can easily be disrupted by disturbances such as wind [1]. Their inherent power and size constraints limit the dimensions, weight and capacity of the onboard sensors, which can have a negative impact in the performance of the inspection task.

However, if the robot is capable of landing and moving on top of the infrastructure, it will consume less energy and cover a larger area in a shorter period of time. By adding two wheels to an aerial robot and adjusting it to be able to interact and move on top of different surfaces or on the ground, the range of possible tasks is expanded: the robot can operate in multiple environments with increased stability and extended battery lifetime. This can ease operations like placing important sensors for inspection, which instead of being done in-air will be done by taking advantage of the new locomotion mode. It can also be used in cooperation with other robots, for example, collaborating with ground robots for power supply, or also using Mixed Reality (MR) and Virtual Reality (VR) for remote control by an operator, further enlarging its usability. Furthermore, this type of hybrid Aerial/Ground vehicles can be designed to take advantage of both modes, by moving on the ground between different assignments to save power and only flying when obstacles are encountered.

The broader goal of this work is to design a simple Bimodal aerial robot that can perform both macro and micro-level inspection of structures. This thesis describes the effort towards designing a robot which is capable of landing and moving in different surfaces, while employing the same set of actuators for all modes, and developing its fundamental control approach. Such robot is expected to enhance the efficiency of many aerial inspection missions as it can land or physically contact with the inspecting surface, to obtain a stable inspection and reduce the consumption of energy. This solution is of great interest for industrial inspection, and falls under a bigger research and development project: DURABLE (H2020 Interreg Atlantic area EAPA_986/2018) [8], a project that aims to promote the use of Renewable Energies in the Atlantic Region, by applying robotic technologies to solar and wind energy farms, which is expected to reduce Operation and Maintenance (O&M) costs. The hybrid robot can be used to perform non-destructive testings, thermography tests (that can detect defects) and it can integrate teams of robots that act together to repair problems in solar panels, wind turbines or any industrial structure, among others. This thesis work fits into the aforementioned research project, for which the Institute for Systems and Robotics (ISR) is contributing.

1.2 Objectives and Contributions

The aim of this thesis is to develop a proof-of-concept for a Bimodal aerial robot that is able to interact with different types of infrastructures, by flying, landing and moving on them autonomously. In order to demonstrate its feasibility, the focus will be on prototyping such robot, by mathematically describing

the new system and designing an autonomous controller, while ensuring the solution is extensible to different types of Bimodal robots. For the purpose of inspecting solar panels, wind turbines and other industrial structures, the vehicle should be able to fly or roll on the ground between them and dock on surfaces with different types of inclinations, as well as on flat or curved areas. This work targets the first three problems.

Figure 1.1: Prototype of the Bimodal aerial robot

The hybrid robot will consist of a multicopter UAV with two linked passive wheels that can rotate freely, which add little extra weight to the frame but cause the robot to rely entirely on its rotors for both flying and rolling, being necessary to develop a control system that allows point-to-point motion and trajectory tracking using only the UAV rotors. On the ground and on top of surfaces, the robot will behave similarly to a Unicycle, meaning the motion tasks are constrained by its nonholonomic kinematics, while on the air the vehicle behaves as a free-flying object and is only constrained by its underactuation. This indicates the dynamics of the robot will change accordingly, and different controllers will have to be designed for different scenarios.

To this end, a simulation will be conducted using the *RotorS* framework [9], which is a Gazebo package for the simulation of UAV's, providing some well-known multicopters and a structure that allows the testing of different controllers using the Robot Operating System (ROS) paradigm. The Gazebo physics engine and the Gazebo plugins will be responsible for simulating the robot and its sensors behaviour, while ROS interfaces the communication between the controller and the simulation, sending input commands to the robot and receiving its sensors data.

Taking advantage of the modular way in which *RotorS* was built, a generic approach was followed, granting the possibility to attach different wheels to any available UAV on the simulation and test its performance. A small prototype of the Bimodal robot will also be built, by attaching an axle with two wheels to an off-the-shelf quadrotor, like it can be observed in Figure 1.1, in order to assess the behaviour of the robot in simple experiments using teleoperation.

The main contributions of this work are summarised as follows:

- Development of a simulation framework for testing wheeled aerial robots for flying and moving in different surfaces. This simulated environment can be the base to also test inspection methods;
- Design of a linear controller for the Bimodal aerial robot, for its different modes: flight and flat/inclined surfaces;
- Design of a non-linear controller, for all modalities, to allow faster and more agile trajectories;
- Creation of a simple planner for hybrid missions, to allow transitioning between the various modes, approaching and landing in different surfaces;
- Prototyping the hybrid robot for future testing;
- Creation of a scalable solution that can be employed in UAVs of different shapes and sizes;

1.3 Thesis Outline

The contents of this thesis were structured as follows:

Chapter 2: In this chapter, some of the research works that cover the same topics as this thesis will be reviewed. Firstly, some examples of industrial uses for unmanned vehicles will be laid out, leading to the conclusion that Bimodal robots are not yet a mainstream option for the inspection of industrial infrastructures. Following this, an examination of the most relevant hybrid robots developed to date will be made, with emphasis on aerial robots adapted for terrestrial locomotion. The main approaches regarding the control of these vehicles will also undergo an analysis in this chapter.

Chapter 3: The third chapter will outline the mathematical model of a passive two-wheeled aerial robot, which is the proposed solution for the problem stated in this thesis. Three models will be derived since the dynamics of the robot will change in accordance with its operation mode.

Chapter 4: Two different control approaches were designed for the problems of trajectory tracking and waypoint following, both presented in this chapter. The first, a linear approach, is based on some simplifying assumptions that allows the use of linear tools for control design, while the second approach considered a non-linear controller for the stabilisation of the robots attitude, allowing for more aggressive manoeuvres.

Chapter 5: In order to verify the correct functioning of the proposed controllers, a simulation framework was developed using the ROS/Gazebo paradigm, that allows to verify the closed-loop behaviour of the Bimodal robot. This chapter delineates the simulation setup and presents a handful of experiments using both controllers in different scenarios.

Chapter 6: Finally, all the results obtained are recapped and summarised, to drive the conclusions of the research project. In addition, some suggestions of possible improvements are discussed in this chapter, for future reference.

Chapter 2

State of the Art

Robots can be tailored for a variety of uses, being possible to find a wide array of solutions regarding their use in industrial applications. However, to the best of our knowledge, the use of hybrid robots for such tasks hasn't been researched in depth. In this chapter, some examples are given regarding the use of aerial robots for industrial inspection, followed by the review and discussion of relevant works on the literature, that focus on the design of multi-modal robots. Finally, a brief revision on typical control solutions for both aerial and ground autonomous vehicles is addressed.

2.1 Inspection of Infrastructures using Aerial Robots

Like mentioned in Section 1.1, aerial robots have seen their use progressively increased for the surveillance and maintenance of infrastructures. UAVs present numerous advantages in performing these kind of tasks, since they can fly, unconstrained, towards infrastructures that are hard-to-reach by human operators, or are in potentially hazardous locations, and perform autonomous inspection of the area, through techniques that can involve contactless monitoring or non-destructive testings, implicating interaction with surfaces and objects. Some solutions using both methods are analysed below:

Contactless Applications: Examples where conventional UAVs and MAVs are used for contactless inspection include combining a lightweight MAV system with a vision-based state estimation autopilot, for inspection of an industrial boiler, using agile movements in this constrained environment [2], inspection of buildings [4], autonomously inspecting a vessel [5], or even acquiring sensor data to reconstruct the indoor walls of a chimney [10]. In fact, inspection of renewable energy plants using UAV's is a growing trend in the last few years, following the great incentives for energy transition. Some examples include using thermal and visual cameras to detect heat leakages [3], vision-based detection of defects [11] and cooperative inspection [12] of photovoltaic installations or wind turbines [13, 14].

Contact-Based Applications: On the other hand, many inspection tasks rely on real-time contact-based approaches, where the UAVs are equipped with an active and compliant manipulator system, that has to be prototyped to compensate for the multi-rotor dynamics, while staying airborne

[15]. These aerial manipulator systems can be used to place ohmmeters that identify defects on wind turbines or pressure sensors on dams [16, 17], to execute hammering tests on a bridge [18] or to scan pipes in industrial environments, like oil and gas plants [19]. Others, are able to stick to the wall and use ultrasonic probes in order to test the structure [20, 21] or even work together with other aerial robots for cooperative assembly [22].

These state of the art solutions, however, are subject to some limitations imposed by the flying platform itself - the operation might be disrupted due to disturbances, like wind, the battery might not be sufficient to carry out the mission to its full extent or it might simply be difficult to perform the task with a standard UAV. Conversely, some systems that provide extra versatility have been proposed, for instance, the "Bi^{2}Copter" [23], that takes advantage of propellers that tilt independently to be able to land, take-off and fly at any given angle and deal with the problem of getting a 360^o coverage of the environment, at the cost of increasing the complexity of the design.

The aim of this thesis work is precisely maximising the flexibility and stability of the robot, while minimising its complexity. Using a Bimodal aerial robot capable of landing and interacting with different types of surfaces and objects might be an alternative and effective way for contact-based inspection, providing advantages that the usual approaches don't have.

2.2 Multi-Modal Aerial Robots

An effective approach for increasing the versatility of an aerial robot, or any other robot in general, is to provide it with additional modes of locomotion, to obtain platforms that can adapt to different situations. In this way, the versatility that aerial robots natively have can be further increased by combining different types of mobility in the same robot. This has led to a growing interest in the research for hybrid robots, that are capable of aerial, terrestrial or even aquatic locomotion. In the last decade, some solutions have been presented in the literature, from multi-modal UAV's to wall-perching aerial robots, summarily reviewed in this section.

One of the most complete approaches can be found in [24], where an all-round two-wheeled quadrotor was designed to work in air, land and sea. It consists of a UAV with 2 rolling protective frames that also act as wheels, making it able to move in all directions on the ground, using only the thrust and rotational torques of the quadrotor. This robot is equipped with a feedback controller for the ground and flight mode and also includes an automatic battery charging device, which although not relevant for the aim of this work, is of great interest. However, the robot has a weight attached to it in order to keep the balance in the event of a poor landing, which increases the load of the overall system. Furthermore, in ground mode, the controller does not account for its nonholonomic constraint, which can lead to underperformance in several situations.

A similar work was employed in [25, 26], where a cylindrical cage was attached to a UAV through two revolute joints, to create a hybrid terrestrial and aerial robot, the "HyTAQ". These experiments validated the premise that a hybrid quadrotor increases its operation time, in view of the lower energy consumption of the ground mode when compared to the flight mode. The cylindrical cage that allows the

”HyTAQ” to roll on the floor presents the disadvantage of creating a resistance torque with the ground, that opposes the rolling and turning motion. To tackle this problem, in [27], two spokeless, independent wheels were developed to rotate around the cage, which facilitates turning and diminishes both ground and air resistance. Nevertheless, the design of this structure is much more complex and relies on the behaviour of the rollers between the cage and the wheel, which can deteriorate with time.

Likewise, in [28], a micro-quadcopter encapsulated in a spherical exoskeleton is proposed, to allow better performance on curved, rolling trajectories on the ground, by rotating about its central axis and applying a thrust force parallel to the ground. Another solution, the ”MUWA”, consists of a quadrotor with an annular disc around it, that works like a mono-wheel, floats and moves on water and can even stand on the ground at a given tilt-angle, besides operating as a conventional UAV [29]. These experiments presented remarkable results, however they fit a different purpose, namely to move in tight and constrained spaces like disaster sites, where robots are required to go through narrow gaps.

In [30] a spherical shell was attached to a UAV through a gimbal mechanism with 3 Degrees-Of-Freedom (DOF), to allow inspection in confined spaces, but not ground locomotion. Notwithstanding, the issue of landing and taking-off from inclined surfaces is discussed. On the contrary, in [31, 32] a hybrid aerial/terrestrial robot is achieved by equipping a ground robot with four rotors, a solution expensive in hardware, that increases the mass and lowers the energy efficiency, built mostly to avoid in-land obstacles. Examples of other distinctive concepts include a two-wheeled ground robot with a helicopter mechanism folded into its own body [33–35], a flying robot that shares its structure for different modalities, using its wings for both flying and walking [36] or the ”DUCK”, a quadrotor combined with passive legs that uses its thrust for dynamic walking [37]. The design employed by [38] is also singular, for using skateboard steering truck wheels below 2 of the rotors for turning, being successful in providing a rolling mechanism in semi-smooth surfaces without altering the avionics of the original quadrotor.

Recent years have also seen the development of a range of strategies to adapt UAV’s for moving in vertical surfaces, from flying robots that can perch onto them using microspine technology [39], suction cups [40] or dry-adhesive grippers [41], quadcopters that have tilt-mechanisms that allow them to shift the direction of their thrusters and attach to walls through a normal force [42], to robots that can fly and perch using a compliant, underactuated gripping mechanism [43]. While successful, these designs implicate additional actuators and mechanisms, increasing weight and complexity, being advantageous to carry out inspection in vertical surfaces or ceilings, but not on surfaces with different types of inclinations like solar panels or wind-turbines.

Finally, it is worth mentioning the efforts made by Team CoSTAR from NASA Jet Propulsion Laboratory [44] with its *Nebula* autonomy solution, which addresses autonomous exploration of extreme environments (for planetary exploration). While this robotic ecosystem is highly complex, it comprises drones with hybrid rolling/flying mechanisms, such as the ”Drivocopter” [45], which is a quadcopter UAV with four spherical shells surrounding the propellers, that act as independent actuated wheels. However, for the development of this thesis, their most interesting design is presented in [46], where a control scheme was developed for a passive two-wheeled hybrid UAV. This work exploited the similarity between differential flatness mappings of quadrotors and nonholonomic vehicles, in order to design a local

planner that generates feasible trajectories for both methods of mobility using the same representation. Nonetheless, the mathematical model of the robot is not given, nor the behaviour of the transition from one mode to the other.

2.3 Control of Autonomous Vehicles

The proposed robot will act like a UAV when operating in the air but will behave like an Unmanned Ground Vehicle (UGV) when operating on a surface, meaning both control problems have to be considered. Since aerial and ground robots present non-linear dynamic models, most approaches are either based on a linearisation of the dynamics around some working condition or exploit these nonlinearities for the design of a more robust controller, which are summarily reviewed in this section.

2.3.1 Aerial Robots Control

A **linear procedure** has the advantage of being more straightforward, requiring less computational power, but is usually constrained to work around a certain point, which in the case of a UAV is the hovering condition [47], when the thrust force produced by the propellers equals the weight of the robot. About this equilibrium, a simplified model can be derived that allows the use of linear control methods.

Most linear approaches use a cascaded architecture, in which the attitude is regulated with an inner-loop that runs at at a faster rate, while the outer-loop is responsible for computing the necessary attitude angles to reach a certain position, achieved by a series of Proportional-Integral-Derivative (PID) control laws [48, 49]. Mellinger et al. [50] proposes a similar approach for small angle control, but inverts the relationship between the Euler angles and the desired accelerations to feed the attitude controller.

Other examples of linear methods can be found, for instance, in [51], where a Linear Quadratic Regulator (LQR) is used to determine the control signal that optimise a given cost function, or in [52] where two other techniques based on optimal control theory are used, granting robustness to external disturbances. Another good example is reported in [53] where extended PID is used together with feedforward linearisation using the flatness properties of an Helicopter UAV.

On the other hand, **non-linear** control of aerial robots has also been addressed in various publications. The work in [54] takes advantage of the fact that the dynamics can de divided into three subsystems - a longitudinal, a lateral and a vertical one - and derives a control law for each using integral backstepping. Similarly, in [55], stabilisation is achieved using a full-state backstepping approach, that derives a global Lyapunov function for the three sub-systems, guaranteeing asymptotic stability.

Another technique consists of using feedback linearisation, where the outputs to be controlled are successively differentiated until the inputs appear in a non-singular way. The resulting relationship is inverted to transform the system into a linear form that can easily be controlled through a polynomial law. One can consider the position and the yaw angle as the outputs to be directly controlled and achieve full-state feedback through a dynamic extension [56, 57], or feedback linearise the rotational dynamics while stabilising the translational dynamics with linear methods [58]. The work employed by Sabatino et

al. [59] compares these two approaches.

However, these non-linear controllers are based on Euler angles and can exhibit singularities when performing complex rotational movements. To solve this issue Lee et al. [60, 61] proposed a very acclaimed solutions: a geometric tracking controller that presents almost global exponentially attractiveness, thus allowing for sophisticated manoeuvres.

2.3.2 Ground Robots Control

When the two-wheeled UAV operates on the ground, the robot can only move along its longitudinal direction while simultaneously rotating, being kinematically equivalent to a unicycle since it cannot move sideways. This is expressed by the pure rolling condition of the wheels - the velocity of its contact point with the surface is zero in the lateral direction - which is a nonholonomic constraint [62]. Therefore, when designing a control scheme for the terrestrial locomotion of the Bimodal robot, this constraint has to be taken into account, similarly to what is done for most ground robots.

Many unicycle tracking controllers consider the rotated version of the position error, which is the cartesian error expressed in a reference frame aligned with the current orientation of the robot, in order to design a feedback loop that stabilises this error dynamics [62]. The work employed in [63] synthesises the main approaches to tackle the problem of trajectory tracking. The simplest solution comprises the combination of a nominal feedforward command with feedback action, by either linearising the dynamics along the reference trajectory or by defining a control law via the use of a Lyapunov function. A third non-linear controller is developed in this work, by means of a dynamic feedback linearisation that achieves good asymptotic convergence but has a singularity when the driving velocity is zero, meaning the robot cannot stop rolling.

In [64] an identical approach was taken using the differential flatness properties of the unicycle, developing both a kinematic and a dynamic controller using this method. Another systematic procedure is to design an input/output linearising controller, by choosing a different output to be controlled, like reported in [62].

On the other hand, for the regulation problem, some examples of well known solutions are described in detail by Dimarogonas et al. [65]. This problem is typically divided into posture or cartesian regulation, with different solutions passing from controllers based on polar coordinates to using a series of time-varying feedback laws.

Notwithstanding, the solutions reviewed above consider that wheeled robots are controlled via their kinematics inputs, that is, the driving and steering velocities. For the Bimodal robot developed in this research project, however, that is not the case. The torque generated by the propellers will create a rotational movement, while the generated thrust force will be responsible for the forwards/backwards movement of the vehicle, as described in the next chapter.

Chapter 3

Design and Mathematical model of the Robot

This chapter follows the formulation of the mathematical description of the robot, adopting the Newton-Euler formalism that dictates the translational and rotational dynamics for a rigid body. Since the moments and forces acting on the vehicle will be different depending on where it is operating, three different models will have to be derived: one for aerial operation, another for ground locomotion and a third for inclined surfaces. This will prove essential in order to derive a good tracking controller for the system.

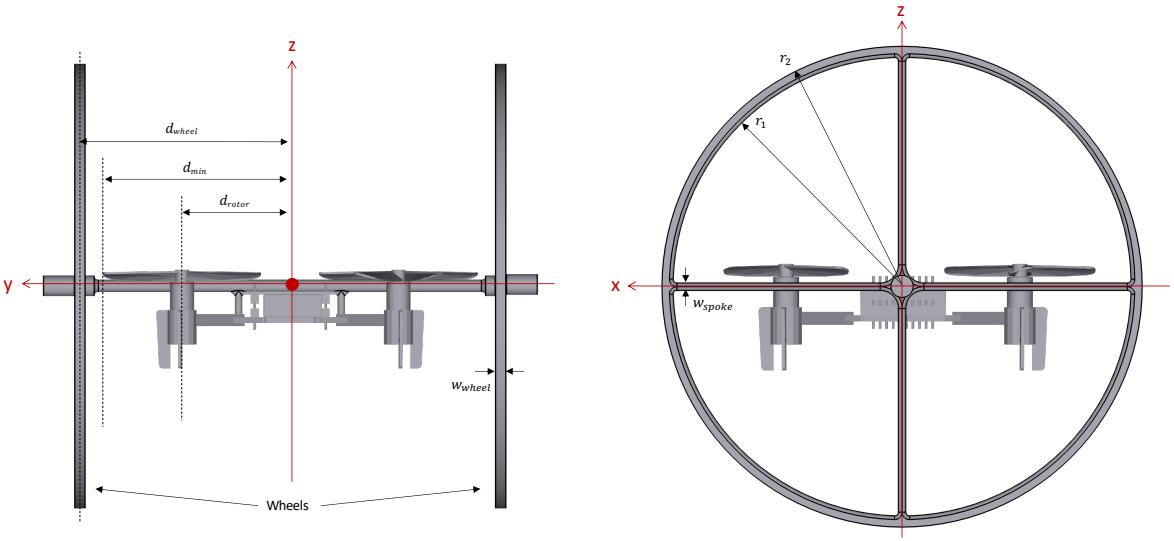
3.1 Mechanical Considerations

The Bimodal robot is composed of a common X-configuration quadrotor attached to two passive all-round wheels through an axle, that can rotate freely and independently around its axis. This simple design allows utilising the same set of actuators for controlling the robot motion in all modes of locomotion. On top of the added mobility, the wheels offer all-round protection for the platform, ensuring that the body and propellers are well protected against impacts and collisions. An example of a concept for a Bimodal aerial robot is depicted in Figure 3.1. We refer the reader to Appendix A, where the prototype developed in this work is shown in detail.

Some considerations have to be made regarding the mechanical specifications of the platform. The designed wheels have to be lightweight and should take into account the payload supported by the rotors. Furthermore, the material of the wheels must ensure enough friction exists between them and the surface, otherwise the wheels might slide instead of rolling. Since the air resistance force is proportional to the cross-sectional contact area of the object [27], the rim and spokes width should be minimised as much as possible.

If the quadrotor is placed below the axle, when the rotors stop the system will act like a pendulum, returning to the equilibrium position with the rotors facing up. However, for the sake of simplicity, the centre of mass of the axle+wheels system was considered to be coincident with the centre of mass of the quadrotor. Another important regard consists of the distance between the wheels and the propellers,

which need to be well protected when functioning and should not collide with the spokes, $d_{wheel} > d_{min}$. To offer all-round protection for the propellers, which should not get damaged even if the robot is perpendicular to the surface where it is moving, the wheel outer radius should be bigger than the length of the quadrotor, i.e, $r_2 > 2d_{min}$.



(a) Front view. (b) Side view.

Figure 3.1: Bimodal aerial robot concept.

However, the distance between the wheels and the centre of mass of the robot should also be reduced, given the inertia increases the further the wheels are. To determine the added inertia, we need to approximate the wheels geometry. If the spokes are thin, the wheel can be considered a hollow cylinder, while the axle can be considered a thin rod. Using the formulation available in Appendix B and bringing the inertias of the wheels to the centre of mass of the robot, using the parallel axis theorem:

$$I_x^{wheel} = I_z^{wheel} = \frac{1}{12} m_{wheel}[3(r_1^2 + r_2^2) + w_{wheel}^2] + m_{wheel} d_{wheel}^2, \tag{3.1}$$

$$I_y^{wheel} = \frac{1}{2} m_{wheel}(r_1^2 + r_2^2), \tag{3.2}$$

$$I_x^{axle} = I_z^{axle} = \frac{1}{12} m_{axle}(2d_{wheel})^2. \tag{3.3}$$

Considering the wheels are placed symmetrically, the products of inertia can be neglected. The robot's final mass and inertia properties will be found by summing the contribution of the wheels+axle system with the quadrotor, without considering the extra inertia on the Y-axis (since the robot can rotate around Y-axis independently of the wheels). Putting everything together:

$$m = m_{quad} + 2m_{wheel} + m_{axle}, \tag{3.4}$$

$$I_x = I_x^{quad} + 2I_x^{wheel} + I_x^{axle}, \tag{3.5}$$

$$I_y = I_y^{quad}, \tag{3.6}$$

$$I_z = I_z^{quad} + 2I_z^{wheel} + I_z^{axle}. \tag{3.7}$$

3.2 Notation

Firstly, it is necessary to define the inertial and robot coordinate frames, which followed the typical UAV conventions, with a positive altitude upwards specifying the Inertial frame. The body-fixed reference origin is the centre of mass of the robot and the x_b-y_b frame is oriented following an X-configuration [48], which simplifies position tracking for ground mode. Moreover, like shown in Figure 3.2, the conventional notations for an aerial robot are used:

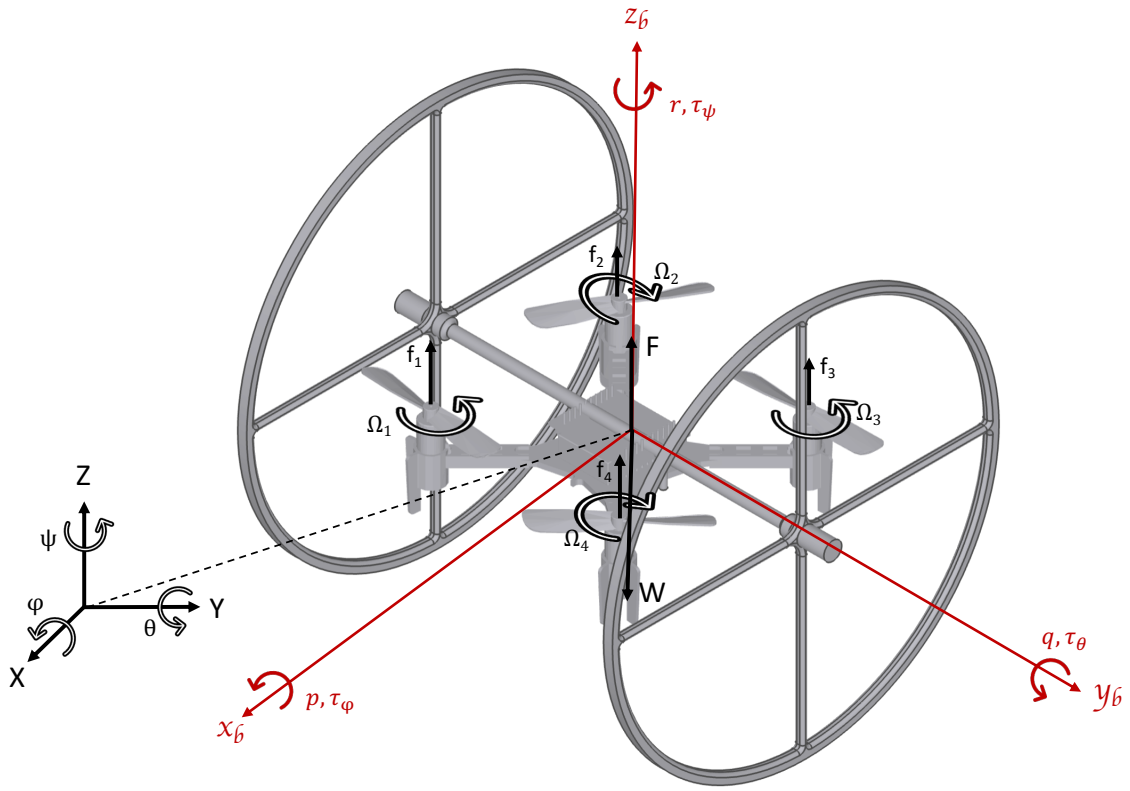


Figure 3.2: Model of free-flying Bimodal aerial robot.

