HAL
open science

Robust tracking of dynamic targets with aerial vehicles using quaternion-based techniques

Hernán Abaunza Gonzalez

To cite this version:

Hernán Abaunza Gonzalez. Robust tracking of dynamic targets with aerial vehicles using quaternion-based techniques. Automatic. Université de Technologie de Compiègne, 2019. English. NNT : 2019COMP2480 . tel-02155857

HAL Id: tel-02155857

https://theses.hal.science/tel-02155857v1

Submitted on 14 Jun 2019

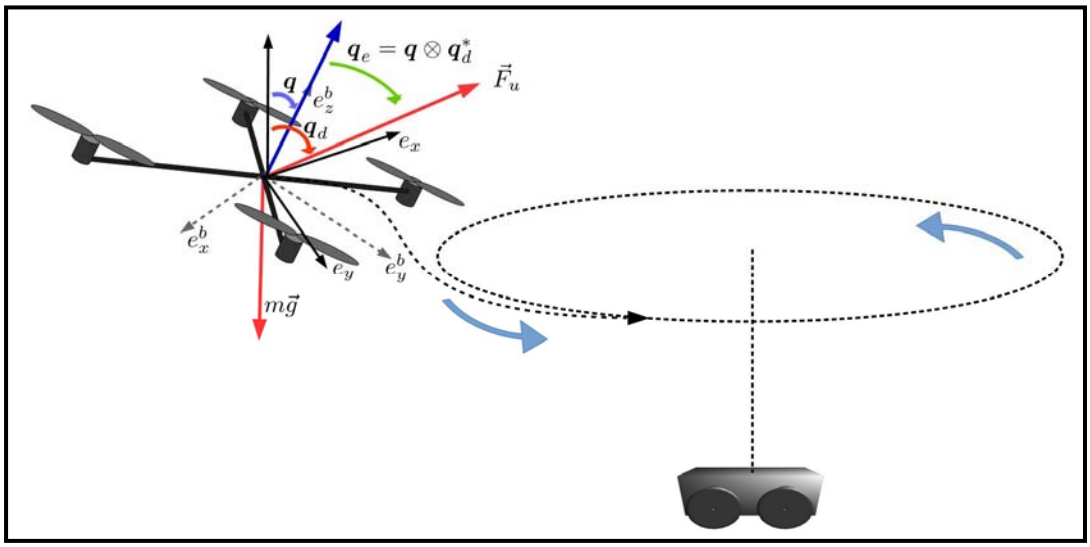
HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Par **Hernán ABAUNZA GONZALEZ**

Robust tracking of dynamic targets with aerial vehicles using quaternion-based techniques

Thèse présentée pour l'obtention du grade de Docteur de l'UTC



Soutenue le 26 avril 2019
Spécialité : Automatique et Robotique : Unité de recherche Heudyasic (UMR-7253)

D2480

SORBONNE UNIVERSITES, UNIVERSITE DE TECHNOLOGIE DE COMPIEGNE

HEUDIASYC, UMR 7253 CNRS

Ecole Doctorale “Sciences Pour l’Ingénieur”

ROBUST TRACKING OF DYNAMIC TARGETS WITH AERIAL VEHICLES USING QUATERNION-BASED TECHNIQUES

T H E S I S

presented by

Hernan ABAUNZA GONZALEZ´

to obtain the degree of

DOCTOR

Under de direction of:

Pedro CASTILLO GARCIA,
Charge de Recherche, CNRS

Alessandro CORREA VICTORINO,
Maître de Conférences

Compiègne, France. April 26 2019

SORBONNE UNIVERSITÉS, UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

HEUDIASYC, UMR 7253 CNRS

ÉCOLE DOCTORALE "SCIENCES POUR L'INGENIEUR"

SUIVI ROBUSTE DES CIBLES DYNAMIQUES AVEC VÉHICULES AÉRIENS À L'AIDE DE TECHNIQUES BASÉES EN QUATERNIONS

T H È S E

présentée par

HERNÁN ABAUNZA GONZALEZ

pour l'obtention du grade de

DOCTEUR

Sous la direction de:

PEDRO CASTILLO GARCIA,
CHARGÉ DE RECHERCHE, CNRS

ALESSANDRO CORREA VICTORINO,
MAÎTRE DE CONFERENCES

Compiègne, France. 26 Avril 2019

Acknowledgements

I want to give a big thank you to the institutions that supported this work from the begining. To the french *Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation (MESRI)* for the funding that was granted to my PhD project, to the Mexican *Consejo Nacional de Ciencia y Tecnología (CONACYT)* for the scholarship program that supported me and my family during my PhD studies in France. I want to also to thank the *Heudiasyc (Heuristique et DIAgnostique des SYstèmes Complexes)* laboratory and the *Ècole Doctorole "Sciences pour l'Ingènieur* at the *Université de Technologie de Compiègne (UTC)*, that made this work possible with their PhD programs. Also to the LABEX MS2T and ROBOTEX projects, that provided the laboratory with funding for material and equipment that was essential to the development of my PhD.