Inertial Frame :	$SR_I = \{X, Y, Z\}$
Body Fixed Frame :	$SR_B = \{x_b, y_b, z_b\}$
Position of SR_B w.r.t SR_I :	$\mathbf{P} = (x, y, z)$
Linear Velocity of SR_B w.r.t SR_I :	$\mathbf{V} = (v_x, v_y, v_z)$
Rotation of SR_B w.r.t SR_I (Euler Angles) :	$\mathbf{\Phi} = (\varphi, \theta, \psi)$
Angular Velocities of SR_B :	$\omega = (p, q, r)$
Torques generated by thrust :	$\tau = (\tau_\varphi, \tau_\theta, \tau_\psi)$
Rotational velocity of i^{th} rotor :	Ω_i
Thrust produced by i^{th} rotor :	f_i

Collective thrust vector in $\mathrm{SR_B}$:	$\mathbf{F} = (0, 0, F)$
Mass of robot :	m
Gravitational acceleration :	g
Weight vector in $\mathrm{SR_I}$:	$\mathbf{W} = (0, 0, -mg)$
Inertia Tensor :	$\mathbf{I} = diag(I_x, I_y, I_z)$

3.3 Motor Dynamics

In order to model the robot with the rigid body equations of motion, it is first required to determine the map between the actual actuators - the motors - and the forces/torques acting on the centre of mass, i.e, how the different rotor speeds will affect the forces and moments acting on the system. Like seen in Figure 3.3, the first and third rotors rotate anti-clockwise while the second and fourth rotate clockwise, with the arm length, d, being the distance from the centre of each propeller to the centre of mass of the robot.

Empirically, one can understand that if a uniform increase or decrease happens on the propeller speeds, the altitude will increase or decrease, respectively, while a change in the pair (Ω_1, Ω_2) or (Ω_3, Ω_4) causes the airframe to tilt around x_b-axis, altering the roll angle (φ). On the other hand, varying the pair (Ω_1, Ω_4) or (Ω_2, Ω_3) will make the aircraft pitch (θ) around the y_b-axis, while the pair (Ω_1, Ω_3) or (Ω_2, Ω_4) cause a rotation around z_b-axis, changing the yaw angle (ψ).

The aerodynamics of rotors have already been largely studied [66]. A simplified model that considers the thrust generated by each propeller is proportional to the square of its angular rotation,

$$f_i = C_T \Omega_i^2, \tag{3.8}$$

is adopted, where $C_T > 0$ is the rotor thrust coefficient. The reaction torque generated by each motor is also a function of its squared angular rotation,

$$\tau_i = C_M \Omega_i^2, \tag{3.9}$$

with C_M being the rotor moment constant. The total thrust generated by the airframe can hence be formulated as

$$F = \sum_{i=1}^{4} f_i = C_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \tag{3.10}$$

while the torques are expressed as:

$$\tau_\varphi = \frac{d}{\sqrt{2}} C_T(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \tag{3.11}$$

$$\tau_\theta = \frac{d}{\sqrt{2}} C_T(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2), \tag{3.12}$$

$$\tau_\psi = C_M(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2). \tag{3.13}$$

In matrix form:

$$\mathbf{U} = \mathbf{A}.\mathbf{\Omega^2} \Leftrightarrow \begin{bmatrix} F \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ -\frac{d}{\sqrt{2}}C_T & -\frac{d}{\sqrt{2}}C_T & \frac{d}{\sqrt{2}}C_T & \frac{d}{\sqrt{2}}C_T \\ -\frac{d}{\sqrt{2}}C_T & \frac{d}{\sqrt{2}}C_T & \frac{d}{\sqrt{2}}C_T & -\frac{d}{\sqrt{2}}C_T \\ -C_M & C_M & -C_M & C_M \end{bmatrix} \cdot \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{3.14}$$

with the Allocation Matrix, $\mathbf{A}$, being invertible for all $d, C_M \neq 0$. This means the control inputs can be considered to be $\mathbf{U} = [F, \tau_\varphi, \tau_\theta, \tau_\psi]^T$, since we can use the inverse of the Allocation matrix to solve for the rotor speeds.

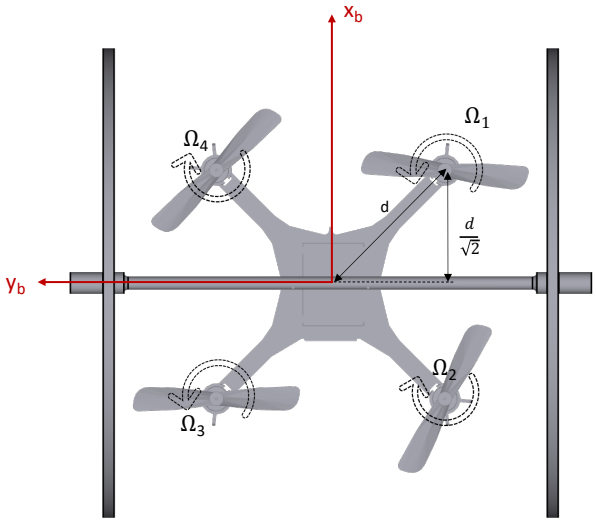


Figure 3.3: Quadrotor motors in X-configuration.

3.4 Dynamic model of the system

The dynamic model for the system will be derived using the Newton-Euler equations for a rigid body. These well known equations state that the sum of the forces acting on the body can be expressed as

$$\sum \mathbf{F_E} = m\dot{\mathbf{V}}, \tag{3.15}$$

whereas the sum of the angular moments is given by

$$\sum \mathbf{M_E} = \mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega. \tag{3.16}$$

3.4.1 Flight Model

When the robot is operating in the air, its dynamics will follow the same model as that of a conventional quadcopter, with increased inertia values due to the addition of the wheels. The problem of formulating the mathematical equations for a quadcopter has already been extensively studied in the literature, some great examples being [47], [59] or [66], which will be closely followed.

Let's start by defining the rotation matrix that relates the orientation of the body fixed frame w.r.t the inertial frame, following a RPY (Roll-Pitch-Yaw) rotation sequence, obtaining

$$\mathbf{R}_{\mathbf{B}}^{\mathbf{I}} = \begin{bmatrix} c_\theta c_\psi & s_\varphi s_\theta c_\psi - c_\varphi s_\psi & c_\varphi s_\theta c_\psi + s_\varphi s_\psi \\ c_\theta s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi \\ -s_\theta & s_\varphi c_\theta & c_\varphi c_\theta \end{bmatrix}, \tag{3.17}$$

where $c_\bullet$ and $s_\bullet$ represent, respectively, $cos(\bullet)$ and $sin(\bullet)$. From equations (3.15) and (3.16), the rigid body equations of motion can be expressed as:

$$\dot{\mathbf{P}} = \mathbf{V}, \tag{3.18}$$

$$m\dot{\mathbf{V}} = \mathbf{W} + \mathbf{R}_{\mathbf{B}}^{\mathbf{I}}\mathbf{F}, \tag{3.19}$$

$$\omega = \mathbf{R}_\omega \dot{\mathbf{\Phi}}, \tag{3.20}$$

$$\tau = \mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega, \tag{3.21}$$

where the relationship between the Euler angles derivatives and the rotational rates of the body-frame is given by

$$\mathbf{R}_\omega = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\varphi & s_\varphi c_\theta \\ 0 & -s_\varphi & c_\varphi c_\theta \end{bmatrix}. \tag{3.22}$$

Solving (3.18)-(3.21) for the **state** of the robot,

$$\xi_f = (x, y, z, v_x, v_y, v_z, \varphi, \theta, \psi, p, q, r)^T,$$

we arrive to the **mathematical model** of the system in **flight mode** as described by (3.23), where $t_\bullet$ stands for $tan(\bullet)$. This model presents a singularity when $\theta = \pi/2$ for using the Euler Angle parametrisation, which can be avoided if a quaternion notation is used instead. For a step-by-step solution, which was omitted for being broadly present in the literature, we refer the reader to [48].

$$\dot{x} = v_x \tag{3.23a}$$

$$\dot{y} = v_y \tag{3.23b}$$

$$\dot{z} = v_z \tag{3.23c}$$

$$\dot{v}_x = (c_\varphi s_\theta c_\psi + s_\varphi s_\psi)\frac{F}{m} \tag{3.23d}$$

$$\dot{v}_y = (c_\varphi s_\theta s_\psi - s_\varphi c_\psi)\frac{F}{m} \tag{3.23e}$$

$$\dot{v}_z = c_\varphi c_\theta \frac{F}{m} - g \tag{3.23f}$$

$$\dot{\varphi} = p + s_\varphi t_\theta q + c_\varphi t_\theta r \tag{3.23g}$$

$$\dot{\theta} = c_\varphi q - s_\varphi r \tag{3.23h}$$

$$\dot{\psi} = s_\varphi sec(\theta) q + c_\varphi sec(\theta) r \tag{3.23i}$$

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{\tau_\varphi}{I_x} \tag{3.23j}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_\theta}{I_y} \tag{3.23k}$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_\psi}{I_z} \tag{3.23l}$$

3.4.2 Ground Model

In the ground or on top of a flat surface, the movement of the robot will be similar to that of a Unicycle, however, it will still rely on the thrust generated by the propellers of the quadrotor in order to move, since the two added wheels are passive, meaning the model has now to incorporate different constraints. The following were the primary assumptions made:

1. The 2 wheels are always on the ground (robot is moving on the X-Y plane), which means rotation around the X-axis should be kept to zero, not allowing roll movement, i.e, $\varphi(t) = 0$.
2. The altitude of the robot is kept constant at surface level, since it must always be in contact with the plane, meaning no translation on the Z-axis can occur, i.e, $z = const \rightarrow \dot{z} = 0 = \dot{v_z}$.
3. Furthermore, it is assumed there is always friction between the wheels and the surface, therefore the wheels roll without slipping (implying there cannot be movement along the y_b-axis).

For this to be true, the following constraints have to be satisfied:

1. **No-roll condition:** The roll rate should be maintained at zero,

$$\dot{\varphi} = 0. \tag{3.24}$$

2. **No-takeoff condition:** The vertical thrust component has to be smaller or equal to the weight of the robot, while the horizontal component will be responsible for moving the robot,

$$|Fcos(\theta)| \leq mg. \tag{3.25}$$

3. **Nonholonomic constraint:** The robot's velocities have to respect the nonholonomic kinematics imposed by the wheels,

$$\dot{x}sin(\psi) - \dot{y}cos(\psi) = 0. \tag{3.26}$$

Figure 3.4 shows a schematic of the robot moving on a surface, where a new auxiliary reference frame was defined, the Rolling frame ($\mathbf{SR_R}$), attached to the robots centre of mass but with the $x_r - y_r$ plane always parallel to the surface where the robot is moving, similarly to what is done in [46].

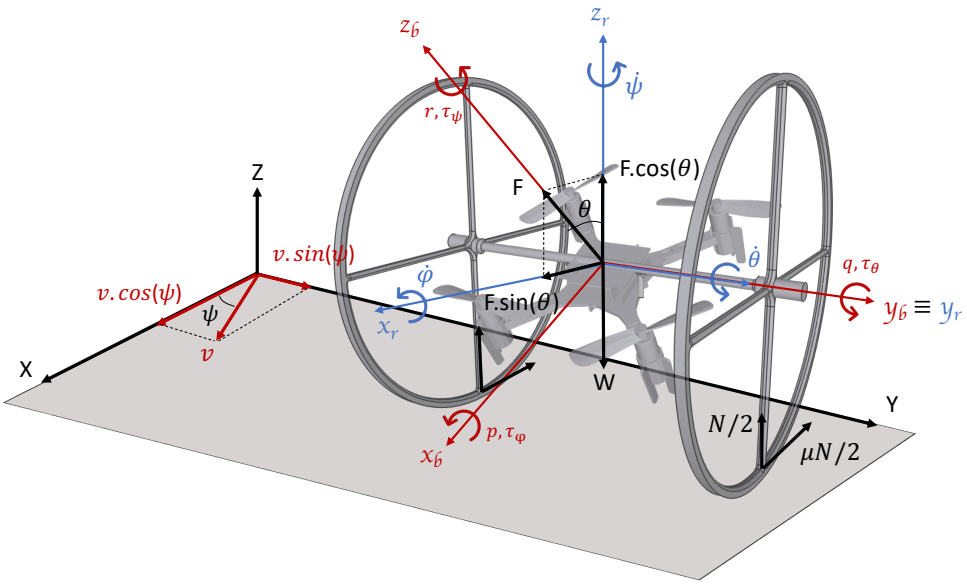


Figure 3.4: Model of Bimodal robot on a planar surface.

This auxiliary frame is very helpful in accurately describing the movement of the hybrid robot on top of a surface. To generate motion, the vehicle needs to pitch around the y_r-axis and have a thrust force satisfying constraint (3.25) - the horizontal component of the thrust will move the system, making the wheels roll without sliding, due to the action of static friction. This accelerated movement makes the robot move on the plane with velocity v, that can only have a component along the x_r-direction, respecting the nonholonomy of the robot, like illustrated by Figure 3.5(a).

On the other hand, the torques produced by the rotors will be responsible for the turning motion around the z_r-direction while guaranteeing there is no rotation around the x_r-direction, like condition (3.24) dictates. Figure 3.5(b) shows the forces acting on the robot when it is moving. The reaction force, N, and the friction force, μN, are assumed to be equally distributed between the two wheels.

In this case, a more straightforward approach is to apply equation (3.15) on the forces acting on the Rolling frame,

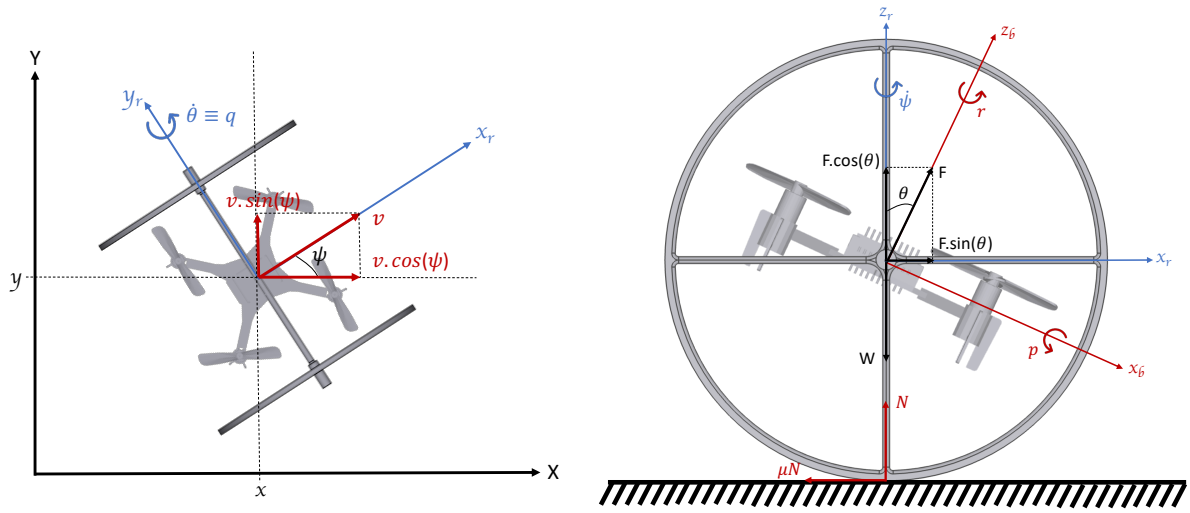
$$\begin{bmatrix} F\ s_\theta \\ 0 \\ F\ c_\theta \end{bmatrix} + \begin{bmatrix} -\mu N \\ 0 \\ N \end{bmatrix} + \mathbf{W} = m \begin{bmatrix} \dot{v} \\ 0 \\ 0 \end{bmatrix}, \tag{3.27}$$

to obtain the expression for the normal force,

$$N = mg - F\ c_\theta, \tag{3.28}$$

and the acceleration of the robot in the x_r-direction,

$$\dot{v} = \frac{F}{m} s_\theta - \mu \frac{N}{m}, \tag{3.29}$$



(a) Top view.

(b) Side view.

Figure 3.5: Bimodal robot moving on the plane.

where μ is the coefficient of friction (between the wheel and the surface).

Unlike the flight mode, where the vehicle acts as a free-flying robot, the kinematics of the ground mode have to consider the nonholonomic constraint, like already discussed and illustrated by Figure 3.5(a). Since the Z-direction is not considered, the **translational kinematics** can be formulated as that of a unicycle,

$$\dot{x} = v\ c_\psi, \tag{3.30a}$$

$$\dot{y} = v\ s_\psi. \tag{3.30b}$$

Next, considering that the roll angle is always zero ($c_\varphi = 1$, $s_\varphi = 0$), the matrix that relates the Euler angle rates with the angular velocity components given by (3.22) can be reduced, in this case, to

$$\mathbf{R}_{\omega\mathbf{g}} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & 1 & 0 \\ 0 & 0 & c_\theta \end{bmatrix}, \tag{3.31}$$

meaning the angular velocity (in SR_B) can be expressed by

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{R}_{\omega\mathbf{g}}. \begin{bmatrix} 0 \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\dot{\psi} s_\theta \\ \dot{\theta} \\ \dot{\psi} c_\theta \end{bmatrix}. \tag{3.32}$$

As one can see, due to the coupling of the pitch+yaw movements, there is a rotation around x_r-axis that needs an offset to satisfy the condition of keeping the wheels in contact with the surface. This makes sense, given that in the extreme case of the robot pitching $\theta = \frac{\pi}{2}$, the rotational rate around x_b-axis will

be responsible for the turning motion of the robot,

$$p = -\dot{\psi} s_\theta = -\dot{\psi}. \tag{3.33}$$

In all other cases, this rate makes sure that $\dot{\varphi}$ always remains zero and consequently there is no change in the roll angle φ. Hence, there is still the need to control the moments around the three axis, whose expressions are given by (3.16) and remain unchanged for the ground mode. The **rotational dynamics** are consequently given by:

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{\tau_\varphi}{I_x}, \tag{3.34a}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_\theta}{I_y}, \tag{3.34b}$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_\psi}{I_z}. \tag{3.34c}$$

Putting everything together, the **state** of the robot in **ground mode** can be fully described by its 2D position, the velocity along the rolling direction, the pitch and yaw angle (since the roll is always zero) and the three angular velocity components:

$$\xi_g = (x, y, v, \theta, \psi, p, q, r)^T.$$

Considering the same control inputs as before, the **mathematical model** of the system in **ground mode** can be derived as in (3.35).

$$\dot{x} = v\ c_\psi \tag{3.35a}$$

$$\dot{y} = v\ s_\psi \tag{3.35b}$$

$$\dot{v} = s_\theta \frac{F}{m} - \mu \frac{N}{m} \tag{3.35c}$$

$$\dot{\theta} = q \tag{3.35d}$$

$$\dot{\psi} = r\ sec(\theta) \tag{3.35e}$$

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{\tau_\varphi}{I_x} \tag{3.35f}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_\theta}{I_y} \tag{3.35g}$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_\psi}{I_z} \tag{3.35h}$$

3.4.3 Inclined Surface Model

Although the previously defined robot model can be a good approximation for surfaces with slight inclinations, it will not be a suitable choice for larger inclinations, such as when moving on the surface of

structures. For the purpose of inspecting outdoor infrastructures like solar panels, the robot will have to move up and down slopes with considerable inclinations, meaning a weight component will appear, changing once again the dynamics of the vehicle, like expressed in Figure 3.6.

Considering the general case of the robot moving on an plane with an inclination angle of $\gamma \in]0, \frac{\pi}{2}[$, where the limit cases of $\gamma = 0$ and $\gamma = \pi/2$ reduce to the ground and flight mode, respectively, the previous conditions will not hold unless new assumptions are made:

1. The inclination angle γ and the orientation of the slope w.r.t SR_I, ψ_0, are known *a priori*.

2. The robot will only move up and down the slope without steering, i.e, $\psi(t) = \psi_0$.

The second condition is to ensure a safe navigation along any type of slope, independently of its inclination and friction properties. This is because the design of the robot only allows the thrust force to point in the direction of z_b, meaning if the vehicle steers to the left or right, there will be no thrust component to move the system against the gravity force (when $\psi - \psi_0 = \pi/2$).

For this reason, the **no-roll** and **nonholonomic constraints**, represented by conditions (3.24) and (3.26), will remain unchanged, so that the wheels do not come off the surface. To satisfy the new constraints, the no-takeoff condition has to be reformulated and a new requirement has to be imposed, as follows:

1. **No-takeoff condition:** The thrust component must be inferior to the weight component in the direction orthogonal to the surface,

$$F\ cos(\theta + \gamma) < mg\ cos(\gamma), \tag{3.36}$$

2. **No-turning motion:** The yaw rate should be maintained at zero,

$$\dot{\psi}(t) = 0. \tag{3.37}$$

Figure 3.6 shows a schematic of this case, where the vehicle is moving up a generic slope. Again, it's easier to represent the movement of the robot by recurring to the auxiliary Rolling frame, with the x_r-direction being the only direction in which the robot can move, similarly to the flat plane case. From the balance of forces acting on the Rolling frame,

$$\begin{bmatrix} F\ s_{(\theta+\gamma)} \\ 0 \\ F\ c_{(\theta+\gamma)} \end{bmatrix} + \begin{bmatrix} -\mu N \\ 0 \\ N \end{bmatrix} + \begin{bmatrix} mg\ s_\gamma \\ 0 \\ mg\ c_\gamma \end{bmatrix} = m \begin{bmatrix} \dot{v} \\ 0 \\ 0 \end{bmatrix}, \tag{3.38}$$

it is possible to find the Normal force expression like before,

$$N = mg\ c_\gamma - F\ c_{(\theta+\gamma)} \tag{3.39}$$

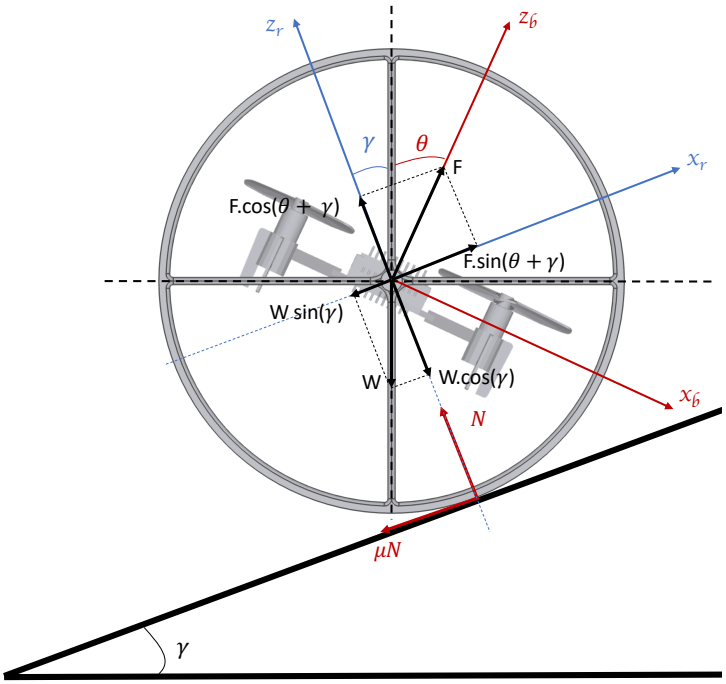


Figure 3.6: Model of the Bimodal robot going up a slope.

and the expression for the acceleration of the robot,

$$\dot{v} = \frac{F}{m} s_{(\theta+\gamma)} - g.s_\gamma - \mu \frac{N}{m}. \tag{3.40}$$

The **translational kinematics** when the robot is operating on top of an incline will be similar to before, but with a component on the Z-axis, respecting the following expression:

$$\dot{x} = v\ c_\gamma c_\psi, \tag{3.41a}$$

$$\dot{y} = v\ c_\gamma s_\psi, \tag{3.41b}$$

$$\dot{z} = v s_\gamma. \tag{3.41c}$$

The rotational movement of the robot when moving along the slope respects the same dynamics as in the previous case, given by (3.32) and (3.34). Since both the roll and yaw are constrained the **state** of the robot in **inclined mode** can be fully characterised by

$$\xi_i = (x, y, z, v, \theta, q)^T, \quad \zeta = (\gamma, \psi_0),$$

where ζ are the parameters that describe the slope. Putting everything together, the **mathematical model** for the **inclined mode** can be formulated as in (3.42).

$$\dot{x} = v\ c_\gamma c_{\psi_0}, \tag{3.42a}$$

$$\dot{y} = v\ c_\gamma s_{\psi_0}, \tag{3.42b}$$

$$\dot{z} = v\ s_\gamma, \tag{3.42c}$$

$$\dot{v} = \frac{F}{m} s_{(\theta+\gamma)} - g\ s_\gamma - \mu \frac{N}{m} \tag{3.42d}$$

$$\dot{\theta} = q, \tag{3.42e}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_\theta}{I_y}. \tag{3.42f}$$

Chapter 4

Control Strategies

In this chapter, two different control approaches will be derived for the hybrid robot, tackling both the problems of trajectory tracking and waypoint-following, for all the different modes of locomotion.

4.1 Cascaded PID Controller

Like already discussed in Section 3.3, the thrust and the three torques acting on the hybrid robot can be controlled by action of the four rotor inputs that change the attitude of the vehicle. Hence, it is not possible to control the position of the vehicle by means of a direct linear approach. Regardless, the control problem can be divided into attitude stabilisation and position tracking, allowing the use of linear techniques, in a nested structure: the first will be responsible for regulating the attitude (φ, θ, ψ) of the robot, while the second calculates the necessary angles at each moment to reach the desired position.

Figure 4.1 shows a block scheme of the closed-loop system in flight mode, where the **Position Controller** receives the user-defined point or trajectory and estimates of the current state, computing the necessary thrust and attitude to reach the reference command, through error feedback. The next block, **Attitude Controller**, determines the necessary torques to bring the robot from the current orientation to the desired one, received from the outer control loop. Finally, the **Motor Allocation** is responsible for allocating the rotors, by taking the control commands and regulating the rotational speeds of the motors ($\mathbf{\Omega}$), computed using the inverse of the Allocation matrix. Further details of this step will be given in Chapter 5.

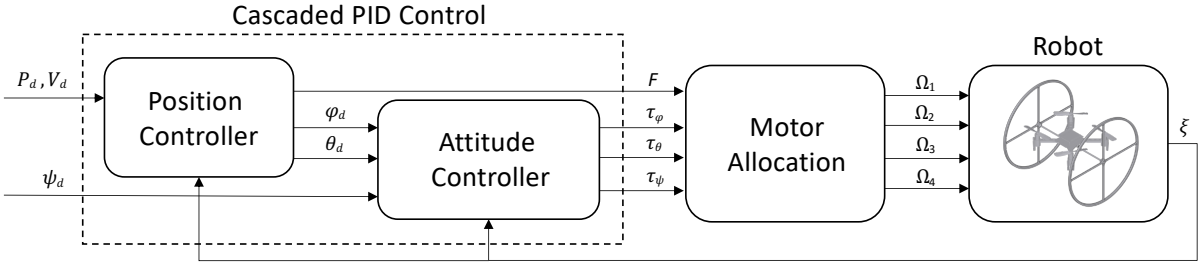


Figure 4.1: PID Flight control of Bimodal aerial robot

4.1.1 PID Flight Mode

The cascaded controller design relies on linearising the systems dynamics around an equilibrium point in which the state is static, meaning $\dot{\xi}_e = 0$. Like mentioned before, in flight mode, this point corresponds with the hovering condition, in which the robots thrust is equal to its own weight, keeping the position and altitude stationary and any linear or angular velocities at zero, i.e,

$$\xi_{f_e} = (x_e, y_e, z_e, 0, 0, 0, 0, 0, \psi_e, 0, 0, 0).$$

To maintain the vehicle stationary in the air, all motors need to rotate with the same speed,

$$\Omega_{1e} = \Omega_{2e} = \Omega_{3e} = \Omega_{4e} = \Omega_e, \tag{4.1}$$

which means, from equation (3.10), one can find the expression for the equilibrium rotational velocities of the rotors,

$$C_T(\Omega_{1e} + \Omega_{2e} + \Omega_{3e} + \Omega_{4e}) = mg \Leftrightarrow \Omega_e = \sqrt{\frac{mg}{4C_T}}. \tag{4.2}$$

If the robot is functioning near this operating point, with small variations on the roll and pitch, it's possible to consider that the rotational dynamics become decoupled [47], hence:

$$\dot{\mathbf{\Phi}} \approx \omega, \tag{4.3}$$

$$\mathbf{I}\dot{\omega} = \tau. \tag{4.4}$$

In this way, each direction can be controlled independently, using nested feedback loops by means of linear techniques - the inner loop will use estimates of its current orientation to compute the necessary change on its angular accelerations, while the outer loop is responsible for determining the necessary attitude angles that bring the position and velocity error to zero, in a similar fashion to [50]. This approximation yields the following simplified model:

$$\ddot{x} = (\theta c_\psi + \varphi s_\psi)g \tag{4.5a}$$

$$\ddot{y} = (\theta s_\psi - \varphi c_\psi)g \tag{4.5b}$$

$$\ddot{z} = \frac{F}{m} - g \tag{4.5c}$$

$$\ddot{\varphi} = \frac{\tau_\varphi}{I_x} \tag{4.5d}$$

$$\ddot{\theta} = \frac{\tau_\theta}{I_y} \tag{4.5e}$$

$$\ddot{\psi} = \frac{\tau_\psi}{I_z} \tag{4.5f}$$

Position Controller:

Like observed in Figure 4.1, the desired position $\mathbf{P_d} = (x_d, y_d, z_d)$ and velocity $\mathbf{V_d} = (v_{xd}, v_{y_d}, v_{zd})$ can be inputed to the position controller, which needs to compute the thrust force and necessary attitude angles that bring the robot to this position, through error feedback.

Starting with the height control, it is necessary to determine the thrust force, F, that brings and keeps

the vehicle at a desired altitude, z_d. The error in the Z-direction can de defined as

$$e_z = z_d - z, \tag{4.6a}$$

$$\dot{e}_z = \dot{z}_d - \dot{z} = v_{zd} - v_z, \tag{4.6b}$$

and assuming direct control over the Z-dynamics, the following PID feedback control law is proposed,

$$U_z = K_z^p.e_z + K_z^d.\dot{e}_z + K_z^i.\int_0^t e_z dt, \tag{4.7}$$

where K_z^p, K_z^d and K_z^i are positive constant gains. Yet, to compute the total thrust the motors should produce, it is necessary to compensate for gravity and the non-linear term, yielding the expression

$$F = m(U_z + g). \tag{4.8}$$

Defining, similarly, the error in position and velocity as

$$e_x = x_d - x, \tag{4.9a}$$

$$e_y = y_d - y, \tag{4.9b}$$

$$\dot{e}_x = \dot{x}_d - \dot{x} = v_{xd} - v_x, \tag{4.9c}$$

$$\dot{e}_y = \dot{y}_d - \dot{y} = v_{y_d} - v_y, \tag{4.9d}$$

one can find the necessary desired accelerations on the X and Y-directions that bring the position error to zero, through the following PD control laws,

$$U_x = K_x^p.e_x + K_x^d.\dot{e}_x, \tag{4.10a}$$

$$U_y = K_y^p.e_y + K_y^d.\dot{e}_y, \tag{4.10b}$$

with K_x^p, K_x^d, K_y^p and K_y^d being positive constant gains. Inverting the relationship from equation (4.5a)-(4.5b), it's possible to compute the reference roll and pitch angles from the desired accelerations, by doing

$$\varphi_d = \frac{1}{g}(U_x.s_\psi - U_y.c_\psi), \tag{4.11a}$$

$$\theta_d = \frac{1}{g}(U_x.c_\psi + U_y.s_\psi), \tag{4.11b}$$

and constraining these values to a small neighbourhood around the hover condition,

$$-\frac{\pi}{12} < \varphi_d < \frac{\pi}{12}, \quad -\frac{\pi}{12} < \theta_d < \frac{\pi}{12}. \tag{4.12}$$

Clearly, there is one DOF left, the yaw angle, which can be chosen arbitrarily or given as a user specified command. Therefore, the third torque control variable will not be necessary for the control of

the vehicles position, allowing for an extra degree of manoeuvrability of the robot in flight mode, since it's possible to grant desired yaw movements for specific tasks.

Attitude Controller:

In near-hover conditions, the rotational dynamics can be divided into 3 Single-Input-Single-Output (SISO) systems [47]:

$$\begin{cases} \dot{\varphi} = p \\ \dot{p} = \tau_{\varphi}/I_x \end{cases} \quad (4.13) \qquad \begin{cases} \dot{\theta} = q \\ \dot{q} = \tau_{\theta}/I_y \end{cases} \quad (4.14) \qquad \begin{cases} \dot{\psi} = r \\ \dot{r} = \tau_{\psi}/I_z \end{cases} \quad (4.15)$$

This means the roll, pitch and yaw rotations can be directly controlled by the torques around the x_b, y_b and z_b-directions, respectively. Since a second-order system is regulated through a proportional-derivative controller [49], the procedure is straightforward. The error in orientation is defined as

$$e_{\varphi} = \varphi_d - \varphi, \tag{4.16a}$$

$$e_{\theta} = \theta_d - \theta, \tag{4.16b}$$

$$e_{\psi} = angdiff(\psi_d, \psi), \tag{4.16c}$$

where *angdiff* is the function that returns the smallest angular difference between the current angle and the desired one, wrapped to $]-\pi, \pi]$, and is expressed by

$$angdiff(\psi_d, \psi) = atan2\big[sin(\psi_d - \psi), cos(\psi_d - \psi)\big], \tag{4.17}$$

whereas the angular rate error is given by

$$\dot{e}_{\varphi} = \dot{\varphi}_d - \dot{\varphi} = p_d - p, \tag{4.18a}$$

$$\dot{e}_{\theta} = \dot{\theta}_d - \dot{\theta} = q_d - q, \tag{4.18b}$$

$$\dot{e}_{\psi} = \dot{\psi}_d - \dot{\psi} = r_d - r. \tag{4.18c}$$

Finally, the necessary angular accelerations to track the reference orientation can be found by PD control laws [49], while the corresponding input torques are computed by multiplying the angular accelerations with the moments of inertia, as demonstrated by equation (4.19). These input commands, together with the thrust command given by expression (4.8), stabilise the vehicles dynamics and bring the position error to zero.

$$\tau_{\varphi} = I_x(K^p_{\varphi}.e_{\varphi} + K^d_{\varphi}.\dot{e}_{\varphi}), \tag{4.19a}$$

$$\tau_{\theta} = I_y(K^p_{\theta}.e_{\theta} + K^d_{\theta}.\dot{e}_{\theta}), \tag{4.19b}$$

$$\tau_{\psi} = I_z(K^p_{\psi}.e_{\psi} + K^d_{\psi}.\dot{e}_{\psi}). \tag{4.19c}$$

4.1.2 PID Ground Mode

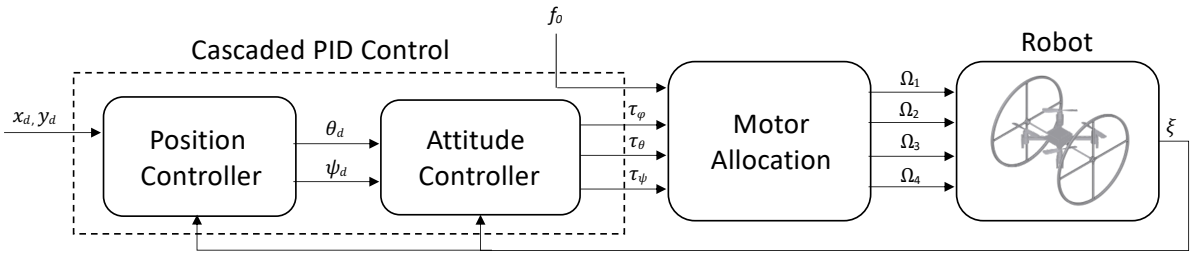


Figure 4.2: PID Ground control of Bimodal aerial robot

As already mentioned in Section 3.4.2, in order to move on a flat surface, the robots collective thrust has to be smaller than its own weight, otherwise it lifts-off from the ground. Putting this, the commanded input is set to a constant $F(t) = f_0$, such that it satisfies condition (3.25),

$$0 < f_0 < mg. \tag{4.20}$$

For the purpose of control design, assuming the friction force, μN, doesn't amply affect the translation of the robot, only acting on the wheels to prevent sliding, the translational dynamics in the ground (or on a flat surface) can be simplified to

$$\dot{v} = s_\theta \frac{f_0}{m}. \tag{4.21}$$

It is however important that a static friction force exists between the wheels and the ground, otherwise the wheel might slide instead of rotating, like mentioned in Section 3.1.

Furthermore, for small pitch angles, it can be considered that each torque only affects the rotation around its direction, as expressed by equation (4.4). However, the angular velocity components cannot be treated as decoupled in this case, since the residual roll movement that is created from pitching and yawing simultaneously needs to be counteracted, to make sure the wheels stay on the ground. The architecture of the PID Ground controller is shown in Figure 4.2.

Position Controller:

Assuming the desired position in the planar surface is given by $\mathbf{P_d} = (x_d, y_d)$, the error vector in the world frame can be defined as

$$\mathbf{e^I} = \mathbf{P_d} - \mathbf{P} \Leftrightarrow \begin{bmatrix} e_x^I \\ e_y^I \end{bmatrix} = \begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} \tag{4.22}$$

with $\mathbf{P} = (x, y)$ being the current position of the centre of mass of the robot. Considering now the rotated version of the error, expressed in the rolling frame,

$$\mathbf{e^R} = \mathbf{R}(\psi)\mathbf{e^I} \Leftrightarrow \begin{bmatrix} e_x^R \\ e_y^R \end{bmatrix} = \begin{bmatrix} c_\psi & s_\psi \\ -s_\psi & c_\psi \end{bmatrix} . \begin{bmatrix} e_x^I \\ e_y^I \end{bmatrix} = \begin{bmatrix} c_\psi(x_d - x) + s_\psi(y_d - y) \\ -s_\psi(x_d - x) + c_\psi(y_d - y) \end{bmatrix}, \tag{4.23}$$

it is trivial to see that if $\mathbf{e^R} \to 0$ then $\mathbf{e^I} \to 0$ and the position converges to the desired one, $\mathbf{P} \to \mathbf{P_d}$.

Since the only admissible velocity is along the x_r-direction, a proportional control law on the error along this axis is used to determine the desired velocity,

$$v_d = K_x.e_x^R, \tag{4.24}$$

implying that the bigger the error is, the faster the robot should move, to arrive at the desired point as quick as possible. In turn, a desired acceleration can be generated by means of a PI controller on the velocity error, given by

$$a_d^R = K_v^p.e_v^R + K_v^i.\int_0^t e_v^R dt, \tag{4.25}$$

where K_x, K_v^p and K_v^i are all constant positive gains and the error is defined as follows,

$$e_v^R = v_d - v. \tag{4.26}$$

Finally, since the thrust force is kept constant, the pitch angle that drives the robot to the desired velocity can be determined by inverting the simplified translational dynamics, given by (4.21), and using the reference acceleration, yielding

$$\theta_d = arcsin(\frac{a_d^R.m}{f_0}), \tag{4.27}$$

which needs to be constrained to only taking values in a neighbourhood around the considered stability point,

$$-\frac{\pi}{6} < \theta_d < \frac{\pi}{6}. \tag{4.28}$$

Conversely, the error along the y_r-direction converges to zero when $\psi \to \psi_d$, meaning the heading angle that turns the robot to the desired orientation can be found by solving

$$e_y^R = 0 \Leftrightarrow s_{\psi_d}(x_d - x) = c_{\psi_d}(y_d - y),$$

which renders

$$\psi_d = atan2(e_y^I, e_x^I) - k\pi. \tag{4.29}$$

The term $k = \{-1, 0, 1\}$ is used to avoid unnecessary turns, taking into account the robots ability to move forwards or backwards and making sure the desired yaw angle sent to the attitude controller is wrapped to the interval $]-\pi, \pi]$. It can be calculated with expression

$$k = \begin{cases} 0 & \text{if } v_d \geq 0 \\ sign\big[atan2(e_y^I, e_x^I)\big] & \text{if } v_d < 0 \end{cases}. \tag{4.30}$$

Attitude Controller:

The inner-loop of the cascaded PID controller is responsible for tracking the reference attitude sent by the position controller, (θ_d, ψ_d), given by equations (4.27) and (4.29), while ensuring the roll angle and rate is kept to zero. This will be done by regulating the torques around the three axis of the body-frame, $\tau = (\tau_\varphi, \tau_\theta, \tau_\psi)$. Again, the rotational kinematics can de divided into 3 sub-systems, yet coupled with each other,

$$\begin{cases} p = -\dot{\psi} s_\theta \\ \dot{p} = \tau_\varphi / I_x \end{cases} \tag{4.31}$$

$$\begin{cases} q = \dot{\theta} \\ \dot{q} = \tau_\theta / I_y \end{cases} \tag{4.32}$$

$$\begin{cases} r = \dot{\psi} c_\theta \\ \dot{r} = \tau_\psi / I_z \end{cases} \tag{4.33}$$

Let's start by defining the angular errors as the smallest angle between the desired orientation and the actual orientation of the body-frame,

$$e_\theta = \theta_d - \theta, \tag{4.34a}$$

$$e_\psi = angdiff(\psi_d, \psi). \tag{4.34b}$$

Since $\dot{\theta} = q$, the pitch dynamics can be stabilised with the following PID control law,

$$U_\theta = K_\theta^p . e_\theta + K_\theta^d (-q) + K_\theta^i \int_0^t e_\theta \, dt, \tag{4.35}$$

where K_θ^p, K_θ^d and K_θ^i are respectively the proportional, derivative and integral positive constant gains.

On the contrary, to track the desired yaw angle, a PD control law is used,

$$U_\psi = K_\psi^p . e_\psi + K_\psi^d . \dot{e_\psi}, \tag{4.36}$$

which generates the reference turning acceleration on the rolling frame (around z_r). In the body-frame, the necessary torques to accomplish this rotational rates, while at the same time ensuring that no roll movement occurs, will be:

$$\tau_\varphi = -I_x s_\theta \, U_\psi, \tag{4.37a}$$

$$\tau_\theta = I_y \, U_\theta, \tag{4.37b}$$

$$\tau_\psi = I_z c_\theta \, U_\psi. \tag{4.37c}$$

Together with f_0, these are the control inputs that will be sent to the Motor Allocation, where the necessary motor velocities of the robot will be computed.

4.1.3 PID Inclined Mode

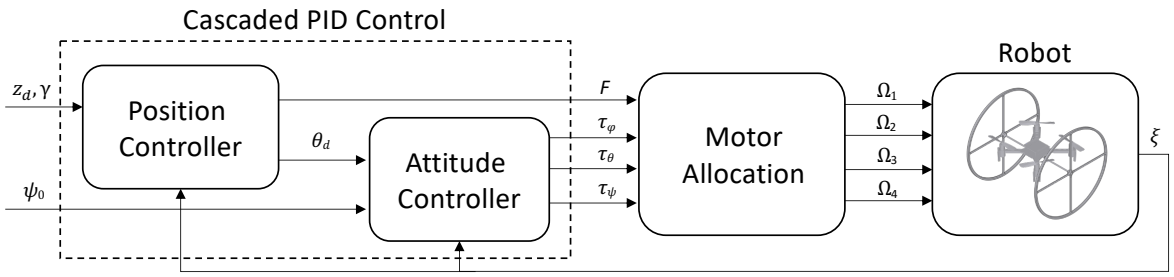


Figure 4.3: PID Inclined control of Bimodal aerial robot

Again, for the case of moving up and down an incline, a stability condition will be studied to simplify the dynamics of the robot, allowing the design of a PID control loop. As depicted in Figure 4.4, an interesting state happens when the thrust force is parallel to the slopes direction, i.e, when the pitch angle is the complement of the inclination angle,

$$\theta = \frac{\pi}{2} - \gamma. \tag{4.38}$$

This particular state has the advantage of being more energy efficient, since the thrust vector is always pointing in the desired direction of movement, resulting in a lower force magnitude being necessary. This increases the battery leverage, which can be an important parameter for lengthier missions.

Evidently, in this case, the thrust force magnitude will be the only input dictating the evolution of the robots state, since z_b is always aligned with x_r. For the robot to stay stationary, it only needs to counteract the lateral component of its weight,

$$F = mg\ s_\gamma, \tag{4.39}$$

which means higher thrust values will result in an upwards motion and smaller values will cause the robot to roll down the slope.

The simplified equation for the translational dynamics can be found by assuming the friction force is negligible, in a similar fashion to the ground mode. Since $s_{(\theta+\gamma)} = s_{\pi/2} = 1$, it is trivial to find

$$\dot{v} = \frac{F}{m} - g s_\gamma. \tag{4.40}$$

The position on top of the inclined surface will be regulated through a feedback loop that determines the correct thrust force to reach the desired point. The position error can be expressed in the rolling frame as

$$e_z^R = \frac{z_d - z}{s_\gamma}, \tag{4.41}$$

which represents the altitude error projected in the direction parallel to the slope, where z_d is the reference height. Similarly as before the reference velocity is found through a proportional control law that will

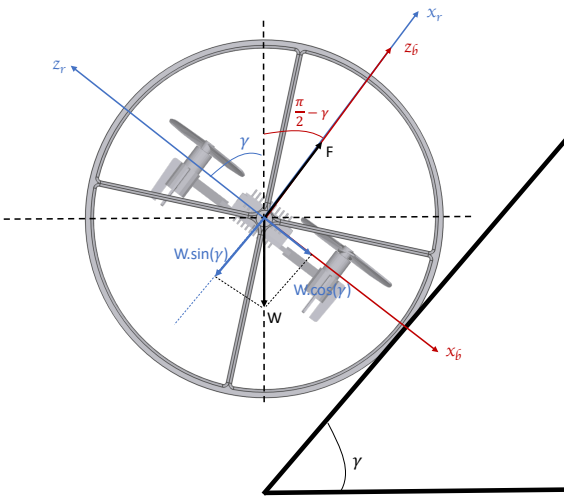


Figure 4.4: Equilibrium case for the inclined mode.

feed, in turn, a velocity error loop that determines the desired acceleration in the x_r (and z_b) direction, as follows:

$$v_d = K_z^p.e_z^R + K_z^i.\int_0^t e_z^R dt \longrightarrow e_v^R = v_d - v, \tag{4.42}$$

$$a_d^R = K_v^p.e_v^R + K_v^d.\dot{e}_v^R + K_v^i.\int_0^t e_v^R dt. \tag{4.43}$$

Inverting relationship (4.40), the thrust input that brings the error to zero can be found,

$$F = m(a_d^R + g\ s_\gamma), \tag{4.44}$$

whereas the remaining inputs, $\tau = (\tau_\varphi, \tau_\theta, \tau_\psi)$, are calculated using control law (4.37). Since the rotational dynamics are the same, the attitude controller has the same structure for flat or inclined surfaces, with the feedforward commands:

$$\begin{cases} \theta_d = \frac{\pi}{2} - \gamma & \text{if } \gamma > 0, \\ \theta_d = -\frac{\pi}{2} - \gamma & \text{if } \gamma < 0, \end{cases} \tag{4.45}$$

$$\psi_d = \psi_0. \tag{4.46}$$

This means that the robot will always be oriented in the slopes direction and will move perpendicularly to its inclination. A negative inclination ($\gamma < 0$) specifies a negative pitch, to take advantage of the robot's ability to move forwards or backwards. It is assumed that the parameters of the slope (inclination and orientation, $\zeta = (\gamma, \psi_0)$) are available to the controller *a priori*.

4.2 Feedback Linearisation with stabilisation of internal dynamics

The linear approach considered in the previous Section only allows for certain angles of movement around the working condition, which means it underperforms when submitted to trajectories that are

more aggressive, and is not robust against disturbances. Alternatively, a feedback linearising controller is proposed in this Section, that allows the use of the full dynamic model of the system, thus being more adequate for complex manoeuvres.

The purpose of this method is to design a feedback control law that transforms the closed-loop system into an equivalent, linear and controllable one, under state transformation. However, performing an exact linearisation on the ground dynamics would result in a singularity when the vehicle is stopped ($v = 0$), which has been proven to be structural for nonholonomic systems [67]. Instead of a full state feedback, it was decided to apply a partial dynamic inversion to compute the control inputs, while the residual dynamics (position and linear velocity) are tracked by an outer loop.

As a result, the controller will comprise three different layers: an outer layer responsible for stabilising the internal dynamics of the system and determining the reference attitude commands, an intermediate layer to track the internal and external reference commands, and an innermost layer which implements the dynamic compensator that produces the input commands sent to the robot. Again, it will be necessary to design a variation of the controller for each modality of the robot, presented below.

4.2.1 Dynamic Feedback Linearisation (DFL)

Feedback linearisation is based on a non-linear change of coordinates and non-linear state feedback and can be achieved through a dynamic control law [59]. Given the non-linear system

$$\dot{\xi} = \mathbf{f}(\xi) + \mathbf{g}(\xi).\mathbf{u}, \quad \xi \in R^n,\ \mathbf{u} \in R^m, \tag{4.47}$$

where ξ is the state vector, $\mathbf{u}$ is the control vector, n is the number of states and m the number of control inputs, the idea is to find a dynamic compensator of the form

$$\mathbf{u} = \alpha(\xi) + \beta(\xi).\vartheta, \tag{4.48}$$

where ϑ is an external reference to be defined later, such that in closed loop the system is equivalent to a linear one [68].