I want to specially thank my PhD directors Pedro Castillo García, Chargé de Recherches CNRS, at the Université de Technologie de Compiègne, and Alessandro Correa Victorino, Maître des Conférences at the Université de Technologie de Compiègne, for their invaluable advice, support, and counseling in this project from its begining to its ending.

I want to expressly thank Frederic Mazenc, Directeur de Recherche INRIA at CentraleSupelec, as well as Nicolas Marchand, Directeur de Recherche CNRS at GIPSA-lab, for accepting to be referees *(rapporteurs)* of my work. Their notations and advise were very appreciated to improve this thesis.

My thanks are also for Sergey Drakunov, Full Professor at Embry-Riddle Aeronautical University, and Véronique Cherfaoui, Professeur des Universités at the Université de Technologie de Compiègne, for accepting to be part of my jury and giving very helpful commentaries of my thesis.

I also want to thank Professor Pedro Garcia Gil, and Emeritus Professor Pedro Albertos Pérez at the Instituto de Automática e Informática Industrial from the Universitat Politècnica de València, and Professor Corina Sandu from the Advanced Vehicle Dynamics Laboratory at the Virginia Polytechnic Institute and State University, for receiving me in their laboratories during my internships, I hope to continue collaborating with all of you.

Special thanks to Guillaume Sanahuja, Gildas Bayard, Thomas Fuhrmann, Thierry Monglon, Sabine Vidal, Brigitte Azzimonti, and Nathalie Alexandre from the administrative and technique services of the laboratory, for their help in the development of this PhD in one way or another.

Thanks to my fellow PhD students and ex-students, members of the team that helped me in multiple occasions, Cristino de-Souza, Belem Rojas, Diego Mercado, Eusebia Guerrero, Fatima Oliva, Efrain Ibarra, Ariane Spaenlehauer, Julio Betancourt, Alexis Offermann, and Angel Alatorre. Thank you for all of the shared moments in collaborations, tests, courses, coffee breaks, trips, events, demonstrations, and many others. I also thank the students that I had the opportunity of supervising during their internships, to Luis Fernando Sanchez, Eduardo Alaniz, Victor González, Carlos Katt, Jorge Chavarin, Javier Lagunas, Erick Rico, Enrique Carvajal, Akari Bazaldua, Lesli Ramirez, and Andres Valle.

Finally, I want to thank my family, to my parents Hector Abaunza and Naela Gonzalez, who gave me the inital but essential formation that led my development as a researcher. To my wife Brenda Maya, who supported me since the begining of my PhD when we moved to France, until the end by giving me the most valuable support until the last but most difficult steps towards its conclusion.

Contents

Acknowledgements iii

Contents v

Summary ix

1 Introduction 1
- 1.1 Problem Statement . . . 2
  - 1.1.1 Objectives . . . 3
  - 1.1.2 Methodology . . . 3
- 1.2 State of the art . . . 4
  - 1.2.1 Unmanned Aerial Vehicles . . . 4
  - 1.2.2 Quadrotor control techniques . . . 8
  - 1.2.3 Quaternion-based approaches for quadrotors . . . 12
  - 1.2.4 Target tracking using UAVs . . . 14

2 Modeling Approaches 17
- 2.1 Force and moment in a rotor . . . 18
- 2.2 Classical Modeling Methods . . . 19
  - 2.2.1 Euler-Lagrange Approach . . . 19
  - 2.2.2 Quadrotor dynamic model . . . 20
  - 2.2.3 Newton-Euler Methodology . . . 22
  - 2.2.4 Quadrotor dynamic model . . . 23
- 2.3 Quadrotor Quaternion Modeling . . . 23
  - 2.3.1 Quaternion algebra . . . 24
  - 2.3.2 Rigid Body Dynamic Modeling . . . 26
  - 2.3.3 Quadrotor quaternion dynamical model . . . 26
  - 2.3.4 Decoupling the vehicle dynamics . . . 27
  - 2.3.5 Coupled Dynamics . . . 28
- 2.4 Dual Quaternion Modeling . . . 30
  - 2.4.1 Dual quaternion operations . . . 31
  - 2.4.2 Dual Quaternion Kinematics . . . 33
  - 2.4.3 Quadrotor Dual Quaternion Model . . . 34
- 2.5 Modeling approaches conclusions . . . 35

3 Control approaches for aerial vehicles **37**
3.1 Euler angles based controllers . . . 37
3.1.1 Sliding mode altitude control . . . 37
3.1.2 Backstepping control . . . 38
3.2 Quaternion state feedback controller . . . 40
3.2.1 Translational Controller . . . 41
3.2.2 Rotational Controller . . . 42
3.2.3 Simulation example . . . 43
3.3 Passivity-based quaternion control . . . 47
3.3.1 Classical PBC methodology . . . 48
3.3.2 PBC Methodology for a Quad-rotor using Quaternions . . . 48
3.4 Energy-based quaternion controllers . . . 50
3.4.1 Energy feedback controller . . . 51
3.4.2 Energy-based optimal control . . . 53
3.5 Geometrical bounding control . . . 54
3.5.1 Rotational Bounded Algorithm . . . 56
3.5.2 Translational Bounded Control Law . . . 57
3.6 Spherical chattering-free sliding mode control . . . 58
3.6.1 Attitude Control Formulation . . . 59
3.6.2 Position Controller . . . 61
3.7 Control approaches conclusions . . . 61

4 Navigation techniques for quadrotors **63**
4.1 Safe quadrotor navigation using arm commands . . . 64
4.1.1 Attitude gestures . . . 64
4.1.2 Electrical references from skeletal muscles . . . 66
4.1.3 Input based on a gesture sequence . . . 68
4.1.4 Safe Human-UAV Interaction . . . 69
4.1.5 Experimental results . . . 72
4.2 Path planning for a fleet of quadrotors . . . 76
4.2.1 Distributed path planning design . . . 78
4.2.2 Construction of the Cost function . . . 79
4.2.3 Emulated tests . . . 80
4.2.4 Experimental validation . . . 83
4.3 Quadrotor aggressive deployment . . . 90
4.3.1 Attitude Trajectory Formulation . . . 91
4.3.2 Experimental validation . . . 96
4.4 Quadrotor navigation conclusions . . . 100

5 Autonomous tracking of dynamic targets **101**
5.1 Circular trajectory for autonomous tracking . . . 102
5.1.1 Trajectory description . . . 102
5.1.2 Autonomous circular UGV tracking . . . 106
5.1.3 Quaternion-based Control . . . 106

5.1.4 Simulations 107
5.1.5 Experimental Validation 108
5.2 Coordinated circular UAV target tracking 113
5.2.1 Distributed Path Planning Algorithm 114
5.2.2 Target Locking 116
5.2.3 Control Algorithm 117
5.2.4 Emulated Results 117
5.2.5 Experimental validation 120
5.3 Autonomous target tracking conclusions 128

6 Conclusions and Future Perspectives 131
6.1 Future perspectives 132

A Publications 133
A.1 Book Chapters 133
A.2 International Journals 133
A.3 National Journals 134
A.4 International Conferences 134

Bibliography 135

Summary

The aim of this PhD work is to design control and navigation algorithms for tracking dynamic ground targets using aerial vehicles. An object was considered to navigate in the ground over a planar surface such that one or multiple aerial vehicles can autonomously describe trajectories to follow it. Control algorithms were also developed to robustly track the proposed trajectories.

A quadrotor configuration was taken into consideration for the development of aerial navigation algorithms, since this kind of platform is mechanically simple, versatile for performing aggressive maneuvers, and easily available for experimentation.

Most works currently found in the literature for quadrotors are based on classical approaches such as Euler angles, which can be understood intuitively, but arise problems such as discontinuities, singularities, gimbal-locks, and highly non-linear equations. Quaternions provide an alternative to classical representations, giving advantages such as their lack of singularities and gimbal lock effect, but the main one is their mathematical simplicity when handling rotations, which helps in the design of robust controllers and aggressive navigation algorithms.

Quadrotor quaternion controllers: The first part of this thesis consisted on developing quadrotor controllers with the aim of robustly tracking trajectories and performing precise navigation tasks. Initially, a simple linear feedback control algorithm based on unit quaternions was introduced, this controller revealed mathematical properties, which made it possible to map the quadrotor model such that its dynamics can be analyzed as a fully actuated system. Later on, more quaternion-based approaches were profoundly explored, resulting in more advanced algorithms such as:

- State-feedback quaternion controller.
- Passivity-based quaternion control.
- Energy-based controllers.
- Cilindrical bounded control.
- Spherical chattering-free sliding mode controller.