In order to find the dynamic state feedback that linearises the system, one must choose an m-dimensional output,

$$\eta = \mathbf{h}(\xi), \tag{4.49}$$

and differentiate it successively (r times) until the inputs appear in a **nonsingular** way [56], i.e, in the form

$$\eta^{(r)} = l(\xi) + \mathbf{J}(\xi).\mathbf{u}. \tag{4.50}$$

Thereafter, by choosing the parameters of the dynamic compensator to be

$$\alpha(\xi) = -\mathbf{J^{-1}}(\xi).l(\xi), \tag{4.51a}$$

$$\beta(\xi) = \mathbf{J^{-1}}(\xi), \tag{4.51b}$$

in such a way that the control law takes the form

$$\mathbf{u} = \mathbf{J^{-1}}(\xi).\big[\vartheta - l(\xi)\big], \tag{4.52}$$

yields a closed loop integrator chain:

$$\eta^{(r)} = \vartheta. \tag{4.53}$$

This system can be simply controlled by adopting a polynomial control law for the virtual reference ϑ, i.e,

$$\vartheta = \eta_d^{(r)} + \sum_{j=1}^{r} \mathbf{K}_{j-1}.\mathbf{e}^{(j-1)}, \tag{4.54}$$

where $\eta_d^{(r)}$ are the feedforward terms, $\mathbf{K}$ is an $m \times r$ matrix of constant positive gains, $\mathbf{e} = \eta_d - \eta$ is the tracking error and $\mathbf{e}^{(j-1)}$ its successive differentiations.

Accordingly, the change of coordinates $\mathbf{Z} = \phi(\xi)$, representing the desired outputs to control η and its (r) differentiations,

$$\begin{cases} Z_1 = h_1(\xi) \\ ... \\ Z_{r+1} = h_1^{(r)}(\xi) \\ Z_{r+2} = h_2(\xi) \\ ... \\ Z_{m*r} = h_m^{(r)}(\xi) \end{cases}, \tag{4.55}$$

transforms the system into an equivalent fully linear and controllable one [68], with a simple canonical structure of the form

$$\begin{cases} \dot{\mathbf{Z}} = \mathbf{A}.\mathbf{Z} + \mathbf{B}.\vartheta \\ \mathbf{y} = \mathbf{C}.\mathbf{Z} \end{cases}, \tag{4.56}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ have very simple arrangements.

4.2.2 DFL Flight Mode

Considering the case when the robot is operating in the air, the inner layer of the robot will determine the necessary inputs to track the reference height and attitude, while the outer loop will be responsible for stabilising the internal states (x and y), determining the reference roll and pitch angles to track the desired position, as illustrated by Figure 4.5, in the **Position Stabilization** block.

The **Tracking + Feedforward** component is in charge of tracking the reference height and attitude, determining the virtual references that are sent to the **Dynamic Compensator**, which in turn generates the input commands through feedback linearisation. It is assumed the full state of the robot is available and like before, the yaw angle can be chosen by the user as an extra degree-of-freedom.

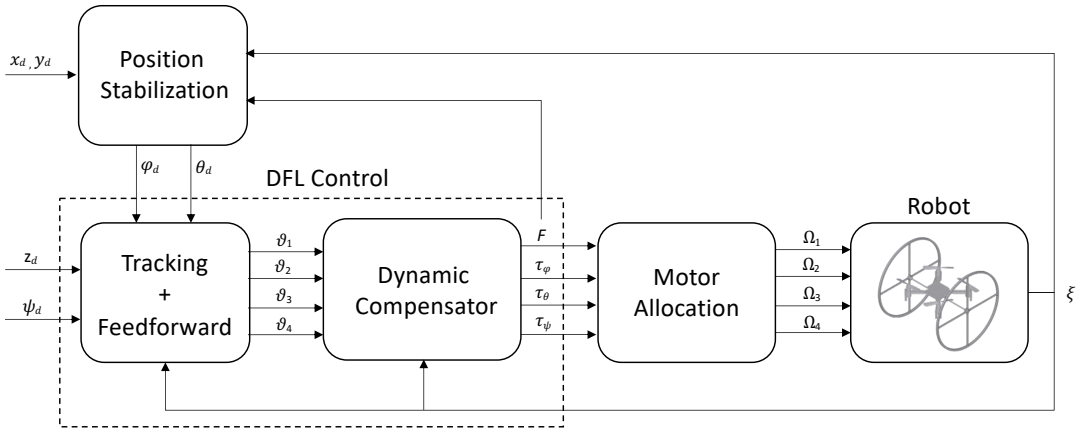


Figure 4.5: DFL Flight control of Bimodal aerial robot.

DFL Controller:

The output to be controlled is the height and the attitude,

$$\eta_f = [z, \varphi, \theta, \psi]^T, \tag{4.57}$$

which can be differentiated to find, according to (3.23),

$$\dot{\eta}_f = \begin{bmatrix} \dot{z} \\ \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v_z \\ p + s_\varphi t_\theta q + c_\varphi t_\theta r \\ c_\varphi q - s_\varphi r \\ s_\varphi sec(\theta) q + c_\varphi sec(\theta) r \end{bmatrix}. \tag{4.58}$$

The input $\mathbf{u}$ appears in a nonsingular way deriving the output vector once more, resulting in

$$\ddot{\eta}_f = \begin{bmatrix} \ddot{z} \\ \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = l_f(\xi) + \mathbf{J_f}(\xi).\mathbf{u_f}, \tag{4.59}$$

with:

- $\mathbf{J_f}(\xi) = \begin{bmatrix} \frac{c_\varphi c_\theta}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & \frac{s_\varphi t_\theta}{I_y} & \frac{c_\varphi t_\theta}{I_z} \\ 0 & 0 & \frac{c_\varphi}{I_y} & -\frac{s_\varphi}{I_z} \\ 0 & 0 & \frac{s_\varphi}{I_y . c_\theta} & \frac{c_\varphi}{I_z . c_\theta} \end{bmatrix}$
- $l_{f_1}(\xi) = -g$
- $l_{f_2}(\xi) = \frac{I_y - I_z}{I_x} qr + \frac{I_z - I_x}{I_y} prs_\varphi t_\theta + \frac{I_x - I_y}{I_z} pqc_\varphi t_\theta + \dot{\varphi} t_\theta (c_\varphi q - s_\varphi r) + \dot{\theta} sec(\theta)(s_\varphi q + c_\varphi r)$
- $l_{f_3}(\xi) = \frac{I_z - I_x}{I_y} prc_\varphi - \frac{I_x - I_y}{I_z} pqs_\varphi - \dot{\varphi}(s_\varphi q + c_\varphi r)$

- $l_{f4}(\xi) = \frac{I_z - I_x}{I_y} prs_\varphi sec(\theta) + \frac{I_x - I_y}{I_z} pqc_\varphi sec(\theta) + \dot{\varphi} sec(\theta)(c_\varphi q - s_\varphi r) + \dot{\theta} t_\theta sec(\theta)(s_\varphi q + c_\varphi r)$
- $l_f(\xi) = [l_{f1}, l_{f2}, l_{f3}, l_{f4}]^T$

The tracking component $\vartheta_f = [\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4]^T$ is then chosen according to expression (4.54),

$$\vartheta_1 = \ddot{z}_d + k_z^1(\dot{z}_d - \dot{z}) + k_z^0(z_d - z), \tag{4.60a}$$

$$\vartheta_2 = \ddot{\varphi}_d + k_\varphi^1(\dot{\varphi}_d - \dot{\varphi}) + k_\varphi^0(\varphi_d - \varphi), \tag{4.60b}$$

$$\vartheta_3 = \ddot{\theta}_d + k_\theta^1(\dot{\theta}_d - \dot{\theta}) + k_\theta^0(\theta_d - \theta), \tag{4.60c}$$

$$\vartheta_4 = \ddot{\psi}_d + k_\psi^1(\dot{\psi}_d - \dot{\psi}) + k_\psi^0(\psi_d - \psi). \tag{4.60d}$$

Considering the control law for the flight mode is determined through equation (4.52), the decoupling matrix $\mathbf{J_f}(\xi)$ has to be invertible. Since the determinant is given by

$$det(\mathbf{J_f}) = \frac{c_\varphi}{m I_x I_y I_z}, \tag{4.61}$$

the matrix is invertible for all $\det(\mathbf{J_f}) \neq 0 \Leftrightarrow \varphi \neq \pm\frac{\pi}{2}$, yielding

$$\mathbf{J_f^{-1}}(\xi) = \begin{bmatrix} \frac{m}{c_\varphi c_\theta} & 0 & 0 & 0 \\ 0 & I_x & 0 & -I_x s_\theta \\ 0 & 0 & I_y c_\varphi & I_y s_\varphi c_\theta \\ 0 & 0 & -I_z s_\varphi & I_z c_\varphi c_\theta \end{bmatrix}, \quad \theta \neq \pm\frac{\pi}{2}, \tag{4.62}$$

which allows to formulate the expression for the input commands as

$$\mathbf{u} = \mathbf{J_f^{-1}}(\xi).\big[\vartheta_f - l_f(\xi)\big]. \tag{4.63}$$

Nonetheless, this controller presents a singularity when $\varphi = \pm\frac{\pi}{2}$ or $\theta = \pm\frac{\pi}{2}$, which has to be taken into account by the outer layers.

Position Stabilisation:

As for the internal states, x and y, that remain uncontrolled, their dynamics are given by

$$\begin{cases} \ddot{x} = (c_\varphi s_\theta c_\psi + s_\varphi s_\psi)\frac{F}{m} \\ \ddot{y} = (c_\varphi s_\theta s_\psi - s_\varphi c_\psi)\frac{F}{m} \end{cases}. \tag{4.64}$$

Position stabilisation can be achieved through a similar approach as in the linear case, by considering the desired accelerations are found with the following PD control laws:

$$U_x = K_x^p.e_x + K_x^d.\dot{e}_x, \tag{4.65a}$$

$$U_y = K_y^p.e_y + K_y^d.\dot{e}_y, \tag{4.65b}$$

where K_x^p, K_x^d, K_y^p and K_y^d are positive constant gains and e_x, e_y the tracking errors.

$[U_x, U_y]$ can be seen as the direction of the acceleration vector that brings the robot to the desired position, meaning that by inverting relationship (4.64), it's possible to compute the reference roll and pitch angles, with

$$\begin{cases} \varphi_d = arcsin\left[\frac{m}{F}(U_y.c_\psi - U_x.s_\psi)\right] \\ \theta_d = arcsin\left[\frac{m}{Fc_\varphi}(U_x.c_\psi + U_y.s_\psi)\right] \end{cases}. \tag{4.66}$$

To avoid allowing the system near the singularity point, the reference roll and pitch angles are constrained between the values

$$-\frac{\pi}{3} < \varphi_d < \frac{\pi}{3}, \quad -\frac{\pi}{3} < \theta_d < \frac{\pi}{3}. \tag{4.67}$$

4.2.3 DFL Surface Mode

When the robot is in terrestrial locomotion, the input torques will be determined by dynamic inversion, while the thrust force is computed in the outer layer, together with the reference angles, like seen in Figure 4.6. Some differences will have to be considered between flat and inclined ground, similarly with the previous sections.

For the case of rolling on a surface, the **Position Stabilisation** block will be responsible for computing the reference pitch and thrust force to move the vehicle from the current position to the desired one. The yaw angle will be computed so that the orientation of the vehicle is aligned with the destination, for the case of flat ground, whereas in a slope it will be a constant defined by the specifications of the incline (ζ).

On the other hand, the **Dynamic Compensator** is the same for both the cases of flat or inclined surfaces, tracking the reference attitude commands while ensuring the wheels are always in contact with the surface. For this reason, the three rotational directions are considered, to ensure no change occurs in the roll rate, which is kept to zero. This also offers to opportunity of having a more resilient controller, which will return to the supposed pose in case of a disturbance that causes the robot to roll.

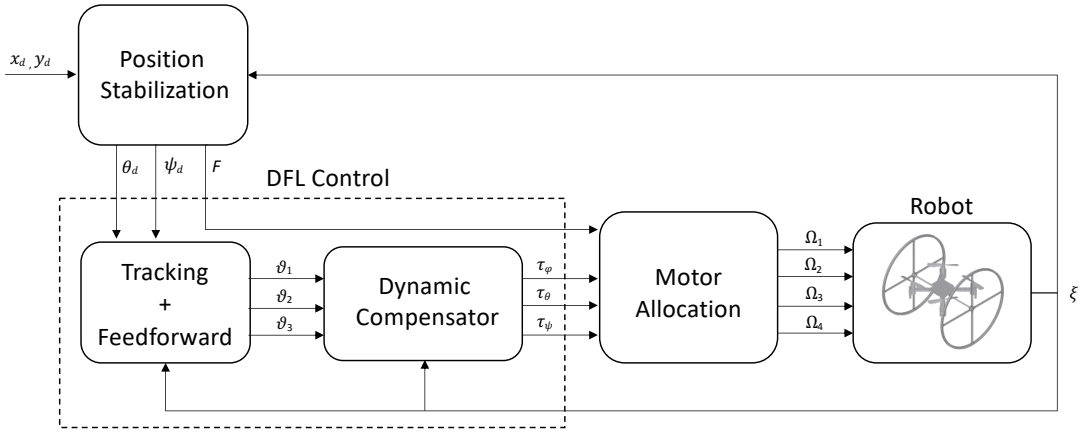


Figure 4.6: DFL Surface control of Bimodal aerial robot

DFL Controller:

The output vector will consist of the three attitude angles,

$$\eta_s = [\varphi, \theta, \psi]^T. \tag{4.68}$$

Again the vector is differentiated until the input appears in a nonsingular way, which happens after two differentiations,

$$\dot{\eta}_s = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p + \dot{\psi} s_\theta \\ q \\ r\ sec(\theta) \end{bmatrix} = \begin{bmatrix} p + r t_\theta \\ q \\ r\ sec(\theta) \end{bmatrix}, \tag{4.69a}$$

$$\ddot{\eta}_s = \begin{bmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = l_s(\xi) + \mathbf{J_s}(\xi).\tau, \tag{4.69b}$$

with:

- $\mathbf{J_s}(\xi) = \begin{bmatrix} \frac{1}{I_x} & 0 & \frac{t_\theta}{I_z} \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & \frac{1}{c_\theta I_z} \end{bmatrix}$
- $l_{s1}(\xi) = \frac{I_y - I_z}{I_x} qr + \frac{I_x - I_y}{I_z} pq\ t_\theta + qr\ sec^2(\theta)$
- $l_{s2}(\xi) = \frac{I_z - I_x}{I_y} pr$
- $l_{s3}(\xi) = \frac{I_x - I_y}{I_z} pq\ sec(\theta) + r t_\theta\ sec(\theta)$
- $l_s(\xi) = [l_{s1}, l_{s2}, l_{s3}]^T$

In contrast, the tracking component for the surface mode, $\vartheta_s = [\vartheta_{s1}, \vartheta_{s2}, \vartheta_{s3}]^T$, is chosen taking into account the imposed conditions (3.24)-(3.25), resulting in

$$\vartheta_{s1} = k^1_\varphi(-\dot{\varphi}) + k^0_\varphi(-\varphi), \tag{4.70a}$$

$$\vartheta_{s2} = k^1_\theta(-\dot{\theta}) + k^0_\theta(\theta_d - \theta), \tag{4.70b}$$

$$\vartheta_{s3} = k^1_\psi(\dot{\psi}_d - \dot{\psi}) + k^0_\psi(\psi_d - \psi), \tag{4.70c}$$

where the θ_d and ψ_d references are generated by the outer layer but $\dot{\psi}_d$ is a feedforward command that can be found using the differential flatness properties of a unicycle [64] - given the desired trajectory, $\mathbf{P_d}(t) = (x_d(t), y_d(t))$, then

$$\dot{\psi}_d = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}^2_d(t) + \dot{y}^2_d(t)}. \tag{4.71}$$

The roll component, ϑ_{s1}, is not set to zero to guarantee that, in case of a disturbance that takes the wheels off the ground, the controller compensates and tries to remain in contact with the surface.

Considering that the determinant of the decoupling matrix $\mathbf{J_s}(\xi)$,

$$det(\mathbf{J_s}) = \frac{1}{I_x I_y I_z c_\theta}, \tag{4.72}$$

is nonsingular for all $\theta \neq \pm\frac{\pi}{2}$, it can be inverted to find

$$\mathbf{J_s^{-1}} = \begin{bmatrix} I_x & 0 & -I_x.s_\theta \\ 0 & I_y & 0 \\ 0 & 0 & I_z.c_\theta \end{bmatrix}, \tag{4.73}$$

thus the dynamic compensator for surface mode (both flat and inclined cases) takes the form

$$\tau = \mathbf{J_s^{-1}}(\xi).\big[\vartheta_s - l_s(\xi)\big]. \tag{4.74}$$

Flat Position Stabilisation:

To stabilise the remaining outputs, a very similar approach as the one considered in the linear case will be undertaken, this time around allowing the computed reference angles to belong to a bigger subspace. Furthermore, since one of the main setbacks of the PID controller was the thrust input being a constant $F(t) = f_0$, a valve input control strategy will now allow to regulate it when the pitch variable gets saturated [69], allowing to increase the thrust force when necessary but keeping it in its nominal value f_0 for shorter movements.

To begin with, the reference velocity is found through a proportional control law on the rotated error, through expression (4.24), which in turn feeds a loop on the velocity error to calculate the reference acceleration, like (4.25). Summarising:

$$v_d = K_x.e_x^R \longrightarrow e_v^R = v_d - v \tag{4.75a}$$

$$a_d^R = K_v^p.e_v^R + K_v^i.\int_0^t e_v^R dt \tag{4.75b}$$

Like before, inverting the translational velocity expression and using the reference acceleration, the reference pitch is calculated as

$$\theta_d = arcsin(a_d^R.\frac{m}{f_0}). \tag{4.76}$$

However, to avoid dominion errors when computing the *arcsine* function, the acceleration has to be saturated to guarantee that the pitch angle is a real number,

$$-\frac{f_0}{m} = a_d^{min} < a_d < a_d^{max} = \frac{f_0}{m}. \tag{4.77}$$

Moreover, to maintain the controller away from the singularity point, the pitch angle is also saturated between the values

$$-\frac{\pi}{3} = \theta_d^{min} < \theta_d < \theta_d^{max} = \frac{\pi}{3}. \tag{4.78}$$

Evidently, there can be a loss in the real acceleration provided to the vehicle, after these saturations, which can be averted by increasing the thrust force, instead of always keeping it in its nominal value. As such, the following deviation variables were defined:

$$\Delta a = \begin{cases} a_d - a_d^{max} & \text{if } a_d > a_d^{max} \\ 0 & \text{if } a_d^{min} \leq a_d \leq a_d^{max} \\ |a_d - a_d^{min}| & \text{if } a_d < a_d^{min} \end{cases}, \tag{4.79}$$

$$\Delta \theta = \begin{cases} \theta_d - \theta_d^{max} & \text{if } \theta_d > \theta_d^{max} \\ 0 & \text{if } \theta_d^{min} \leq \theta_d \leq \theta_d^{max} \\ |\theta_d - \theta_d^{min}| & \text{if } \theta_d < \theta_d^{min} \end{cases}, \tag{4.80}$$

that represent the amount of acceleration "lost" in the saturation process. Therefore, the thrust input is defined as

$$F = f_0 + K_a^f.\Delta a + K_\theta^f.\Delta \theta, \tag{4.81}$$

where f_0 is the thrust nominal value and K_a^f, K_θ^f are positive constant gains. The total thrust must still satisfy condition (3.25), to avoid exceeding the lift-off force.

This scheme improves the dynamic performance of the system, since the primary variable, θ_d, always controls the output velocity v for the fast control, while the secondary input, F, will be kept at its nominal value until the first reaches its upper or lower limit [69], being responsible for the long-term control. A quick analysis allows to understand that the robot will pitch towards the desired point, until reaching its cap, moment when the thrust force increases to reach the necessary velocity; once the vehicle is closer to its destiny, it will pitch in the opposite direction, to start breaking, until eventually it reaches the saturation point and once again the thrust force will increase to make sure the robot breaks in time.

Finally, the yaw reference is once again found by considering that the orientation of the wheels need to be aligned with the desired point,

$$\psi_d = atan2(e_y^I, e_x^I) - k\pi.$$

Inclined Position Stabilisation:

The cascaded PID controller for the inclined case, that was studied in Section 4.1.3, relied on maintaining the vehicle on a static pose, only increasing or decreasing the thrust force to move up and down, respectively. This can cause some constraints, especially for low inclinations, since the required pitch angle will magnify the robots velocity to values difficult to counteract only with the thrust magnitude. For this reason, to track the position, a valve scheme will be applied instead, allowing to regulate the pitch angle for the primary control and the thrust force when the pitch gets saturated.

The reference velocity is proportional to the altitude error projected on the Rolling Frame, which allows to calculate the necessary acceleration on the longitudinal direction, to achieve that velocity,

through error feedback. Summing up:

$$v_d = K_z^p.e_z^R, \tag{4.82}$$

$$a_d^R = K_v^p.(v_d - v), \tag{4.83}$$

with $K_z^p, K_v^p > 0$ being the proportional gains, and where the position error e_z^R is calculated as in (4.41).

Once again, it is assumed that the friction on the wheels is only relevant to avoid sliding, and its contribution for the robot's translation is thus ignored. The pitch angle is found by inverting relationship (3.40) and neglecting the friction force,

$$\theta_d = arcsin\big[\frac{m}{f_0}(a_d^R + g.s_\gamma)\big] - \gamma, \tag{4.84}$$

hence, there is the need to saturate the value that goes inside the *arcsine* function

$$-\frac{f_0}{m} - gs_\gamma = a_d^{min} < a_d < a_d^{max} = \frac{f_0}{m} - gs_\gamma. \tag{4.85}$$

Similarly, the reference pitch angle is also restrained to a subspace,

$$\begin{cases} \theta_d^{min} = 0 < \theta_d < \frac{\pi}{2} - \gamma = \theta_d^{max} & \text{if } \gamma > 0 \\ \theta_d^{min} = -\frac{\pi}{2} - \gamma < \theta_d < 0 = \theta_d^{max} & \text{if } \gamma < 0 \end{cases} \tag{4.86}$$

that allows the correct functioning of the controller and ensures the wheels stay in contact with the surface, which could not be prevented if, for example, the robot pitched in the direction opposite to the surface. A negative inclination angle ($\gamma < 0$) specifies once more that the robot should pitch in the negative direction instead.

As for the thrust force, it will remain in its nominal value,

$$f_0 = mg\ s_\gamma, \tag{4.87}$$

which is the equilibrium force that guarantees the vehicle stays at hover when stopped, and will increase or decrease when either the acceleration or the pitch angle reach its upper or lower limit, respectively. The deviation variables are defined accordingly,

$$\Delta a = \begin{cases} a_d - a_d^{max} & \text{if } a_d > a_d^{max} \\ 0 & \text{if } a_d^{min} \leq a_d \leq a_d^{max} \\ a_d - a_d^{min} & \text{if } a_d < a_d^{min} \end{cases}, \tag{4.88}$$

$$\Delta\theta = \begin{cases} \theta_d - \theta_d^{max} & \text{if } \theta_d > \theta_d^{max} \\ 0 & \text{if } \theta_d^{min} \leq \theta_d \leq \theta_d^{max} \\ \theta_d - \theta_d^{min} & \text{if } \theta_d < \theta_d^{min} \end{cases}, \tag{4.89}$$

where it can be noted that the lower variation is no longer in absolute values, since in this case we will want the thrust force to actually decrease. The expression for the thrust force is given exactly as before, through (4.81), and the yaw must be restrained to the value of the slope orientation, $\psi_d = \psi_0$, once more.

Chapter 5

Simulation and Results

In this chapter, the methodology utilised to set the simulation environment is briefly described. This framework is used to test the developed controllers and compare them against each other, assessing the closed-loop response of the proposed system. Finally, two examples of a hybrid mission are shown, in which the robot moves in different surfaces and successfully transitions between modes.

5.1 Framework for the simulation of Bimodal robots

A simulation environment was created using *RotorS* Simulator [9], a well-known tool to assess the behaviour of aerial robots, that uses the ROS+Gazebo paradigm. ROS [70] provides the ability to easily test robotic algorithms, interfacing its functionalities with Gazebo Simulator [71], comprising a physics engine that effectively reproduces real-life rigid body dynamics. *RotorS* is a UAV simulation framework, designed in a modular way, allowing to easily test control and path planning algorithms for aerial platforms. It provides a structure that simulates realistic conditions, in which the sensors attached to the robot provide information to a ROS node running the controller in the background, that updates the motor velocities, simulating the behaviour of the robot.

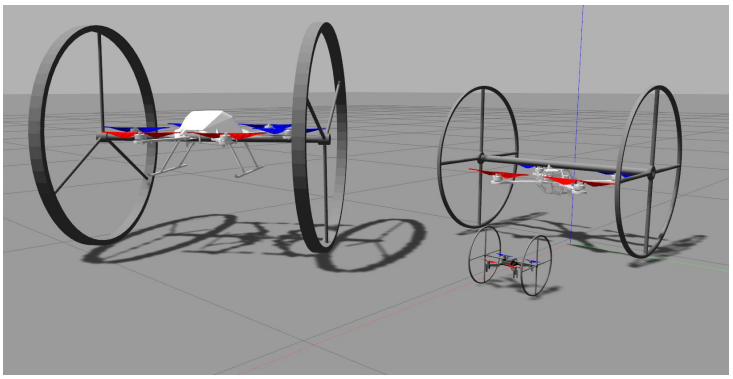


Figure 5.1: Bimodal aerial robots in the Gazebo simulation environment.