Autonomous navigation algorithms: In order to validate the previous quaternion-based controllers, several autonomous and semi-autonomous navigation schemes for aerial vehicles were introduced, the control algorithm for each scenario was selected from the ones previously developed.

Firstly, a quaternion feedback attitude controller was used along a safe navigation algorithm for piloting a quadrotor in semi-autonomous mode, using intuitive gestures from a user wearing an armband equipped with accelerometers, gyroscopes, and electromyographic sensors which trigger different actions in the quadrotor.

Then, autonomous trajectory generation and control approaches were explored for systems consisting on multiple aerial vehicles. In the context of a collaboration with the CRAN at the Université de Loraine, a path planning algorithm for quadrotors was proposed which generates vehicle trajectories in real time as a result of an online optimization of a distributed cost function, the trajectory was then robustly tracked using a quaternion-based controller.

Then, in order to improve the capabilities of aerial vehicles, and to facilitate their operability in unfavorable scenarios and spaces, an aggressive deployment strategy was proposed where a quadrotor is hand-tossed trough the air with its motors turned off, then it autonomously recovers from its free falling conditions using quaternion-based strategies to perform an autonomous or semiautonomous mission.

Autonomous target tracking algorithms: The last part of this thesis was dedicated to the conception of autonomous navigation techniques for tracking static and dynamic ground targets, combined with quaternion-based controllers to ensure system robustness.

First, a trajectory generation algorithm based on Hopf bifurcating differential equations was introduced for a single quadrotor for tracking a ground vehicle while describing circles, this technique inherently includes takeoff, tracking, centering and landing stages as part of the solution of a dynamic differential equations set.

Finally an extension of a distributed path planning algorithm was developed for two drones to autonomously follow a target ground vehicle while describing coordinated circles, the trajectory is obtained as the solution of an online optimization problem.

Chapter 1

Introduction

The topic of autonomous navigation on aerial and ground vehicles has been an important one for many researchers in the last years. Technological developments such as the miniaturization of electronic and mechanical components, constantly improving wireless communication systems, and the increasing capabilities of storage and computation devices have allowed an accelerating improvement of the capabilities of mobile robots.

Nowadays, it is relatively easy and cheap to build small Unmanned Aerial (UAVs) and Ground Vehicles (UGVs) which can be equipped with a wide range of cameras, sensors, and actuators that can assist in many activities. This versatility makes them very attractive, not only for military purposes, but also for numerous civil implementations including humanitarian assistance tasks such as search-and-rescue missions and fast response in disaster scenarios.

Both UAVs and UGVs present advantages and drawbacks inherent to their mechanical properties and configurations. On one hand, UGVs provide larger autonomy times than their aerial counterparts, but they find difficulties for example when navigating through large obstacles, or when surveying large areas. On the other hand, UAVs offer a 3-dimensional movement which gives them flexibility and privileged points of view for aerial imagery, and are able of reaching difficult access sites, they are also capable of performing more aggressive maneuvers, resulting in faster displacements, in contrast, they have a limited autonomy time and restricted payload capacities.

By combining aerial and ground vehicles for complex missions, their respective weaknesses can be counteracted by their combined capabilities such that the overall system has more advantages than a single robot. The dynamics of UGVs are commonly considered to evolve over a plane with near-zero inclination and with slow movements compared to UAVs, this makes them relatively easy to control and navigate. The main challenge when simultaneously employing both systems revolves around generating navigation algorithms and robust control techniques for aerial vehicles to ensure accurate tracking of the ground targets.

1.1 Problem Statement

Autonomously tracking moving targets using UAVs is not an easy task, many challenges have to be overcome in different stages, first, a mathematical model of the aerial vehicle needs to be constructed such that it accurately describes the real-world vehicle behavior while at the same time provides mathematical simplicity which helps in the development of controllers and navigation algorithms, then the vehicle rotational and translational dynamics must be stabilized using control algorithms capable of robustly following any given reference, finally, navigation techniques must be designed to follow the moving targets, and if multiple aerial vehicles are used, to coordinate them. Therefore, the problem is divided in three main axes.

The first part concerns the development of mathematical models that describe aerial vehicles dynamics. In previous years, multiple methodologies have been proposed and studied by many researchers, by combining solid theories in mechanics and physics such as Newton's equations of motion, and the laws of energy conservation, with mathematical descriptions of rotation and translation sequences such as Euler angles, rotation matrices, quaternions, among others, such that the vehicle behavior is described in one form or another.

Nevertheless, since the dynamics of aerial vehicles are generally unstable, complex, nonlinear, and underactuated, the design of control and navigation algorithms becomes difficult. To deal with this problem many works have proposed approaches that simplify mathematical expressions to avoid dealing with the undesired characteristics of the vehicle dynamics, for instance, it is a common practice to consider only slow movements and small angles, such that many nonlinear terms can be neglected, another common practice is to analyze the system only in 2 dimensions, ignoring some effects that arise in 3-dimensional movements. All of these considerations, although they help with the algorithm development tasks, put limitations on the real capacities of UAVs such as their fast motion and reaction capabilities.

Non-conventional approaches such as using quaternions to describe the vehicle dynamics can provide mathematical simplicity without sacrificing accurateness and generality, this becomes helpful for the second part of the problem which consists on developing better controllers, capable of performing bolder maneuvers than conventional algorithms, giving the capability of performing aggressive and fast flights, while giving robustness to the system against unknown disturbances, and ensuring a safer operation.

Finally, navigation algorithms have to be designed in order to enable autonomous flight skills for aerial vehicles. Trajectory generation and tracking techniques are a common approach that has been explored, sometimes to find the best path for the UAV to arrive to its destination, other times to coordinate with multiple vehicles in cooperative missions, in other scenarios these techniques can be used to track or survey fixed or moving targets.

1.1.1 Objectives

This thesis is dedicated to the development of control and navigation algorithms for aerial vehicles, more specifically quadrotors, with the goal of robustly tracking dynamic targets.

To achieve this goal, and to deal with the challenges that arise from this problem, the first objective is to develop control algorithms to ensure stability for the vehicle while enabling path following and trajectory tracking capabilities and providing robust performance against disturbances and model uncertainties. By studying the different methodologies for UAV modeling, quaternion-based approaches have emerged as the best suited for this work, since they provide intrinsic robustness to the system by avoiding singularities and providing mathematical simplicity without needing conservative considerations such as small angles and velocities.

The second objective is to propose quadrotor navigation strategies to enhance the vehicle functionalities. Algorithms for safely interacting with operators are desired to facilitate the implementation of these systems in real-world applications, intuitive semi-autonomous piloting and easy deployment and recovery strategies under unfavorable conditions are some of the techniques that have been explored in this thesis. Also, since the ultimate goal is to track moving targets using one or more UAVs, autonomous navigation algorithms for a fleet of quadrotors are aso concerned to reach this goal.

Finally, the third objective is to introduce navigation strategies for tracking moving ground targets, trajectory generation algorithms that include the translational behavior of a ground vehicle will be designed such that it can be surveyed using one or multiple quadrotors. Techniques such as the solution of a mathematical formulation will be considered such that its resolution yields the desired tracking behavior of the aerial vehicles.

1.1.2 Methodology

As it was previously stated, quaternions provide an alternative to classical representations, that gives many advantages to the development of navigation strategies for UAVs. The most important for the design of quadrotor controllers, is that they can be used to map the nonlinear behavior of the system into an equivalent representation that is easier to handle without needing to restrict the vehicle capabilities by limiting its inclination angles or velocity.

For this reason, the preferred methodology to design the controllers of this thesis is by using quaternion representations of rotations, which are employed to describe the vehicle dynamics, to introduce control algorithms, and to describe rotation references that regulate the desired movements of the vehicle.

As for the design of trajectory generation and fleet formation algorithms, two methodologies will mainly be studied, one based on a differential equations set that yield the path of a quadrotor that tracks a moving UGV. The other one will be based on the real-time solution of a distributed optimization problem, such that it results on the best trajectory for a group of quadrotors that survey said ground target.

All of the proposed approaches will be validated using simulations and experiments. Since it is desired to give quadrotor as much autonomy as possible, the algorithms will also be designed with the aim of implementing them on the onboard computers which are embedded on the quadrotors, such that they can be run in real-time.

1.2 State of the art

1.2.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs, also known as aerial drones) are flying machines that lack of an onboard human pilot, and also have a certain degree of autonomy. The design and configuration of a drone can vary according to its specifications and applications.

Even if a drone can be built at any size, ranging from a few centimeters to several meters, most designers choose to keep small sizes (from a couple of centimeters to a little more than one meter) offering great flight abilities and better energy efficiency while maintaining a low cost. These vehicles are known as minidrones or mini-UAVs.

Modern drones began existing in the second half of the *XXth* century when the concept of "Remotely Piloted Aircraft Systems" started to be popularized among some researchers and aviators. Nevertheless, miniaturized mechanical and electronic technology was, at this time, very expensive and not as advanced, requiring a very skilled ground pilot [1], therefore, real implementations of these vehicles were almost impossible.