The architecture of the program, built on top of *RotorS*, followed a generic approach to grant the possibility of attaching different wheels to any available UAV on the simulation and test the performance

of any developed controller. The proposed Bimodal robot was then modelled in the simulator by creating a structure that adds 2 passive wheels to any of the default aerial vehicles shipped with the package, like displayed in Figure 5.1.

A macro that generates the passive wheels links and joints and another that calculates its inertia values was created. This allows to add the wheels to any simulated UAV by simply calling the macro and specifying the wheels mass, width, inner and outer radius. In addition, it is also possible to define the number of spokes and its mass if they influence the inertia values. The calculated inertia assumes the wheel's rim can be approximated by a hollow cylinder and the spokes as thin rods, whose formulas are available for consultation in appendix B. The experiments present in this chapter were performed using the *Wheelbird* Bimodal robot, which is a two-wheeled *Hummingbird*, the first robot on the right of Figure 5.1.

5.1.1 Controller implementation

The controller Node that implements the motion control systems described in chapter 4, reads from two different parameter files, which initialise the vehicle specifications - mass, inertia tensor, arm length, rotors thrust and moment constants - and the controller gains. As illustrated by Figure 5.2, the node subscribes to the sensors information provided by gazebo and to a topic where the setpoint is specified:

- '/odometry_sensor1/odometry' contains data from the current pose of the robot;
- '/IMU_sensor' is a topic containing information regarding the robots linear and angular velocities, as well as accelerometer data;
- '/desired_position', where the desired waypoint and controller mode can be published by the user.

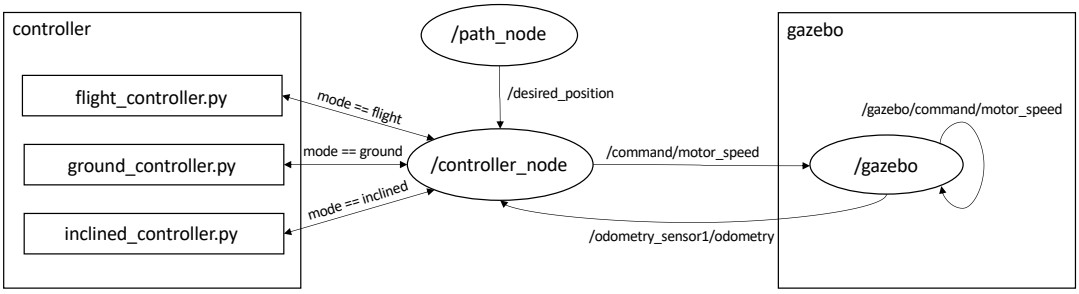


Figure 5.2: ROS Nodes and Topics.

Depending on the specified mode, the control Node calls the corresponding motion controller and publishes back the updated motor velocities through the '/command/motor_speed' topic. Since the output of the controllers is the thrust and moments, the node has to first implement the Motor Allocation, i.e, the inverse of equation (3.14), outlined in algorithm 1. This algorithm runs in each iteration of the controller, making sure that the take-off threshold is not surpassed in Ground mode and no negative motor velocities are requested (which would be physically impossible for the robot).

Algorithm 1: Motor Allocation

Result: Computes the 4 motor speeds from the input thrust and torques

$\Omega^2 = A^{-1}.u$
for Ω_i^2 ***in*** Ω^2 **do**
  if *Mode == Ground* ***and*** $\Omega_i^2 > \Omega_e^2$ **then**
    $\Omega_i^2 = \Omega_e^2$
  end
  if $\Omega_i^2 < 0$ **then**
    $\Omega_i^2 = 0$
  end
end
$\Omega = sqrt(\Omega^2)$

RotorS simulator also provides a state estimator that can be turned on by an argument. However, for the aim of testing the controllers, the experiments provided in this thesis work use the ground truth as the vector of states, which are passed through the odometry and IMU topics.

5.1.2 Transition between modes

Together with the specified setpoint, the developed planner also accepts mobility modes - flight, ground or inclined - or it can receive a take-off or landing command, to allow transition between flying and moving on a surface:

- **Take-off Command -** Takes the robot from ground or a slope to flight mode, rising vertically.
- **Land Command -** This command signals the robot to smoothly descend until the planner detects that the robot has landed on a surface (through IMU sensor data).

Algorithm 2: Landing function

Landed = False
Time landed = 0
while *Mode == Land* **do**
  Thrust = 0.95mg
  if *Landed == True* **then**
    while *Time landed* $<$ *Time treshold* **do**
      if $|v_x| > 0$ *and* $v_z < 0$ **then**
        Gamma = atan2(v_z, vx)
        Mode = Inclined
        break
      end
      Time landed = Time landed + Δt
    end
    Mode = Ground
  end
  if $a > a_{min}$ **then**
    Landed = True
  else
    Landed = False
  end
end

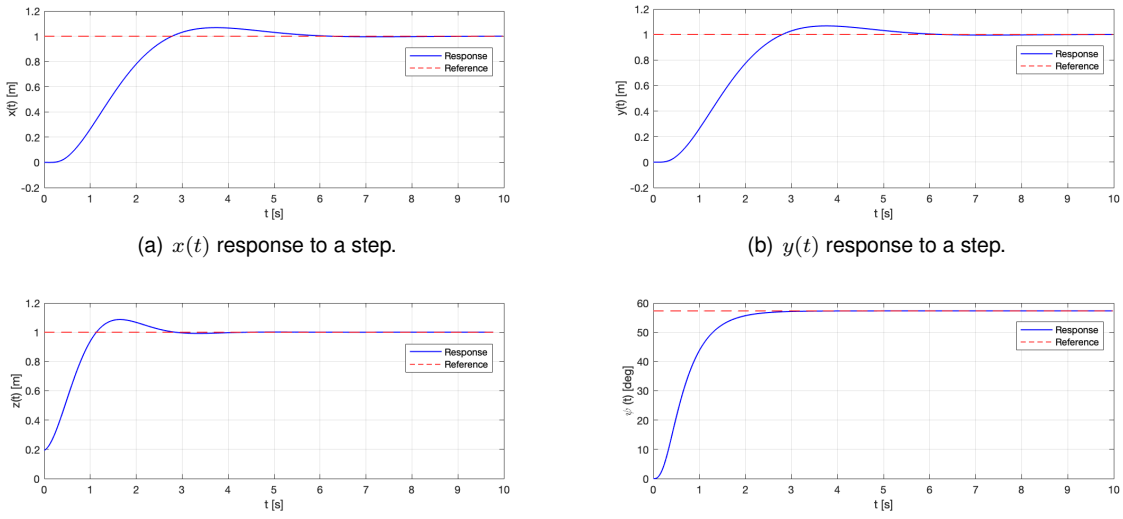
When the robot is landing, the controller enters in an idle state until it detects a spike in the accelerometer values. Afterwards, for a small time duration, if the vehicles keeps descending in a certain direction, the planner predicts that the robot has landed on a slope, estimates the inclination angle (using the longitudinal and vertical velocity values) and activates the inclined controller automatically. Otherwise, the ground controller is switched on. Algorithm 2 shows the structure of this function, where a represents the vertical acceleration value provided by the accelerometer and a_{min} is a threshold value to detect the sudden change in the accelerometer values.

Transitioning between flat ground and inclined surfaces is also possible, given that the distance to travel is small and the inclination angle is known. This means that the robot should be close to the slope base, to transition from ground to inclined state and vice-versa. In different circumstances, the controller might find unexpected states and possibly become unstable. In section 5.5, some examples of a hybrid navigation mission, using all three modes and transitions, will be given.

5.2 Flight Mode Simulation

The flying mode of the Bimodal robot works as a typical quadrotor, like discussed in the previous sections, thus its behaviour is not extensively reported in this section. Instead, the focus is put on the difference between the linear and non-linear controller and comparing the dynamic response when performing a flight or rolling on a surface.

5.2.1 PID Flight controller results



(a) $x(t)$ response to a step.

(b) $y(t)$ response to a step.

(c) $z(t)$ response to a step.

(d) $\psi(t)$ response to a step.

Figure 5.3: PID Flight controller: Position and Yaw response to a step input.

Firstly, a step test was performed on the four trajectory inputs of the flight controller, whose response is used to tune the controller gains, adjusting the values empirically to get the best trade-off between oscillatory behaviour and response speed.

Figure 5.3 shows the response of the system when submitted to a step signal in each of the outputs (x, y, z, ψ). As expected, the altitude and yaw response are faster since they are controlled directly by the thrust and yaw torque, respectively. The position along the longitudinal and lateral directions have bigger response times and present slight overshoots, as reported in Table 5.1, since they are dependent on the stabilisation of the pitch and roll angles, accordingly, which are restricted to taking values only in a small neighbourhood around the hover condition.

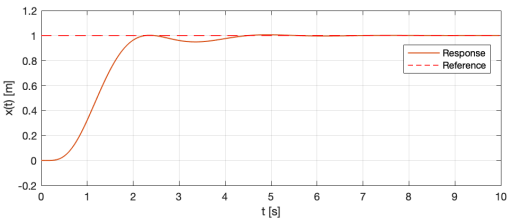
	$x(t)$	$y(t)$	$z(t)$	$\psi(t)$
Rise Time [s]	1.69	1.70	0.78	1.15
Settling Time [s]	5.31	5.33	2.56	2.17
Overshoot	6.80 %	6.74 %	8.7 %	0 %

Table 5.1: Step response characteristics of PID Flight controller.

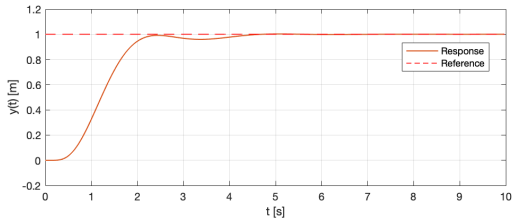
5.2.2 DFL Flight controller results

Looking instead to the results obtained when using the DFL controller, displayed in Figure 5.4, it is noticeable the improvement on the performance of the controller. In particular, the altitude and yaw response are very fast, taking about one second to converge, even though the Z-direction presents a constant steady-state error of $e_z = 0.0142$ m, due to lack of integral action.

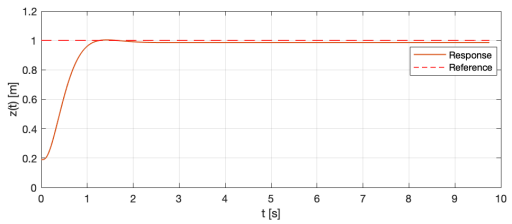
It can be seen from the data in Table 5.2 that the overshoot is almost eliminated, while the settling time of the longitudinal and lateral directions sees an improvement, taking now about 4 seconds. These results suggest that the non-linear approach enhanced the closed-loop performance of the system, however, to different extents: whilst the z and ψ response times decrease about 60%, the x and y decrease 25%. This is a consequence of having a cascaded architecture, where the attitude and altitude are controlled with a feedback linearising law, while the outer-loop determines the necessary orientation to reach the desired setpoint, which will take longer to converge.



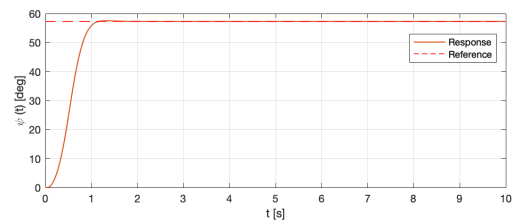
(a) $x(t)$ response to a step.



(b) $y(t)$ response to a step.



(c) $z(t)$ response to a step.



(d) $\psi(t)$ response to a step.

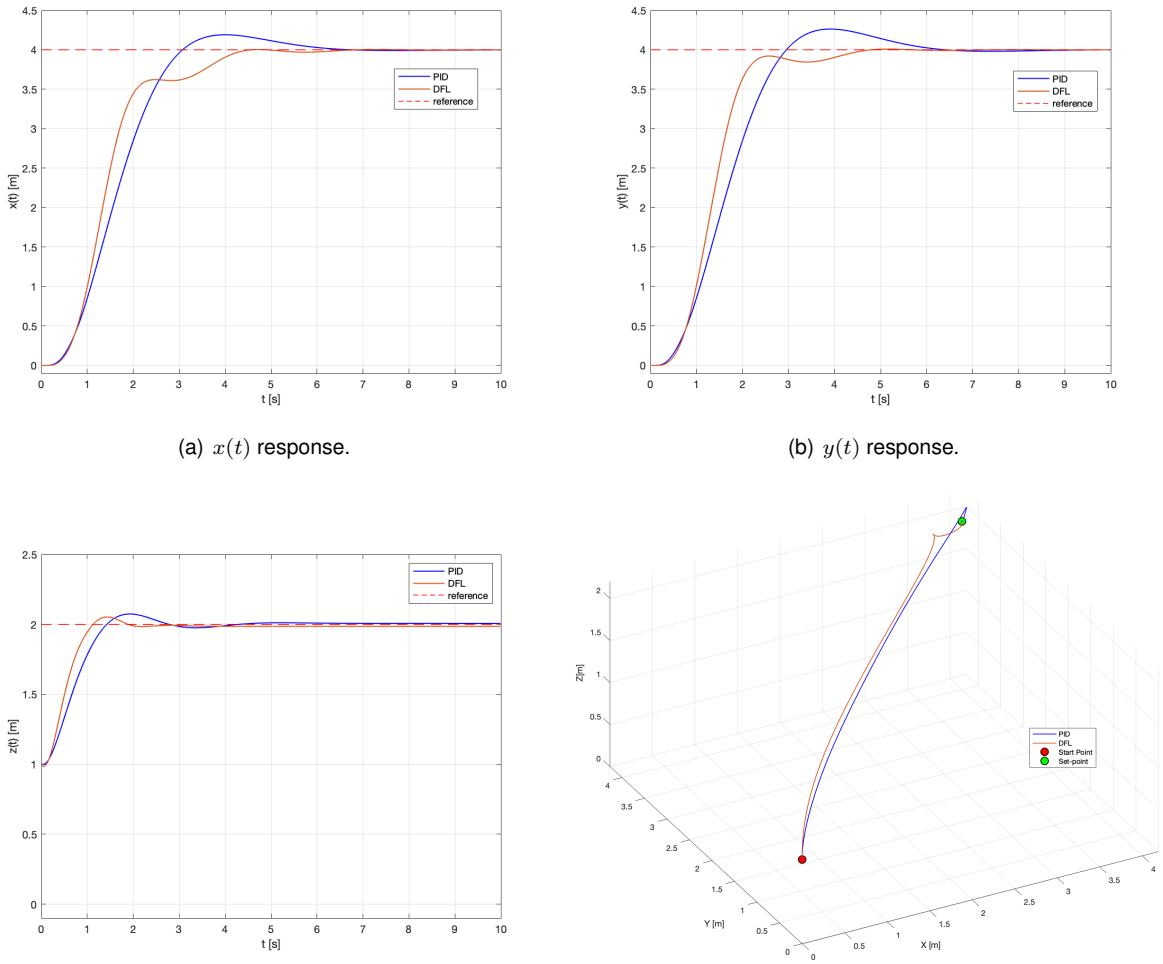
Figure 5.4: DFL Flight controller: Position and Yaw response to a step input.

	$x(t)$	$y(t)$	$z(t)$	$\psi(t)$
Rise Time [s]	1.14	1.19	0.68	0.58
Settling Time [s]	4.08	4.06	1.11	1.02
Overshoot	0.60 %	0.23 %	0.38 %	0.40 %

Table 5.2: Step response characteristics of DFL Flight controller.

5.2.3 Flight Waypoint Following

In order to compare the tracking performance between the two controllers in flight mode, the system was submitted to a simple waypoint test, in which it had to go from the initial pose $(x_0, y_0, z_0, \psi_0) = (0, 0, 1, 0)$ to setpoint $(x_1, y_1, z_1, \psi_1) = (4, 4, 2, 0)$, whose results are reported in figure 5.5.



(a) $x(t)$ response.

(b) $y(t)$ response.

(c) $z(t)$ response.

(d) 3D trajectory of robot.

Figure 5.5: Flight Test #3: Comparison between PID and DFL Flight controllers to waypoint following.

In both cases, as expected, the robot responds faster in the Z-direction and takes longer converging in the X-Y directions. Nevertheless, after a slight overshoot, the PID controller reaches the desired position in about 6 seconds, while the DFL controller takes approximately 4 seconds to arrive at the setpoint, presenting a minor drift before converging. Comparing the three-dimensional trajectories, present in

Figure 5.5(d), one can conclude both are very similar and successfully take the robot to the desired point. A video showing a side-by-side comparison of the controllers performance in Flight mode is available[1], where Test#3 is the equivalent to the situation in Figure 5.5.

5.3 Ground Mode Simulation

If we now turn to the case of ground locomotion, given that the actuators of the robot are meant for flying and not rolling, the response times will be slower to accommodate all the constraints that moving on a surface implies (like discussed in Section 3.4.2). Furthermore, since the thrust force cannot exceed the weight of the robot (condition 3.25), the rotational velocities of the motors are forced to be smaller than the equilibrium velocity, i.e,

$$\Omega_1,\ \Omega_2,\ \Omega_3,\ \Omega_4 < \Omega_e, \tag{5.1}$$

where Ω_e is given by equation (4.2), to avoid passing the take-off threshold.

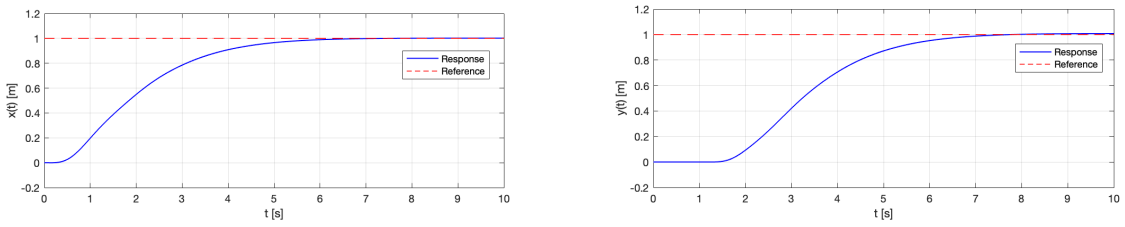
In this Section, the PID and DFL controllers will be compared through a series of different tests, analysing the attitude response and the controllers behaviour to particular situations, while inspecting the efficacy of the rolling controller when transversing in the plane.

5.3.1 PID Ground controller results

Figure 5.6 provides the experimental data of a step test performed on the X-Y position, where the response was used to tune the gains of the controller, adjusting the behaviour of the response to be more stable, avoiding oscillatory behaviour but increasing response times. The constant thrust input was chosen to be 60% of the lift-off force,

$$f_0 = 0.6mg,$$

which respects condition (3.25).



(a) $x(t)$ response to a step.

(b) $y(t)$ response to a step.

Figure 5.6: PID Ground controller: Position response to a step input.

Table 5.3 displays the response characteristics of the closed-loop system to the step input, highlighting the difference between the x and y settling time. These results suggest that the overshoot was

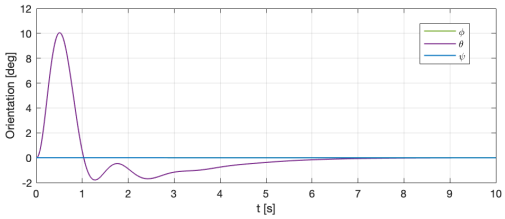
[1] `https://youtu.be/6COnBPgDS3M`

eliminated at the expense of the fastness of the response, in particular when the robot is not aligned with the desired setpoint.

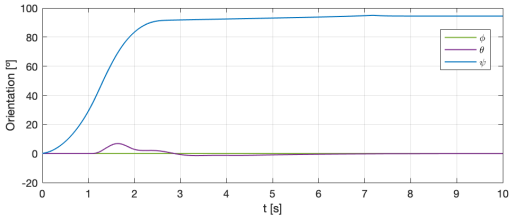
	$x(t)$	$y(t)$
Rise Time [s]	3.15	3.25
Settling Time [s]	5.50	6.70
Overshoot	0.17 %	0.77 %

Table 5.3: Step response characteristics of PID Ground controller.

Since the robot starts at $(x, y) = (0, 0)$, aligned with the X-axis ($\psi = 0$), a step signal on this direction will cause the vehicle to pitch and accelerate forward, pitching backwards when it gets closer to point $(x, y) = (1, 0)$, in order to decelerate and stop, as shown in Figure 5.7(a). On the other hand, a step signal on the Y-direction implicates a rotation around the Z-axis, to align the orientation of the wheels with the desired point, which increases the response time. Looking at Figure 5.7(b), it is evident that the vehicle immediately rotates around itself and only then starts pitching, aligned with the destination point. In both cases it can be verified that the controller successfully maintains the roll angle to zero, keeping the wheels always in contact with the ground.



(a) Attitude response to a step in x(t).



(b) Attitude response to a step in y(t).

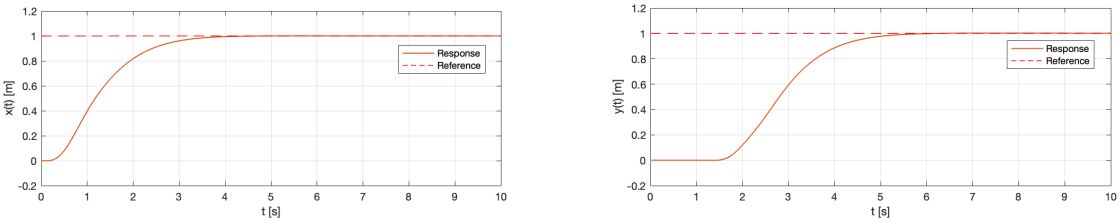
Figure 5.7: Attitude response of PID Ground controller to step inputs.

In comparison with the flight mode, we see an increase in the response time, with the yaw dynamics being much slower than in flight due to interaction with the ground. As for the the pitch dynamics, assuming there is no friction in the wheels joints, they are as fast as in flight mode, which results in very similar time responses in the longitudinal direction.

5.3.2 DFL Ground controller results

Proceeding with the DFL Ground controller step test, again it is possible to notice an increase in the closed-loop system performance when using the feedback linearising control, as displayed in Figure 5.8. Looking at Table 5.4, one can see that the trends are kept - the overshoot maintains values very close to zero and the settling time is bigger for the step on the Y-direction, given that the robot's motion implies turning and pitching, similarly to the PID controller.

Analysing the orientation of the vehicle during the step test, illustrated in Figure 5.9, it can be seen that the controller successfully maintains the wheels in contact with the ground, by keeping the roll angle at a constant zero value, while the pitch presents a smoother and faster response, but having a similar



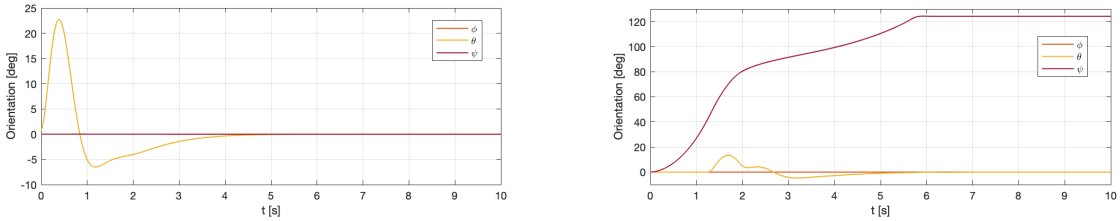
(a) $x(t)$ response to a step. (b) $y(t)$ response to a step.

Figure 5.8: DFL Ground controller: Position response to a step input.

	$x(t)$	$y(t)$
Rise Time [s]	1.88	2.14
Settling Time [s]	3.38	5.08
Overshoot	0.03 %	0.22 %

Table 5.4: Step response characteristics of DFL Ground controller.

behaviour as before. One of the most interesting differences between the PID and the DFL controller can be seen in Figure 5.9(b), where the robot starts accelerating (by pitching forward) before being aligned with the setpoint and continues turning along the trajectory until reaching its destination. An immediate consequence of this is that the final yaw angle will be greater than that of the PID controlled system.



(a) Attitude response to a step in x(t). (b) Attitude response to a step in y(t).

Figure 5.9: Attitude response of DFL Ground controller to step inputs.

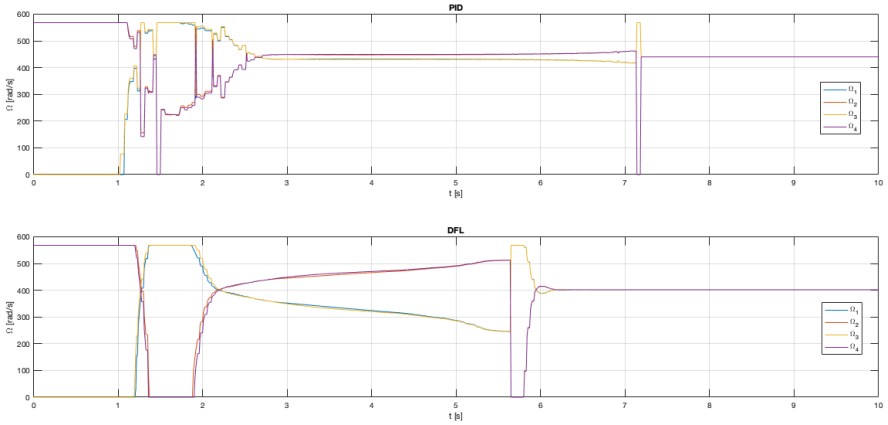


Figure 5.10: Comparison between PID and DFL motor velocities during a step in Y-direction.