In the years 2010's, technological milestones drastically reduced the cost of miniature electronics which in contrast became increasingly powerful, some examples are microprocessors, cameras, Global Positioning Systems (GPS), Inertial Measurement Units (IMUs), batteries, and motion capture systems. This progress revolutionized the conception of many different mini-UAVs because many researchers, companies, and even hobbyists could now afford the cost of developing their own UAVs [2].

UAVs can be categorized in three general groups in terms of their mechanical architecture, which will be detailed in the following subsections.

1.2.1.1 Fixed-wing Aerial Vehicles

For these vehicles, the lift force (the one that sustains the aircraft in the air) is generated by the airflow that hits a fixed surface of the vehicle (wings), which is dependent on the forward translational velocity. A simple example is a classical airplane, see Figure 1.1.

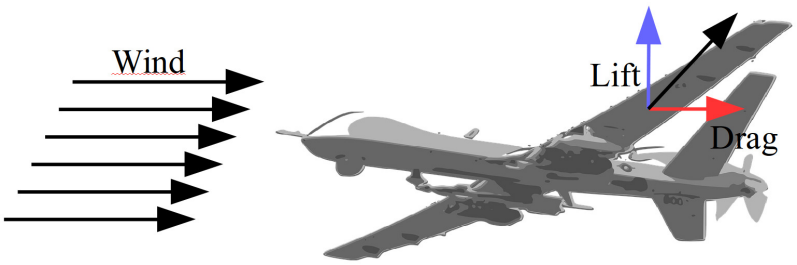


FIGURE 1.1: Example of a fixed-wing unmanned aerial vehicle with the forces acting on its wings.

It is possible to describe the mechanical behavior of a fixed-wing Aerial Vehicle (AV) by using nonlinear equations, for more details see [3] - [5], a simplified model is usually considered where aerodynamic and mechanical effects such as turbulence, vibrations, and other disturbances can be neglected, it is also hypothesized that the earth is locally flat and the AV's mass is constant.

The state equations of these drones are defined by x, y, z, ψ and v, which represent the Cartesian coordinates, the yaw angle, and longitudinal speed respectively, see Figure 1.2. The kinematic and dynamic navigation equations of an airplane are then given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \psi \\ v \end{bmatrix} = \begin{bmatrix} v\cos(\gamma)\cos(\psi) \\ v\cos(\gamma)\sin(\psi) \\ v\sin(\gamma) \\ \frac{gn}{v}\frac{\sin(\varphi)}{\cos(\gamma)} \\ \frac{1}{m}(T_r - D) - g\sin(\gamma) \end{bmatrix}, \tag{1.1}$$

where the command inputs are the thrust force T_r, the banking angle φ and the angle of attack γ. The gravitational acceleration is represented by g. The drag force D is written as

$$D = \frac{1}{2}\rho v^2 S C_D, \tag{1.2}$$

where S symbolizes the wing projected surface area and ρ represents air density. The lift force is defined as

$$L = \frac{1}{2}\rho v^2 S C_L, \tag{1.3}$$

C_L and C_D respectively represent the drag and lift forces aerodynamic parameters, which are dependent on the wing's geometry. The relationship between the lift force and the vehicle mass m is denoted as

$$n = \frac{L}{mg}. \tag{1.4}$$

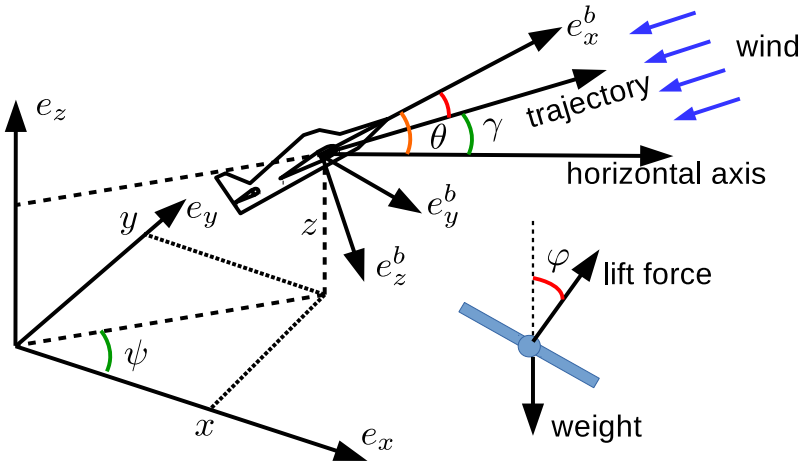


FIGURE 1.2: Diagram with some of the most common angles in an airplane model, where e_x, e_y and e_z denote the world-fixed frame axes, while e_x^b, e_y^b and b_z^b symbolize the moving body reference coordinate system.

It is easy to observe from the previous equations that the dynamic model of an airplane is complex, even if many aerodynamic effects are neglected. The model is nonlinear with multiple inputs and outputs, which represents a real challenge for designing control strategies.

1.2.1.2 Rotating-wing AVs

These vehicles generate a lift force from rotating propellers which are attached to their motors, see Figure 1.3. The term "rotating wing" comes from the fact that propellers can be considered as wings that rotate around an axis.

A direct example of these aircraft are helicopters. Nevertheless, many different configurations consisting on several rotors have been proposed in the last decades such as quadrotors, hexarotors and octarotors, which consist on four, six, or eight aligned opposing propellers.

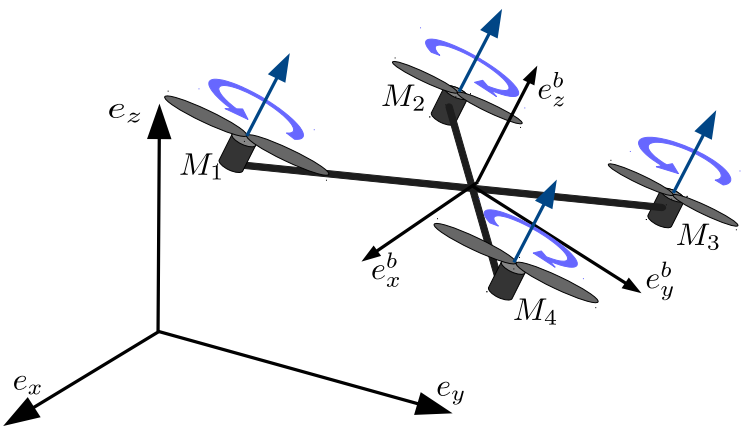


FIGURE 1.3: Rotating-wing vehicle principle, where the forces are generated by rotating propellers.

Quadrotors are one of the most popular platforms for UAV research due to their mechanical simplicity, flight capabilities, and relatively low-cost implementation. In its classical configuration, two diagonal opposing propellers rotate clockwise (M_2 and M_4) while the other two rotate counter-clockwise (M_1 and M_3) such that gyroscopic effects and aerodynamic couples tend to compensate each-other in stationary flight. This vehicle is an underactuated system since it possesses four control inputs and six degrees of freedom.

Due to its versatility, availability, and mechanical characteristics, a quadrotor was chosen in this thesis for developing control and navigation algorithms although all of the proposed approaches can be extrapolated to other multirotor configurations. Detailed modeling and control techniques for quadrotors will be described in the next chapters.

1.2.1.3 Hybrid UAVs

Some developers have conceived new UAV configurations by combining mechanical properties of both fixed and rotating-wing systems. These vehicles then usually comprise rotating propellers and fixed wings at the same time.

The idea of these platforms is to be capable of performing plane-like maneuvers (like surveying large surfaces in a short time span) and tasks like stationary flight, vertical takeoff and landing using the same vehicle.

Tandem-wing Tail-sitter UAV example: In [6] and [7], the authors conceived an aerial vehicle that reproduces the characteristics of a helicopter (vertical takeoff and landing) and an airplane (horizontal flight), see Figure 1.4. It uses rotating propellers that generate a vertical force during the takeoff and landing stages, and a horizontal force after transitioning into airplane mode.