Comparing the linear and the non-linear approach, we see an increase in responsiveness of about

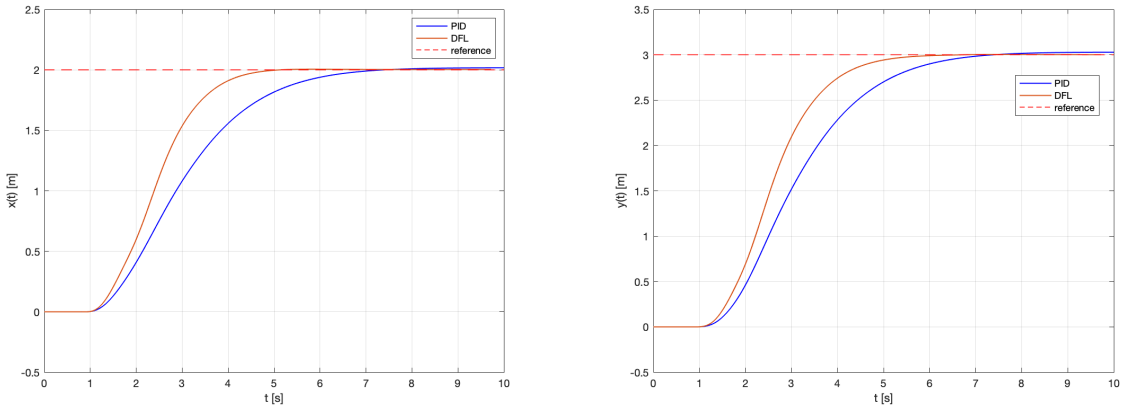
40% in the X-direction and 25% in the Y-direction. These discrepancy is due to the fact that the angular velocities of the rotors have a saturation value equal to Ω_e, like seen in 5.10, where for the first second, rotor 2 and 4 achieve the saturation value in both the PID and DFL cases, which implies having a maximum yaw angular acceleration that the system is not able to surpass.

5.3.3 Ground Waypoint Following

Intuitively, it is possible to affirm that the non-linear controller will have a greater impact in lengthier trajectories, since it can achieve higher velocities and therefore take less time in going from one point to another, whereas for small distances, the difference between the linear and non-linear control will be less meaningful. To show the distinctive behaviour of the tracking controllers in ground mode, the robot was submitted to two waypoint following tests[2]:

Test #1: $(x_0, y_0) = (0, 0) \longrightarrow (x_1, y_1) = (2, 3)$

Test #2: $(x_1, y_1) = (2, 3) \longrightarrow (x_2, y_2) = (-8, 8)$



(a) X-Direction.

(b) Y-Direction.

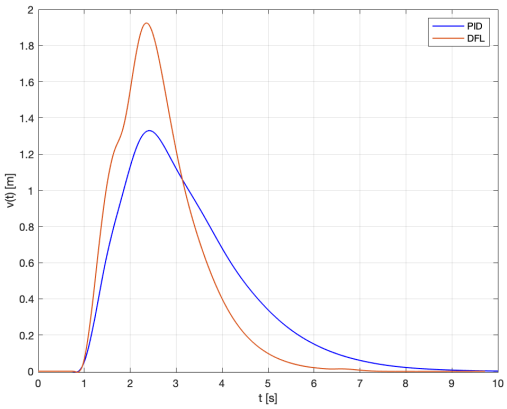
Figure 5.11: Ground Test #1: Comparison between position response of PID and DFL controllers.

In the first case, the distance the robot has to travel is smaller and as it can be seen from Figure 5.11, the DFL controller converges after 5 seconds while the PID controller converges after 7 seconds. In fact, by looking at Figure 5.12(a), we see that the non-linear controlled system is able to achieve a higher velocity and break in less time, since its pitch bandwidth is larger (Figure 5.12(b)), which allows bigger accelerations.

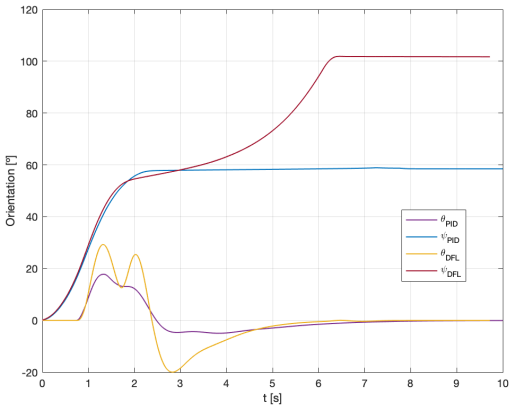
For the second waypoint, the robot has to travel a larger distance, and as it can be seen in Figure 5.14, this is where the DFL controller excels. After 6 seconds the DFL controlled system converges with the desired setpoint, while with the PID controller the vehicle takes about 10 seconds to arrive at its destination.

One of the most striking differences of test #2 lies within the decision to move forwards or backwards. The divergence between the two reactions is explained by considering the initial yaw orientation of the

[2] `https://youtu.be/eUILmzxTJdo`

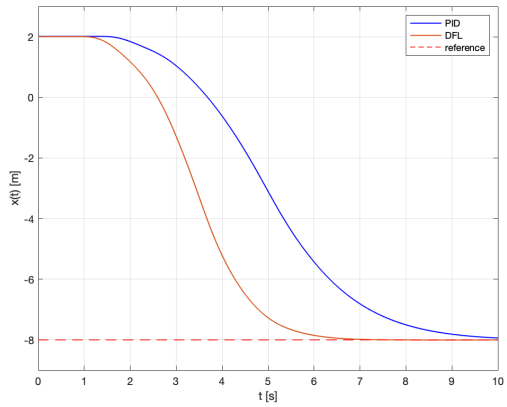


(a) Velocity.

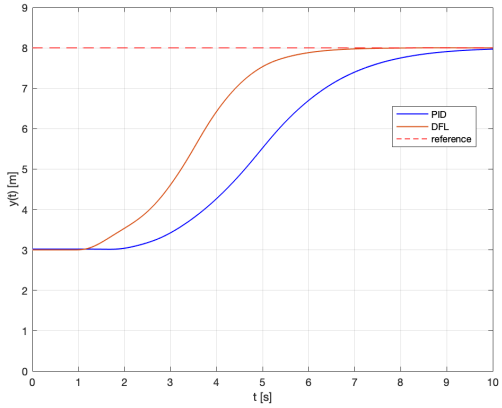


(b) Attitude.

Figure 5.12: Ground Test #1: Comparison between velocity and attitude response of PID and DFL controllers.



(a) X-Direction.



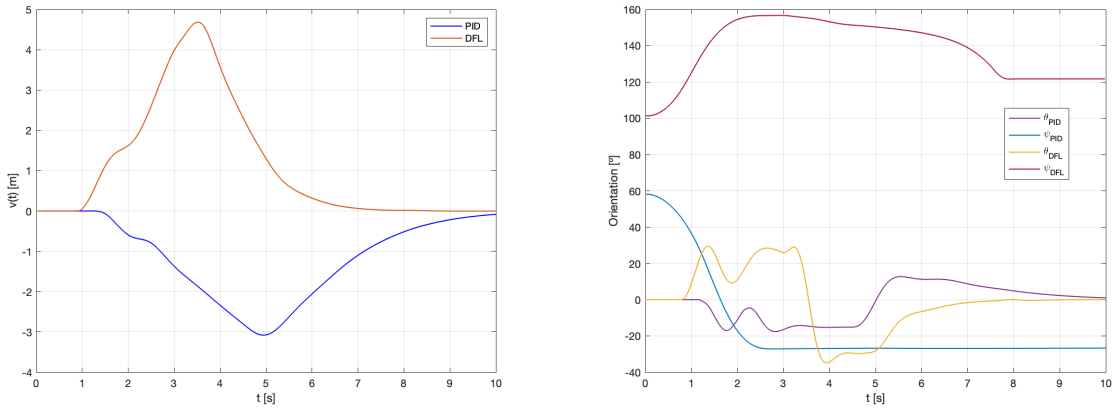
(b) Y-Direction.

Figure 5.13: Ground Test #2: Comparison between position response of PID and DFL controllers.

vehicle, as illustrated in Figure 5.14(b):

- In the PID case the robots initial orientation is 60º, which will render a negative error component on the x_r-direction, according to the rotated error definition (4.23). To avoid an unnecessary turn, the controller decides to rotate in the negative direction, by means of the component calculated according to equation (4.30), and pitches anti-clockwise, moving backwards (regarding the body frame).
- In the DFL scenario, the robots initial orientation in the plane is 100º, meaning it's faster to rotate clockwise, pitch in the positive direction and move forward.

The trajectories performed on the plane, as illustrated in Figure 5.15, are very similar and successfully take the robot from the initial position to the first and second waypoint. Performing the same tests in-flight, suggest that moving on the ground can be quite advantageous, since with 60% of the necessary thrust force the system has similar response times. For instance, to move from (0, 0, 1) to (2, 3, 1) (Test#1), the robot takes 6 seconds with the PID controller in Flight Mode and 7 seconds in Ground



(a) Longitudinal Velocity.

(b) Attitude Variables.

Figure 5.14: Ground Test #2: Comparison between velocity and attitude response of PID and DFL controllers.

Mode, which represents only 1 second in difference to perform the same path. In Ground Mode, the robot can save battery and even turn-off its rotors to execute inspection tasks, increasing the length of missions. Flight can be used to avoid obstacles and reach different structures.

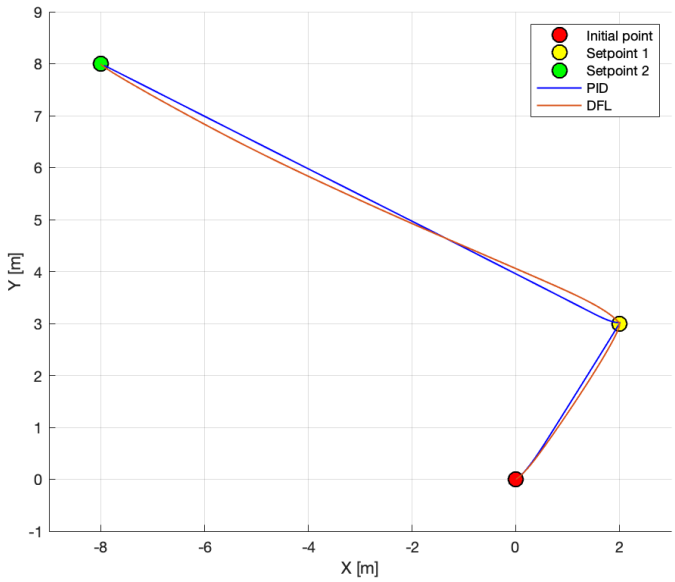


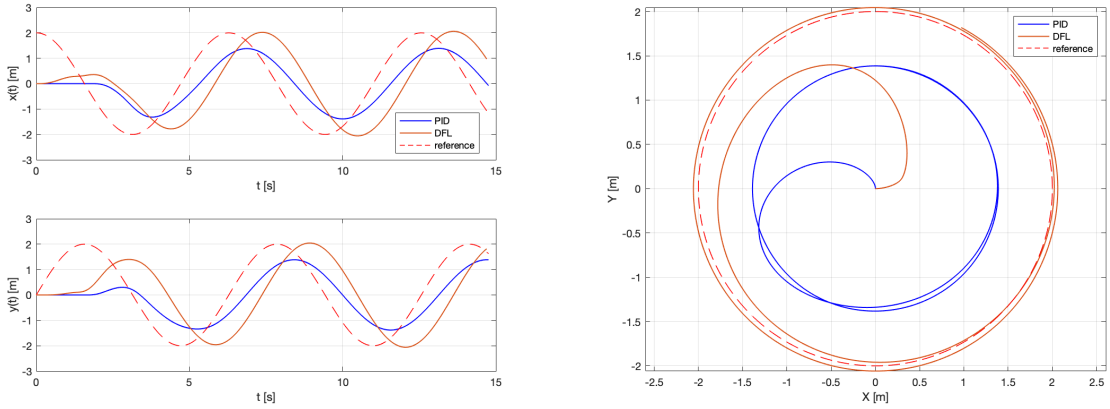
Figure 5.15: Waypoint Following in Ground mode.

5.3.4 Ground Trajectory Tracking

The robot was also subject to track trajectories that change over time. As an example, the system was submitted to tracking a circular trajectory defined as

$$\begin{cases} x_d(t) = 2\ cos(t) \\ y_d(t) = 2\ sin(t) \end{cases}, \tag{5.2}$$

which specifies a circle with a radius of 2 meters and a frequency of $1/(2\pi)\ Hz$. The response of the system can be analysed in Figure 5.16, where it is evident that the DFL controller is able to successfully converge with the specified trajectory, while the PID controller underperforms, since it is not fast enough to track the trajectory in time. A side-by-side comparison video is available[3].



(a) Response over time.

(b) 2D Trajectory

Figure 5.16: Trajectory Tracking in Ground Mode.

Nevertheless, this test assessed the controllers ability to eliminate the roll dynamics that appear with the coupling between the pitch and yaw attitude, keeping both the wheels on the ground while performing a translation and rotation on the ground.

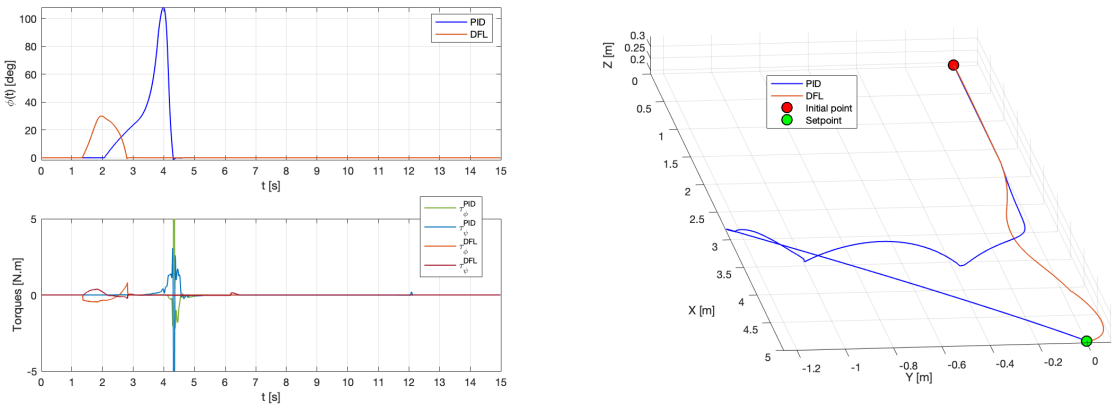
5.3.5 Ground Disturbance Rejection

Lastly, the Ground controllers were submitted to a disturbance while transversing the plane, that causes a lift on the left wheel. As depicted in Figure 5.17, we can see that using the DFL controller, the robot successfully recovers from the disturbance, alighting the left wheel back to the ground and converging with the setpoint, whereas the PID controller doesn't compensate for the disturbance and eventually rotates around itself before being able to return to its normal status, which can possibly damage the wheels of the robot. A video showing this situation is available[4].

This is because the DFL controller, as can be seen in Figure 5.17(a), immediately complies with the disturbance - as soon as the roll angle starts increasing, the input commanded torques τ_φ and τ_ψ are

[3] https://youtu.be/u5EpDoP-zEI
[4] https://youtu.be/SKFkECoF2MM

compensated and bring the roll angle back to zero. On the other hand, the PID controller enters in an unexpected state and is not able to eliminate the roll disturbance.



(a) Roll angle and Torques.

(b) 3D Trajectory.

Figure 5.17: Disturbance rejection in Ground Mode.

5.4 Inclined Mode Simulation

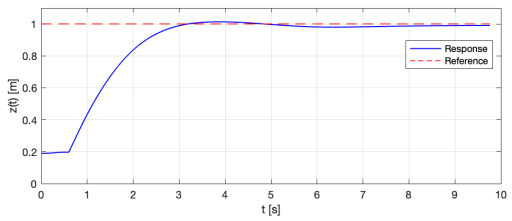
The third and final case is the case of going up and down an inclined surface, which will be controlled by a single degree-of-freedom, the altitude $z(t)$, meaning the user can input how high it wishes the robot to go on the slope.

From the three studied cases, this is by far the most specific one, given that prior knowledge about the surface is necessary, namely the inclination angle (γ) and its orientation in the world plane, i.e, the angular difference with the X-axis (ψ_0). The robot also has the ability to move on a surface forwards or backwards, meaning if a positive inclination is fed to the controller, the vehicle pitches in the positive direction while if a negative inclination is specified, the vehicle will pitch anti-clockwise and move backwards.

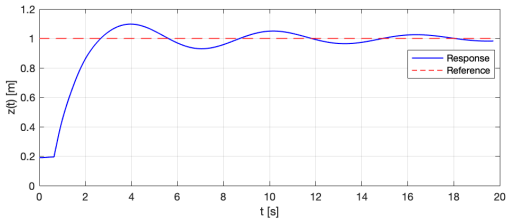
5.4.1 PID Inclined controller results

Firstly, to assess the behaviour of the closed-loop performance, a step test was performed in a surface with an inclination of 28º (0.5 rad), which was used to tune the parameters of the PID controller, to obtain the best response. Secondly, a step was applied in a slope with 46º (0.8 rad), to investigate the efficiency of the controller in different inclinations. Figure 5.18 compares the obtained results, where it can be seen the significant discrepancy between the tests.

As evidenced by Table 5.5, the response of the system was much better for the first slope, for which the controller was tuned. In fact, the controller underperforms when submitted to a slope with a different inclination, having an oscillatory behaviour and a large overshoot. Furthermore, the system takes about 18 seconds to fall within 2% of the reference value. This indicates a limitation of using the PID controller for inclined surfaces - the gains will have to be tuned differently for surfaces with different inclinations.



(a) Response to a step on a slope with $\gamma = 0.5$ (28º).



(b) Response to a step on a slope with $\gamma = 0.8$ (46º).

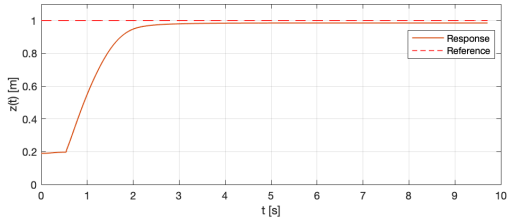
Figure 5.18: PID Inclined controller: Altitude response to a step input.

	$z(t)$, $\gamma = 0.5$	$z(t)$, $\gamma = 0.8$
Rise Time [s]	1.62	1.43
Settling Time [s]	4.52	18.44
Overshoot	2.32 %	11.68 %

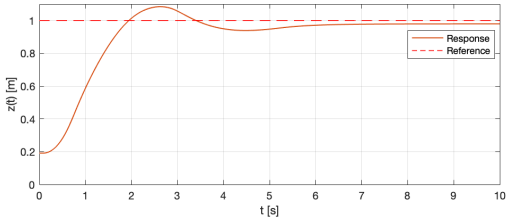
Table 5.5: Step response characteristics of PID Inclined controller.

5.4.2 DFL Inclined controller results

Considering now the results when employing the DFL controller in an inclined surface, as shown in Figure 5.19, one can detect the increased speed at which the system responds to the step input. Similarly as before, the controller gains were selected to have a good steady-state response in an incline with 28º (0.5 rad) and then submitted to a second test in a slope with 46º (0.8 rad) to assess its behaviour in a different scenario. The respective characteristics are reported in table 5.6.



(a) Response to a step on a slope with $\gamma = 0.5$ (28º).



(b) Response to a step on a slope with $\gamma = 0.8$ (46º).

Figure 5.19: DFL Inclined controller: Altitude response to a step input.

	$z(t)$, $\gamma = 0.5$	$z(t)$, $\gamma = 0.8$
Rise Time [s]	1.12	1.62
Settling Time [s]	2.34	4.52
Overshoot	0 %	2.32 %

Table 5.6: Step response characteristics of DFL Inclined controller.

Looking first at Figure 5.19(a), we can see that the DFL controller behaves much better than the previous approach, halving the settling time and eliminating the overshoot entirely, as evidenced in Table 5.6. Nevertheless, due to lack of integral action, there is a steady-state error of $e_z = 0.0155$ m.

Even more significant is the difference in behaviour between the PID and the DFL when submitted to a slope with a different inclination, as evidenced by comparing figures 5.18(b) and 5.19(b). While

the linear approach lacks the robustness to adapt to different inclinations without re-tuning the gains, the non-linear controller is capable of complying with different specifications and still have a fairly good response. Regardless, in comparison with Figure 5.19(a), it can be noted the decrease in the response time, which again suggests that in order to obtain the best reaction the controller should be tuned according to the specifications of the slope.

5.4.3 Inclined Surface Waypoint Following

Two further tests were performed on both slopes, to compare the controllers against each other when submitted to different waypoints in different inclinations:

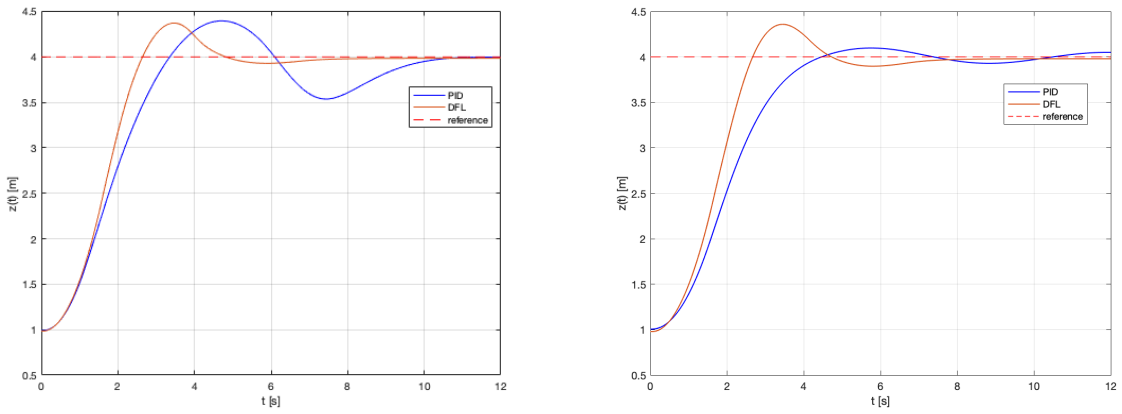
Test #1: $z_0 = 1 \longrightarrow z_1 = 4$ in a slope with $\gamma = 0.5$

Test #2: $z_1 = 4 \longrightarrow z_2 = 2$ in a slope with $\gamma = 0.5$

Test #3: $z_0 = 1 \longrightarrow z_1 = 4$ in a slope with $\gamma = 0.8$

Test #4: $z_1 = 4 \longrightarrow z_2 = 2$ in a slope with $\gamma = 0.8$

Test #1 and #3 compare how the PID and DFL controlled system climb both slopes, as portrayed in Figure 5.21. Overall, the non-linear controller (orange line) reaches the desired height faster, in both situations, while the linear controller (blue line) presents a big overshoot on the first case, only stabilising after 10 seconds, demonstrating again an oscillatory steady-state behaviour in the second situation, which is not desirable. Once more, it can be inferred that unless the PID controller is re-tuned for different inclinations, the DFL controller should be used instead.



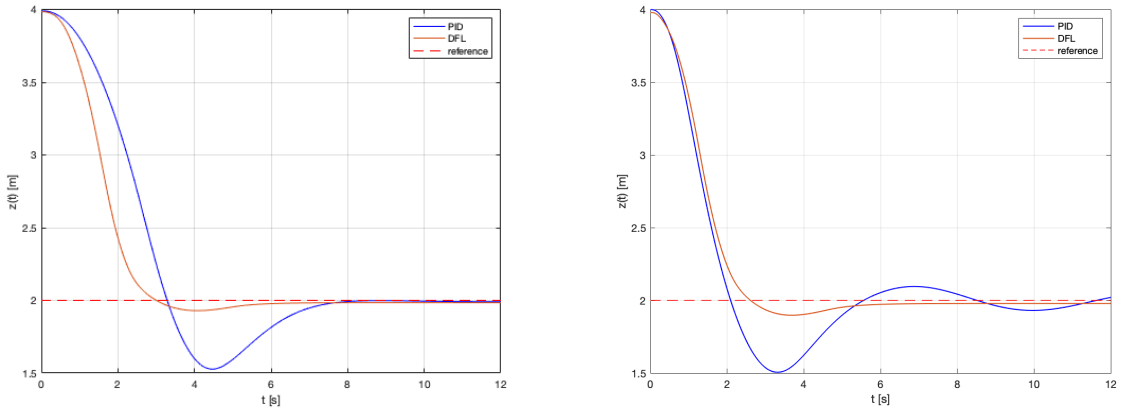
(a) Test #1 - $\gamma = 0.5$ (28º). (b) Test #3 - $\gamma = 0.8$ (46º).

Figure 5.20: Inclined Test #1 and #3: Comparison between position response of PID and DFL controllers.

On the other hand, tests #2 and #4 compare the robots descent in the same two slopes, as depicted in Figure 5.20. Once again, the DFL controller presents much better results, reaching the desired height in less time and breaking faster to avoid big overshoots, contrary to the PID controller.

Globally, when moving on an inclined surface, the feedback linearising controller behaves much better, with a stable response and a steady-state error inferior to 1cm, thus it will provide better results

for inspection applications, which need the robot to be stable. The PID controller can also be used, with the disadvantage of the gains having to be re-tuned for different situations, which might not be feasible for certain tasks. A video[5] of these tests is available for the reader.



(a) Test #2 - $\gamma = 0.5$ (28°).

(b) Test #4 - $\gamma = 0.8$ (46°).

Figure 5.21: Inclined Test #2 and #4: Comparison between position response of PID and DFL controllers.

5.4.4 Rotating on top of an Inclined Surface

Although both controllers assume that the robot can only move up and down without rotating, it was found that for small rotation angles the system is able to effectively comply without becoming unstable. This will, of course, depend on the friction properties of the surface and how much the robot rotates. The experiments performed suggested that the robot can perform small rotations up to 30 degrees on top of the inclined surface, i.e, $\psi - \psi_0 = \pm\pi/6$.

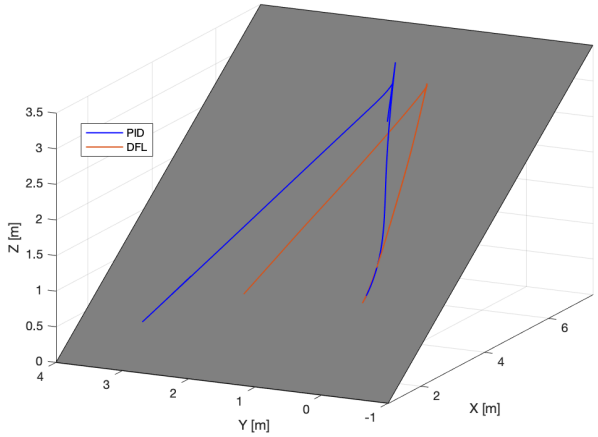


Figure 5.22: Rotating on top of an inclined surface experiment.

An example from such an experiment[6], where the robot rotates while climbing the slope and again when descending, can be seen in Figure 5.22. This simple experiment only intends to show that the

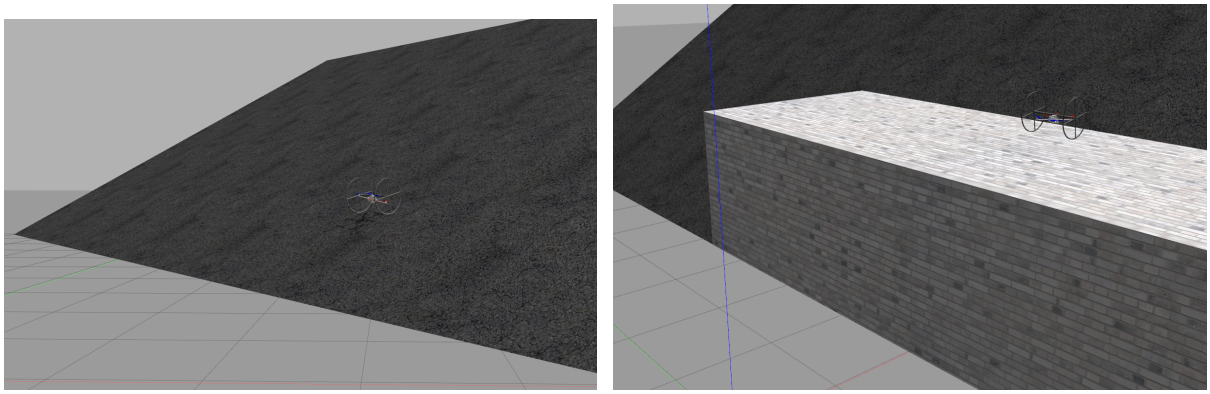
[5] https://youtu.be/yC0uWZ5nCdo
[6] https://youtu.be/XEc94118djg

Bimodal robot has the possibility of performing trajectories more complex than straight lines on top of an incline, as long as the thrust vector does not become perpendicular to the lateral weight component, like already discussed.

5.5 Hybrid Waypoint Navigation

The most interesting aspect of the Bimodal aerial robot developed in this work is undoubtedly the ability to move in the air, ground and within different types of surfaces. To perform micro-level inspection of structures, the robot must be able to effectively land and transition modes, continuing its movement on top of the surface. The controller has the ability to switch between modes, moving autonomously towards user defined waypoints, like described in Section 5.1.2.

Multiple experiments were performed to test complete missions consisting of all modes together, which showed the success of the system in navigating and switching between them. Two different examples are reported below, the first using the PID controller and the second employing the DFL controller. The simulation testbeds are shown in Figure 5.23, where we can see the robot on top of a flat and inclined surface.



(a) Testbed #1. (b) Testbed #2.

Figure 5.23: Gazebo Simulation Testbed.

Hybrid Test #1:

In the first scenario, there is a slope in the first quadrant of the plane with parameters $\zeta = (0.5, \frac{\pi}{4})$. The robot starts in the world frame origin and receives the following commands:

1. Move in **Ground Mode** to (3, -5, 0);
2. **Take-off** and move in **Flight Mode** to (4, 4, 2, $\frac{\pi}{4}$);
3. **Land**, detecting an inclined surface;
4. Climb the **Inclined Surface** to $z = 3$m;
5. **Take-off** and **Fly** to (0, 0, 4, $\frac{\pi}{4}$);

6. Descend to (0, 0, 1, 0) in **Flight Mode**;

7. **Land**, detecting a flat surface;

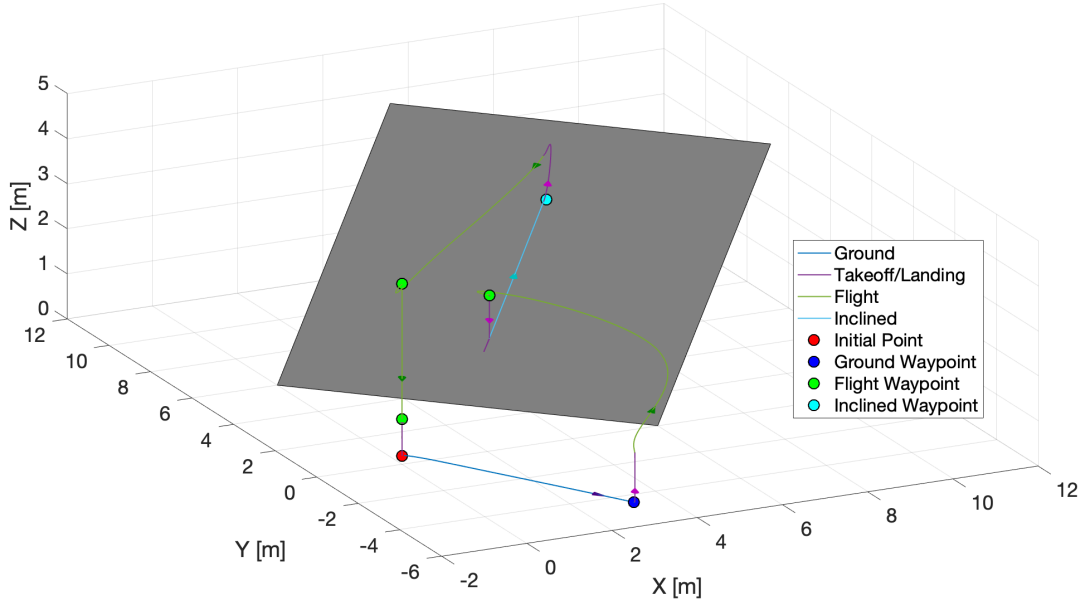


Figure 5.24: Hybrid Test #1 using PID: 3D trajectory of the robot.

The corresponding trajectory performed by the robot can be seen in Figure 5.24. The robot successfully lands on the inclined surface and automatically switches to the corresponding controller, predicting an inclination of $\gamma = 0.48$ rad, like reported in Figure 5.25, which is very close to the actual value of the ramp. A video portraying this experiment, using the PID[7] and also the DFL[8], is available for consultation.

Hybrid Test #2:

The second scenario portrays a flat and an inclined surface in which the robot needs to move. Starting in the origin it moves within the following set of coordinates:

1. **Take-off** and **Fly** to (2.5, 0, 3, 0);

2. **Land**, detecting a flat surface;

3. Move in **Ground Mode** to (4, 0, 2);

4. Climb the **Inclined Surface** to $z = 3$m;

[7] https://youtu.be/VEySHOihbwM
[8] https://youtu.be/vGfKHOEp5b8

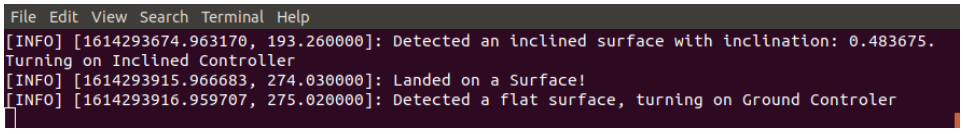


Figure 5.25: Command Line showing Land command response.

5. **Take-off** and **Fly** to (6, 5, 4, 0);
6. **Land**, detecting an inclined surface;
7. Descend to $z = 1$ in **Inclined Mode**;
8. Move in **Ground Mode** to (5, 1, 0) and then to (-6, 2, 0);
9. Return to the origin in **Ground Mode**;

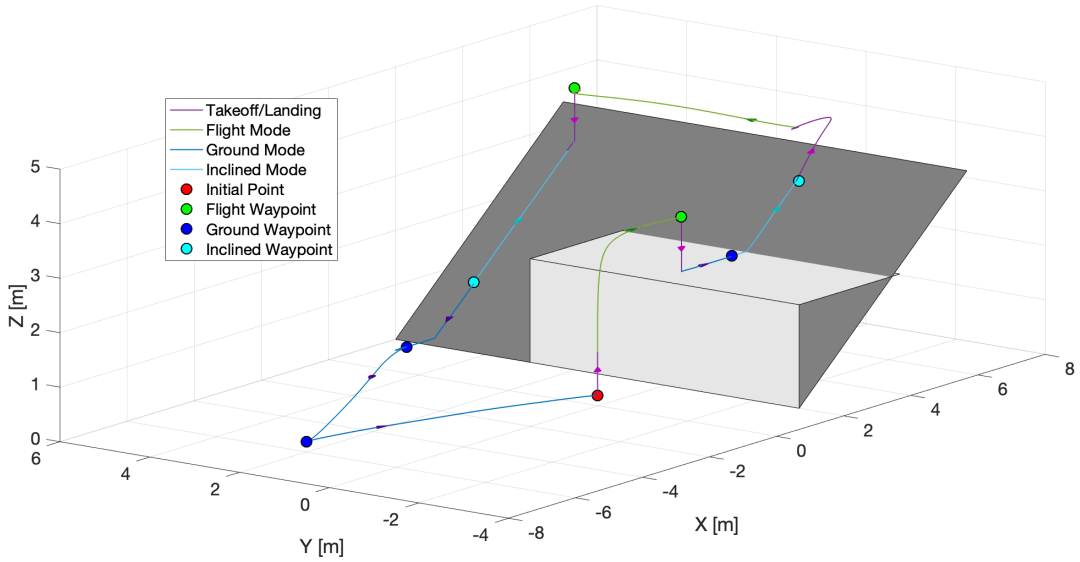


Figure 5.26: Hybrid Test #2 using DFL: 3D trajectory of the robot.

This case, besides showing once more two successful landings in a flat and an inclined surface, showed that it is also possible to transition from Ground to Inclined mode, between waypoints 3 and 4, and vice-versa, between waypoints 7 and 8. This experiment is also reported in video, for both the cases of PID[9] and DFL[10].

These experiments show the usefulness of the proposed system, which is able to interact with multiple surfaces, thus it can be of great interest to perform multiple inspection tasks. For example, the robot can move between structures to inspect, like solar panels, by rolling on the ground or flying between them if the terrain is rough or there are obstacles. It can then land on them, moving up and down to cover all the surface area, detecting any defects if they exist, all while using less energy and increasing stability when compared to in-flight inspection.

[9] https://youtu.be/oYF7YHTlCms
[10] https://youtu.be/VJQzz-BBlDo

Chapter 6

Conclusions

A Bimodal aerial robot capable of aerial and surface locomotion was described, allowing rapid micro-level inspection of flat and inclined surfaces. Consisting of a common quadrotor attached to two passive wheels, the system results in a mechanically simple and efficient solution that exploits the same actuation mechanism for all modes of operation. The mathematical models and two types of controllers were derived and tested for different locomotion modes and their performance was compared against each other. A simulation environment demonstrated the success of the motion control system of the robot and its applicability in full hybrid missions.

6.1 Achievements

An overview of the state-of-the-art solutions revealed that the use of unmanned vehicles for inspection purposes is a growing area of interest, however, there are few available solutions that allow the robot to be in contact with the structure to be analysed. This study set out to investigate the feasibility of adapting an aerial robot, such as a quadrotor, to move within different types of surfaces, without the addition of complex and heavy actuators. A robot design which uses the same four actuators for its multiple modes was proposed and prototyped, its mathematical model laid out and its fundamental control approach derived and successfully tested.

To control the robot in its different modes using the same actuation set, different conditions had to be imposed, since the vehicles primary function is to fly, namely constraining some degrees-of-freedom in surface locomotion. The dynamical model of the system was formulated by taking into account the specific circumstances of flying, moving on the ground or on inclined surfaces and two controllers were designed, the first by simplifying the model, which allowed the use of a linear approach, and the second making use of a non-linear approach.

Furthermore, a simulation environment was created, that allowed to test the aforementioned controllers and perform multiple experiments, comparing their performance. These tests provided evidence that the non-linear controller is more efficient, as was expected, and evaluated the effectiveness of a simple planner, that can be used for hybrid missions. Transitioning between states was also shown to

be successful, with the robot having the capability of approaching and landing in different surfaces, by switching controllers and stabilising after touch-down. The findings clearly indicate that this types of robots can be a very useful solution for inspection of industrial structures, which promises to decrease the associated O&M costs.

These framework can serve as a base for future developments, since it grants the possibility to create and control multiple Bimodal aerial robots, laying the groundwork for future research into state estimation, sensorization, path planning, or other higher-level tasks.

6.2 Limitations and Future Work

Finally, a number of important limitations need to be considered. First, although *Gazebo* is the reference simulator for roboticist's, modelling a great part of real-life effects, to effectively verify the robustness of the control methods, experimental tests with real-time applications should be addressed. Implementing the proposed controllers on the developed prototype (appendix A) should be the first priority for future work.

Secondly, the study did not evaluate the use of state estimation algorithms, assuming at all-times that the full vector of states is available for the controller. Further research could focus on developing a state estimator for the Bimodal robot, or employing one of the available estimators in *RotorS*, tailoring them for the three vehicle modes. Moreover, observability and controllability studies can be carried out in future, which would allow a better understanding of the underlying dynamics of the system. The PID control can also benefit from gain-tuning methodologies, to obtain better and stabler results and fairer comparisons with other control strategies.

More specifically, the performed experiments revealed that the inclined mode controllers behave differently for slopes with different inclinations, suggesting that a controller with adaptive gains could be used, to obtain the best response regardless of the inclination angle. This adaptive structure could also be used to estimate and adapt to uncertain and varying parameters, like the friction coefficients, which depends entirely on the interaction between the wheels and the surface. To roll without sliding, there must exist a static friction force on the wheels, which might not happen in extremely polished surfaces, meaning future wheels could be designed as to increase the friction between the surface and the wheel but decreasing it in the bearings. Nevertheless, it is suggested that the effects of friction on the system are further investigated in future studies.

Developing and testing surface inspection methods, by means of different sensors, are some of the areas of work to pursue. To explore the potential use of Bimodal robots for inspection of renewable energy plants, like wind turbines and solar panels, the robot should carry the correct sensors like thermal cameras, and have the capability to detect problems within this structures. In this scope, the Inclined Mode can also be improved to perform multiple types of trajectories, and move within round surfaces. Lastly, it would be interesting to develop computer vision algorithms, using a visual camera attached to the robot, for automatic detection of surfaces to inspect and obstacles to avoid.

Bibliography

[1] V. Kumar and N. Michael. Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11):1279–1291, Sept. 2012. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364912455954.

[2] M. Burri, J. Nikolic, C. Hurzeler, G. Caprari, and R. Siegwart. Aerial service robots for visual inspection of thermal power plant boiler systems. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 70–75, Zurich, Sept. 2012. IEEE. ISBN 978-1-4673-4587-3 978-1-4673-4585-9 978-1-4673-4586-6. doi: 10.1109/CARPI.2012.6473374.

[3] Y. Zefri, A. Elkcttani, I. Sebari, and S. A. Lamallam. Inspection of Photovoltaic Installations by Thermo-visual UAV Imagery Application Case: Morocco. In *2017 International Renewable and Sustainable Energy Conference (IRSEC)*, pages 1–6, Tangier, Dec. 2017. IEEE. ISBN 978-1-5386-2847-8. doi: 10.1109/IRSEC.2017.8477241.

[4] H. Kayan, R. Eslampanah, F. Yeganli, and M. Askar. Heat leakage detection and surveiallance using aerial thermography drone. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, Izmir, Turkey, May 2018. IEEE. ISBN 978-1-5386-1501-0. doi: 10.1109/SIU.2018.8404366.

[5] A. Ortiz, F. Bonnin-Pascual, and E. Garcia-Fidalgo. Vessel Inspection: A Micro-Aerial Vehicle-based Approach. *Journal of Intelligent & Robotic Systems*, 76(1):151–167, Sept. 2014. ISSN 0921-0296, 1573-0409. doi: 10.1007/s10846-013-9852-4.

[6] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse. State-of-the-art technologies for UAV inspections. *IET Radar, Sonar & Navigation*, 12:151–164, Feb. 2018. ISSN 1751-8792, 1751-8792.

[7] S. Omari, P. Gohl, M. Burri, M. Achtelik, and R. Siegwart. Visual industrial inspection using aerial robots. In *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*, pages 1–5, Foz do Iguassu, Brazil, Oct. 2014. IEEE. ISBN 978-1-4799-6422-2 978-1-4799-6420-8.

[8] Durable Project. URL `https://www.durableproject.eu`. Interreg Atlantic Area. [Online; Accessed: 20 May 2020].

[9] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26054-9. doi: 10.1007/978-3-319-26054-9_23. URL http://dx.doi.org/10.1007/978-3-319-26054-9_23.

[10] J. Quenzel, M. Nieuwenhuisen, D. Droeschel, M. Beul, S. Houben, and S. Behnke. Autonomous MAV-based Indoor Chimney Inspection with 3D Laser Localization and Textured Surface Reconstruction. *Journal of Intelligent & Robotic Systems*, 93(1-2):317–335, Feb. 2019. ISSN 0921-0296, 1573-0409. doi: 10.1007/s10846-018-0791-y.

[11] P. Addabbo, A. Angrisano, M. L. Bernardi, G. Gagliarde, A. Mennella, M. Nisi, and S. L. Ullo. UAV system for photovoltaic plant inspection. *IEEE Aerospace and Electronic Systems Magazine*, 33 (8):58–67, Aug. 2018. ISSN 0885-8985, 1557-959X.

[12] P. B. Quater, F. Grimaccia, S. Leva, M. Mussetta, and M. Aghaei. Light Unmanned Aerial Vehicles (UAVs) for Cooperative Inspection of PV Plants. *IEEE Journal of Photovoltaics*, 4(4):1107–1113, July 2014. ISSN 2156-3381, 2156-3403.

[13] M. Stokkeland, K. Klausen, and T. A. Johansen. Autonomous visual navigation of Unmanned Aerial Vehicle for wind turbine inspection. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 998–1007, Denver, CO, USA, June 2015. IEEE. ISBN 978-1-4799-6010-1.

[14] B. E. Schafer, D. Picchi, T. Engelhardt, and D. Abel. Multicopter unmanned aerial vehicle for automated inspection of wind turbines. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 244–249, Athens, Greece, June 2016. IEEE. ISBN 978-1-4673-8345-5.

[15] M. Fumagalli, R. Naldi, A. Macchelli, F. Forte, A. Q. Keemink, S. Stramigioli, R. Carloni, and L. Marconi. Developing an Aerial Manipulator Prototype: Physical Interaction with the Environment. *IEEE Robotics & Automation Magazine*, 21(3):41–50, Sept. 2014. ISSN 1070-9932. doi: 10.1109/MRA.2013.2287454.

[16] S. Hamaza, I. Georgilas, and T. Richardson. An Adaptive-Compliance Manipulator for Contact-Based Aerial Applications. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 730–735, Auckland, July 2018. IEEE. ISBN 978-1-5386-1854-7. doi: 10.1109/AIM.2018.8452382.

[17] S. Hamaza, I. Georgilas, and T. Richardson. Towards an Adaptive-Compliance Aerial Manipulator for Contact- Based Interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Madrid, Oct. 2018. IEEE. ISBN 978-1-5386-8094-0. doi: 10.1109/IROS.2018.8593576.

[18] T. Ikeda, S. Yasui, M. Fujihara, K. Ohara, S. Ashizawa, A. Ichikawa, A. Okino, T. Oomichi, and T. Fukuda. Wall contact by octo-rotor UAV with one DoF manipulator for bridge inspection. In *2017*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5122–5127, Vancouver, BC, Sept. 2017. IEEE. ISBN 978-1-5386-2682-5. doi: 10.1109/IROS.2017.8206398.

[19] A. Ollero, J. Cortes, A. Santamaria-Navarro, M. A. Trujillo Soto, R. Balachandran, J. Andrade-Cetto, A. Rodriguez, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J. R. Martinez-de Dios, and F. Pierri. The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance. *IEEE Robotics & Automation Magazine*, 25(4): 12–23, Dec. 2018. ISSN 1070-9932, 1558-223X. doi: 10.1109/MRA.2018.2852789.

[20] Rami Mattar and Remy Kalai. Development of a Wall-Sticking Drone for Non-Destructive Ultrasonic and Corrosion Testing. *Drones*, 2(1):8, Feb. 2018. ISSN 2504-446X. doi: 10.3390/drones2010008.

[21] B. B. Kocer, T. Tjahjowidodo, M. Pratama, and G. G. L. Seet. Inspection-while-flying: An autonomous contact-based nondestructive test using UAV-tools. *Automation in Construction*, 106: 102895, Oct. 2019. ISSN 09265805. doi: 10.1016/j.autcon.2019.102895.

[22] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, Jan. 2011. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-010-9205-0.

[23] K. Kawasaki, Y. Motegi, M. Zhao, K. Okada, and M. Inaba. Dual connected Bi-Copter with new wall trace locomotion feasibility that can fly at arbitrary tilt angle. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 524–531, Hamburg, Germany, Sept. 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353422.

[24] N. Takahashi, S. Yamashita, Y. Sato, Y. Kutsuna, and M. Yamada. All-round two-wheeled quadrotor helicopters with protect-frames for air–land–sea vehicle (controller design and automatic charging equipment). *Advanced Robotics*, 29(1):69–87, Jan. 2015. ISSN 0169-1864, 1568-5535. doi: 10.1080/01691864.2014.991754.

[25] A. Kalantari and M. Spenko. Design and experimental validation of HyTAQ, a Hybrid Terrestrial and Aerial Quadrotor. In *2013 IEEE International Conference on Robotics and Automation*, pages 4445–4450, Karlsruhe, Germany, May 2013. IEEE. ISBN 978-1-4673-5643-5 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6631208.

[26] A. Kalantari and M. Spenko. Modeling and Performance Assessment of the HyTAQ, a Hybrid Terrestrial/Aerial Quadrotor. *IEEE Transactions on Robotics*, 30(5):1278–1285, Oct. 2014. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2014.2337555.

[27] M. Yamada, M. Nakao, Y. Hada, and N. Sawasaki. Development and field test of novel two-wheeled UAV for bridge inspections. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1014–1021, Miami, FL, USA, June 2017. IEEE. ISBN 978-1-5090-4495-5. doi: 10.1109/ICUAS.2017.7991308.

[28] C. J. Dudley, A. C. Woods, and K. K. Leang. A micro spherical rolling and flying robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5869, Hamburg, Germany, Sept. 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7354210.

[29] K. Kawasaki, M. Zhao, K. Okada, and M. Inaba. MUWA: Multi-field universal wheel for air-land vehicle with quad variable-pitch propellers. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1880–1885, Tokyo, Nov. 2013. IEEE. ISBN 978-1-4673-6358-7 978-1-4673-6357-0. doi: 10.1109/IROS.2013.6696605.

[30] S. Mizutani, Y. Okada, C. J. Salaan, T. Ishii, K. Ohno, and S. Tadokoro. Proposal and experimental validation of a design strategy for a UAV with a passive rotating spherical shell. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1271–1278, Hamburg, Germany, Sept. 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353532.

[31] M. Ootsuka, C. Premachandra, and K. Kato. Development of an air-ground operational robot and its fundamental controlling approach. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 1470–1474, Kita-Kyushu, Japan, Dec. 2014. IEEE. ISBN 978-1-4799-5955-6. doi: 10.1109/SCIS-ISIS.2014.7044710.

[32] C. Premachandra, M. Otsuka, R. Gohara, T. Ninomiya, and K. Kato. A study on development of a hybrid aerial terrestrial robot system for avoiding ground obstacles by flight. *IEEE/CAA Journal of Automatica Sinica*, 6(1):327–336, Jan. 2019. ISSN 2329-9266, 2329-9274. doi: 10.1109/JAS.2018.7511258.

[33] A. Kossett, J. Purvey, and N. Papanikolopoulos. More than meets the eye: A hybrid-locomotion robot with rotary flight and wheel modes. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5653–5658, St. Louis, MO, USA, Oct. 2009. IEEE. ISBN 978-1-4244-3803-7. doi: 10.1109/IROS.2009.5354632.

[34] A. Kossett, R. D'Sa, J. Purvey, and N. Papanikolopoulos. Design of an improved land/air miniature robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 632–637, Anchorage, AK, May 2010. IEEE. ISBN 978-1-4244-5038-1. doi: 10.1109/ROBOT.2010.5509453.

[35] A. Kossett and N. Papanikolopoulos. A robust miniature robot design for land/air hybrid locomotion. In *2011 IEEE International Conference on Robotics and Automation*, pages 4595–4600, Shanghai, China, May 2011. IEEE. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5979845.

[36] L. Daler, J. Lecoeur, P. B. Hahlen, and D. Floreano. A flying robot with adaptive morphology for multi-modal locomotion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1361–1366, Tokyo, Nov. 2013. IEEE. ISBN 978-1-4673-6358-7 978-1-4673-6357-0. doi: 10.1109/IROS.2013.6696526.

[37] C. J. Pratt and K. K. Leang. Dynamic underactuated flying-walking (DUCK) robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3267–3274, Stockholm, Sweden, May 2016. IEEE. ISBN 978-1-4673-8026-3. doi: 10.1109/ICRA.2016.7487498.

[38] J. R. Page and P. E. I. Pounds. The Quadroller: Modeling of a UAV/UGV hybrid quadrotor. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4834–4841, Chicago, IL, USA, Sept. 2014. IEEE. ISBN 978-1-4799-6934-0 978-1-4799-6931-9. doi: 10.1109/IROS.2014.6943249.

[39] M. T. Pope, C. W. Kimes, H. Jiang, E. W. Hawkes, M. A. Estrada, C. F. Kerst, W. R. T. Roderick, A. K. Han, D. L. Christensen, and M. R. Cutkosky. A Multimodal Robot for Perching and Climbing on Vertical Outdoor Surfaces. *IEEE Transactions on Robotics*, 33(1):38–48, Feb. 2017. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2016.2623346.

[40] H. Tsukagoshi, M. Watanabe, T. Hamada, D. Ashlih, and R. Iizuka. Aerial manipulator with perching and door-opening capability. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4663–4668, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139845.

[41] A. Kalantari, K. Mahajan, D. Ruffatto, and M. Spenko. Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4669–4674, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139846.

[42] W. Myeong and H. Myung. Development of a Wall-Climbing Drone Capable of Vertical Soft Landing Using a Tilt-Rotor Mechanism. *IEEE Access*, 7:4868–4879, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2889686.

[43] C. E. Doyle, J. J. Bird, T. A. Isom, J. C. Kallman, D. F. Bareiss, D. J. Dunlop, R. J. King, J. J. Abbott, and M. A. Minor. An Avian-Inspired Passive Mechanism for Quadrotor Perching. *IEEE/ASME Transactions on Mechatronics*, 18(2):506–517, Apr. 2013. ISSN 1083-4435, 1941-014X. doi: 10.1109/TMECH.2012.2211081.

[44] Team CoSTAR Nebula Autonomy Solution. URL `https://costar.jpl.nasa.gov`. [Online; Accessed: 18 November 2020].

[45] A. Kalantari, T. Touma, L. Kim, R. Jitosho, K. Strickland, B. T. Lopez, and A.-A. Agha-Mohammadi. Drivocopter: A concept Hybrid Aerial/Ground vehicle for long-endurance mobility. In *2020 IEEE Aerospace Conference*, pages 1–10, Big Sky, MT, USA, Mar. 2020. IEEE. ISBN 978-1-72812-734-7. doi: 10.1109/AERO47225.2020.9172782.

[46] D. D. Fan, R. Thakker, T. Bartlett, M. B. Miled, L. Kim, E. Theodorou, and A.-a. Agha-mohammadi. Autonomous Hybrid Ground/Aerial Mobility in Unknown Environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3070–3077, Macau, China, Nov. 2019. IEEE. ISBN 978-1-72814-004-9. doi: 10.1109/IROS40897.2019.8968276.

[47] M.-D. Hua, T. Hamel, P. Morin, and C. Samson. Introduction to Feedback Control of Underactuated VTOL Vehicles. *IEEE Control Systems Magazine*, 33(1):61–75, Feb. 2013.

[48] C. Luis and J. L. Ny. Design of a Trajectory Tracking Controller for a Nanoquadcopter. *arXiv:1608.05786 [cs.SY]*, Aug. 2016.

[49] P. I. Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*. Number v. 73 in Springer tracts in advanced robotics. Springer, Berlin, 2011. ISBN 978-3-642-20143-1 978-3-642-20144-8. OCLC: ocn754988355.

[50] D. W. Mellinger. *Trajectory Generation and Control for Quadrotors*. PhD thesis, University of Pennsylvania, 2012. URL `https://repository.upenn.edu/edissertations/547`.

[51] C. K. Ashis and K. Rahul Sharma. Dynamic Modeling and Altitude Control of Parrot Rolling Spider using LQR. In *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 1377–1381, Kannur,Kerala, India, July 2019. IEEE. ISBN 978-1-72810-282-5 978-1-72810-283-2.

[52] O. Araar and N. Aouf. Full linear control of a quadrotor UAV, LQ vs H∞. In *2014 UKACC International Conference on Control (CONTROL)*, pages 133–138, Loughborough, UK, July 2014. IEEE. ISBN 978-1-4799-5011-9.

[53] S. Formentin and M. Lovera. Flatness-based control of a quadrotor helicopter via feedforward linearization. In *IEEE Conference on Decision and Control and European Control Conference*, pages 6171–6176, Orlando, FL, USA, Dec. 2011. IEEE. ISBN 978-1-61284-801-3 978-1-61284-800-6 978-1-4673-0457-3 978-1-61284-799-3.

[54] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, San Diego, CA, USA, Oct. 2007. IEEE. ISBN 978-1-4244-0911-2 978-1-4244-0912-9.

[55] T. Madani and A. Benallegue. Backstepping Control for a Quadrotor Helicopter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260, Beijing, China, Oct. 2006. IEEE. ISBN 978-1-4244-0258-8 978-1-4244-0259-5.

[56] A. Mokhtari, N. K. M'Sirdi, K. Meghriche, and A. Belaidi. Feedback linearization and linear observer for a quadrotor unmanned aerial vehicle. *Advanced Robotics*, 20(1):71–91, Jan. 2006. ISSN 0169-1864, 1568-5535.

[57] S. A. Al-Hiddabi. Quadrotor control using feedback linearization with dynamic extension. In *2009 6th International Symposium on Mechatronics and its Applications*, pages 1–3, Sharjah, United Arab Emirates, Mar. 2009. IEEE. ISBN 978-1-4244-3480-0.

[58] F. Lewis, A. Das, and K. Subbarao. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET Control Theory & Applications*, 3(3):303–314, Mar. 2009. ISSN 1751-8644, 1751-8652.

[59] Sabatino, Francesco. Quadrotor Control: modeling, nonlinear control design and simulation. Master's thesis, KTH Electrical Engineering, Sweden, 2015.

[60] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Atlanta, GA, Dec. 2010. IEEE. ISBN 978-1-4244-7745-6 978-1-4244-7746-3.

[61] T. Lee, M. Leok, and N. H. McClamroch. Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3). *arXiv:1003.2005 [cs, math]*, Sept. 2011.

[62] B. Siciliano, L. Villani, L. Sciavicco, and G. Oriolo. *Robotics: modelling, planning and control*. Advanced textbooks in control and signal processing. Springer, London, 2009. ISBN 978-1-84628-641-4 978-1-84628-642-1.

[63] A. De Luca, G. Oriolo, and M. Vendittelli. Control of Wheeled Mobile Robots: An Experimental Overview. In M. Thoma, M. Morari, S. Nicosia, B. Siciliano, A. Bicchi, and P. Valigi, editors, *Ramsete*, volume 270, pages 181–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-42090-3 978-3-540-45000-9. doi: 10.1007/3-540-45000-9_8. Series Title: Lecture Notes in Control and Information Sciences.

[64] C. P. Tang. Differential flatness-based kinematic and dynamic control of a differentially driven wheeled mobile robot. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2267–2272, Guilin, China, Dec. 2009. IEEE. ISBN 978-1-4244-4774-9.

[65] D. V. Dimarogonas, Y. Karayiannidis, and M. Zambelli. Posture regulation for unicycle-like robots with prescribed performance guarantees. *IET Control Theory & Applications*, 9(2):192–202, Jan. 2015. ISSN 1751-8644, 1751-8652.

[66] R. Mahony, V. Kumar, and P. Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, Sept. 2012. ISSN 1070-9932. doi: 10.1109/MRA.2012.2206474.

[67] A. D. E. Luca and M. D. D. Benedetto. CONTROL OF NONHOLONOMIC SYSTEMS VIA DYNAMIC COMPENSATION. *Kybernetika*, 29(6):593–608, 1993.

[68] J.-J. E. Slotine and W. Li. *Applied nonlinear control*. Prentice Hall, Englewood Cliffs, N.J, 1991. ISBN 978-0-13-040890-7.

[69] A. Reyes-Lúa and S. Skogestad. Multiple-Input Single-Output Control for Extending the Steady-State Operating Range—Use of Controllers with Different Setpoints. *Processes*, 7(12):941, Dec. 2019. ISSN 2227-9717.

[70] Robot Operating System (ROS). URL `https://www.ros.org`.

[71] Gazebo: Robot simulation made easy. URL `http://gazebosim.org`.

[72] W. Giernacki, M. Skwierczynski, W. Witwicki, P. Wronski, and P. Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42, Miedzyzdroje, Poland, Aug. 2017. IEEE. ISBN 978-1-5386-2402-9. doi: 10.1109/MMAR.2017.8046794.

[73] Crazyflie2.0 Datasheet, . URL https://www.bitcraze.io/documentation/hardware/crazyflie_2_0/crazyflie_2_0-datasheet.pdf. Bitcraze. [Online; Accessed: 27 July 2020].

[74] Shapr3D. URL https://www.shapr3d.com.

[75] BLOCKS One MKII. URL http://www.blockstec.com/onemkii.html. Online; Accessed: 7 March 2021.

[76] Filament Properties Table. URL https://www.simplify3d.com/support/materials-guide/properties-table/. [Online; Accessed: 27 July 2020].

[77] Detailed description of the hardware, . URL https://wiki.bitcraze.io/projects:crazyflie2:userguide:index. [Online; Accessed: 27 July 2020].

[78] J. Forster. System identification of the crazyflie2.0 nano quadrocopter, Aug. 2015.

[79] List of moments of inertia, . URL https://en.wikipedia.org/wiki/List_of_moments_of_inertia. [Online; Accessed: 27 July 2020].

[80] Parallel axis theorem, . URL https://en.wikipedia.org/wiki/Parallel_axis_theorem. [Online; Accessed: 27 July 2020].

Appendix A

Prototype of the Robot

A small prototype of the Bimodal robot was created, to perform minor experiments using teleoperation, which can be used for future research and to test different solutions. This is a small step towards the bigger goal of creating a fully-functional autonomous Bimodal robot that is capable of performing inspection tasks. In this appendix the reader will be guided through the process of assembling the prototype for the robot, justifying the decisions made and emphasising the methods utilised.

A.1 Methods and Equipments

The mock-up of the robot was built by attaching an axle with two removable wheels to an off-the-shelf quadrotor. The chosen platform to be modified was the *Crazyflie2.0*, since it's a low-cost, expandable and upgradable nano-Quadcopter [72], that fits the purpose of developing and demonstrating the potential of adapting a UAV for surface locomotion, which can be scaled to a bigger and robust platform.

The wheels were designed following the constraints imposed by the *Crazyflie2.0* specifications, which were found by consulting the Datasheet available in *Bitcraze* website [73]. The wheels and the axle were modelled using *Shapr3D* [74], which is a CAD software powered by Siemens Parasolid engine, and printed using the BLOCKS One MKII 3D printer [75] available at ISR.

Considering this, the chosen material had to be printable, lightweight and resistant at the same time. Comparing different filaments, PLA (PoliLactic Acid) plastic was chosen, considering it's a strong, light (low density) and stiff material [76].

A.2 Mechanical Specifications

According to the *Crazyflie2.0* datasheet [73], the length of the quadcopter from one motor shaft to the other is 92mm, while the maximum recommended payload is 15g, which means the wheels need to be extremely lightweight. Although there is no information about the rotor dimensions in the datasheet, in [72] it is said they are 45mm diameter plastic propellers.

The distance from the center of the rotors to both the x and y axis is

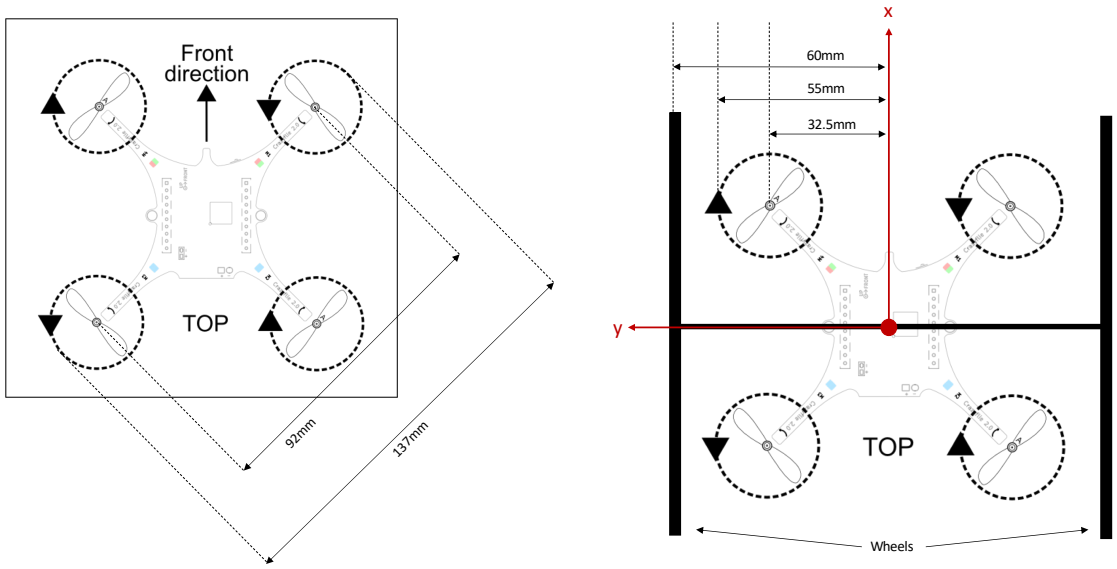
$$d_{rotor} = \frac{92}{2} \times cos(45°) = 32.5\text{mm},$$

meaning when the rotors are functioning, the minimum distance from each wheel to the origin (centre of mass of robot) that ensures the propellers are not damaged is

$$d_{wheel}^{min} = 32.5 + \frac{45}{2} = 55\text{mm}.$$

To ensure maximum safety when the robot is operating, like Figure A.1(b) shows, the distance between the centre of mass and the wheels is,

$$d_{wheel} = 60\text{mm}$$



(a) *Crazyflie2.0* dimensions

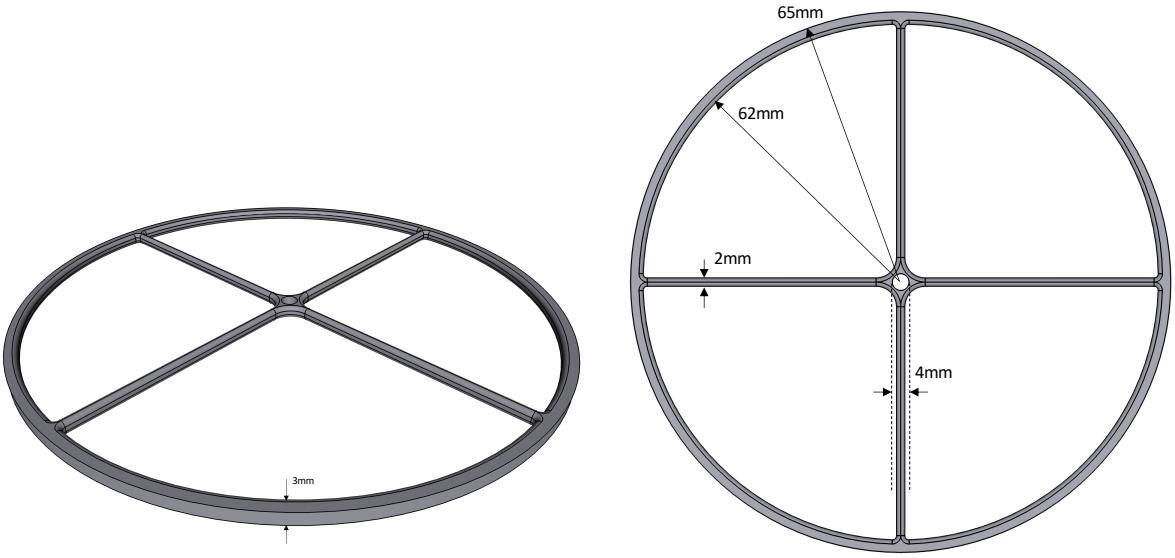
(b) Proposed wheels position

Figure A.1: Dimensions of original and modified Crazyflie2.0, adapted from [77].

A.3 Wheel and Axle Design

In figure A.2(a) a render of the wheel can be observed, while in Figure A.2(b) the corresponding dimensions are presented. The outer radius of 65mm was chosen to ensure the propellers are not damaged even if the robot is perpendicular to the surface where it is moving and the wheel only has four 2mm spokes to create minimal drag forces when operating normally in the air. The thickness and width of the wheel was set to be 3mm, to comply with weight constraints but certifying stability and shock-resistance.

The axle function is to unite both wheels with the quadrotor and has to allow free rotation of the wheels. *Crazyflie2.0* has two 2.1mm mounting holes separated by 30mm, which can be used for mechanically mounting something to the platform [77], as can be seen by the mechanical drawings which can be consulted in the datasheet [73]. These were used to attach the axle+wheels system to the



(a) 3D render (b) Wheel Dimensions

Figure A.2: CAD model of the wheel.

quadrotor, like observed in Figure A.3.

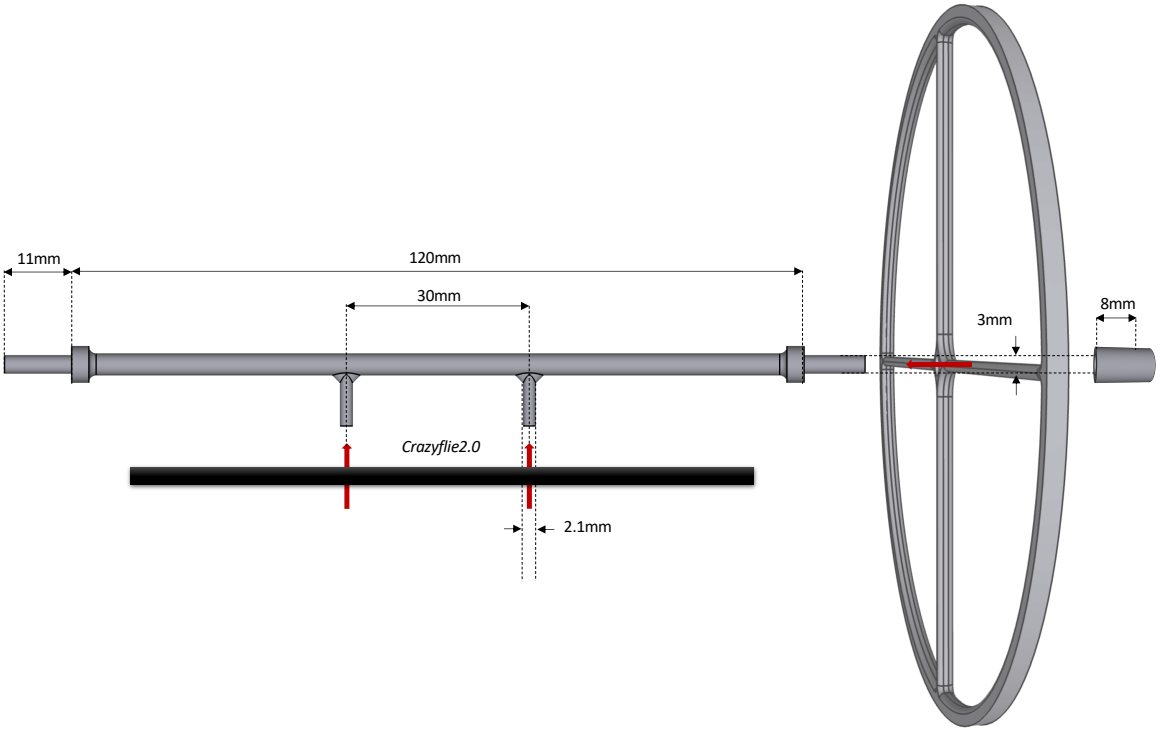


Figure A.3: Axle system and dimensions.

After being printed, the wheels and the axle were weighed giving a final mass of the wheel system of 12.9g, which does not surpass the payload of the system. The calculated moments of inertia of the complete system are reported in table A.1. The necessary values for the principal moments of inertia of the *Crazyflie2.0* and the motor coefficients were taken from [78].

	$Mass(g)$	$I_x(Kg.m^2)$	$I_y(Kg.m^2)$	$I_z(Kg.m^2)$
Axle	1.9	2.2800×10^{-6}	–	2.2800×10^{-6}
Wheel	5.5	3.0899×10^{-5}	2.2190×10^{-5}	3.0899×10^{-5}
Axle+Wheels	12.9	6.4078×10^{-5}	4.4200×10^{-5}	6.4078×10^{-5}
Crazyflie2.0	27.0	1.6572×10^{-5}	1.6656×10^{-5}	2.9262×10^{-5}
Crazywheel	39.9	9.3340×10^{-5}	1.6656×10^{-5}	9.3340×10^{-5}

Table A.1: Mass and Inertias of Crazywheel.

A.4 Assembly and Teleoperation

The assembled prototype, which was called the *Crazywheel* robot (two-wheeled *Crazyflie2.0*) can be seen in Figure A.4. Some experiments, using manual control, were performed in flight, ground and different surfaces. In Figure A.5 a snapshot of two experiments is shown, one rolling on the ground and the other climbing a slope.

Figure A.4: Two-wheeled *Crazyflie2.0* - Crazywheel.

(a) Teleoperation in Ground Mode.

(b) Teleoperation in an Inclined Surface.

Figure A.5: Teleoperation experiments, using the prototype.

The main conclusions driven from these simple experiments were that friction between the wheels

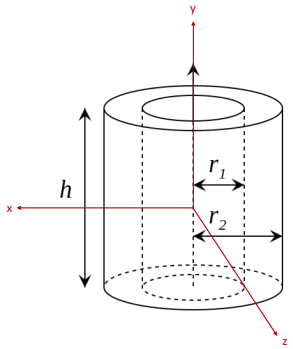
and the surface is fundamental for the correct behaviour of the robot, since the wheels might start sliding. It was verified that lack of friction (for example in polished surfaces), results in the Bimodal robot behaving erratically. Clearly, both the material of the wheels and of the surface where the vehicle moves, will influence the friction forces that arise and consequently the performance of the system, which is a subject that requires further research.

Appendix B

Moments of Inertia

The formulas to calculate the moments of inertia deemed necessary are listed below, according to [79].

B.1 Hollow Cylinder

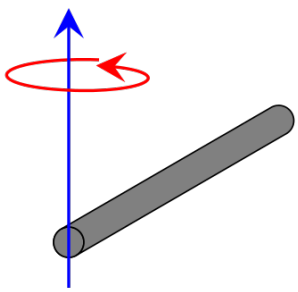


$$I_x = I_z = \frac{1}{12} m[3(r_1^2 + r_2^2) + h^2] \tag{B.1}$$

$$I_y = \frac{1}{2} m(r_1^2 + r_2^2) \tag{B.2}$$

Figure B.1: Moments of inertia of hollow cylinder, w.r.t center of mass reference frame.

B.2 Thin Rod



$$I_{rod}^{end} = \frac{1}{3} mL^2 \tag{B.3}$$

$$I_{rod}^{center} = \frac{1}{12} mL^2 \tag{B.4}$$

Figure B.2: Moments of inertia of thin rod, rotating about its centre and end.

B.3 Parallel Axis Theorem

Given the moment of inertia w.r.t to an axis passing through the body centre of mass, I_{CM}, the moment of inertia w.r.t to a new axis, parallel to the first and displaced by a distance d, is given by [80]:

$$I = I_{CM} + md^2 \tag{B.5}$$

where m is the mass of the body.

Appendix C

Simulation Gains and Parameters

This appendix contains the tuned gains and parameters used in the simulation of the *Wheelbird* robot.

C.1 PID Controller Gains

Mode	Gains	x	y	z	v	φ	θ	ψ
Flight	Proportional	1.0	1.0	4.5	–	56.0	56.0	7.3
	Derivative	1.2	1.2	2.6	–	12.0	12.0	5.2
	Integral	–	–	0.25	–	–	–	–
Ground	Proportional	0.48	–	–	3.8	–	15.6	5.5
	Derivative	–	–	–	–	–	7.3	5.2
	Integral	–	–	–	0.1	–	0.4	–
Inclined	Proportional	–	–	0.55	3.5	–	15.6	5.5
	Derivative	–	–	–	0.1	–	7.3	5.2
	Integral	–	–	0.8	0.6	–	0.4	–

Table C.1: PID Controller gains for all three modes.

C.2 DFL Controller Gains

The tracking component was used with the following gains:

Mode	Gains	z	φ	θ	ψ
Flight	k^0	8.5	22	22	15
	k^1	4.4	8	8	7.5
Surface	k^0	–	19	19	14.6
	k^1	–	10	10	7.5

Table C.2: DFL gains for flight and surface locomotion.

The position stabilisation block uses the gains reported in Table C.3.

Mode	Gains	x	y	z	v	Acceleration Valve	θ Valve
Flight	Proportional	1.7	1.7	–	–	–	–
	Derivative	2.05	2.05	–	–	–	–
Surface	Flat	0.7	–	–	4.8	0.8	0.8
	Inclined	–	–	1.1	5.5	1.5	0.8

Table C.3: DFL controller position stabilisation gains.

C.3 Configuration parameters

The parameters used in the simulation of the *Wheelbird* robot are the same found in *RotorS* for the *Hummingbird* with the extra mass and inertias the wheels represent in the robot:

- Mass: $1.126\ kg$
- Inertia: $diag(0.039, 0.007, 0.044)\ [kg.m^2]$
- Arm length: $0.17m$
- Rotor force constant: $8.54858 \times 10^{-6}\ kg.m.rad^{-2}$
- Rotor moment constant: $1.6 \times 10^{-2}\ kg.m^2.rad^{-2}$