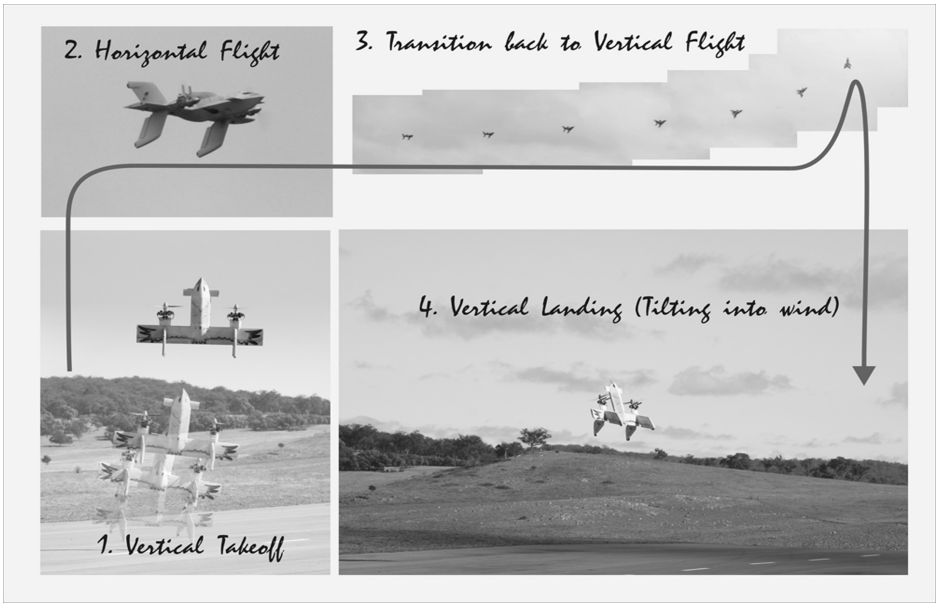


FIGURE 1.4: Tandem-wing vehicle combining helicopter and airplane features, built by the University of Sidney, presented in [6] and [7].

This vehicle combines principles from both the airplane and quadrotor models, in [6], a simplified expression of the vehicle dynamics (named *T-wing*) is described where the main equations are

$$
\begin{array}{rcll}
\dot{u} & = & rv - qw - g\cos(\psi)\cos(\theta) + \frac{F_x}{m}, & c_1 = \dfrac{I_{yy} - I_{zz}}{I_{xx}}, \quad c_3 = \dfrac{1}{I_{xx}}, \\
\dot{v} & = & -ru + pw + g\sin(\psi)\cos(\theta) + \frac{F_y}{m}, & c_5 = \dfrac{I_{zz} - I_{xx}}{I_{yy}}, \quad c_7 = \dfrac{1}{I_{yy}} \\
\dot{w} & = & qu - pv - g\sin(\theta) + \frac{F_z}{m}, & c_8 = \dfrac{I_{xx} - I_{yy}}{I_{zz}}, \quad c_9 = \dfrac{1}{I_{zz}}, \\
\dot{\phi} & = & \dfrac{\cos(\psi)}{\cos(\theta)}p - \dfrac{\sin(\psi)}{\cos(\theta)}q, & \dot{p} = c_1 rq + c_3 L, \\
\dot{\theta} & = & \sin(\psi)p + \cos(\psi)q, & \dot{q} = c_5 pr + c_7 M, \\
\dot{\psi} & = & \cos(\psi)\tan(\theta)p + \sin(\psi)\tan(\theta)q + r, & \dot{r} = c_8 pq + c_9 N,
\end{array}
$$

where u, v, w are the angular speeds with respect to the body reference frame of the vehicle around the x, y and z directions respectively, while p, q, r represent the translational speeds with respect to the same axes. Inertial terms are symbolized by I_{xx}, I_{yy} and I_{zz}, considering a quasi-symmetrical vehicle geometry. L, M, N define the torques over each axis, and g denotes the gravitational acceleration.

This model is nonlinear, which makes the conception of control algorithms a difficult task. To overcome this obstacle, symplified hypothesis can be made. For instance, considering quasi-stationary movements (small angles) yields

$$
\begin{array}{rcl}
u & \rightarrow & u_1 + \delta u, \\
v & \rightarrow & 0 + \delta v, \\
w & \rightarrow & 0 + \delta w, \\
p & \rightarrow & 0 + \delta p, \\
q & \rightarrow & 0 + \delta q, \\
r & \rightarrow & 0 + \delta r, \\
\phi & \rightarrow & 0 + \delta \phi, \\
\theta & \rightarrow & 0 + \delta \theta, \\
\psi & \rightarrow & 0 + \delta \psi,
\end{array}
\quad \Rightarrow \quad
\begin{array}{rcll}
\dot{u} & = & -g + \dfrac{F_x}{m}, & \dot{p} = \dfrac{L}{I_{xx}}, \\
\dot{v} & = & -\Delta r u_1 + g\Delta\psi + \dfrac{F_y}{m}, & \dot{q} = \dfrac{M}{I_{yy}}, \\
\dot{w} & = & \Delta q u_1 - g\Delta\theta + \dfrac{F_z}{m}, & \dot{r} = \dfrac{N}{I_{zz}}, \\
\dot{\phi} & = & p, & \\
\dot{\theta} & = & q, & \\
\dot{\psi} & = & r, &
\end{array}
$$

This simplified model was used in [7] to develop control strategies and implement them in a real vehicle.

1.2.2 Quadrotor control techniques

In recent years, there has been an increasing interest in the robotics and control research communities towards developing different strategies for controlling and navigating quadrotors, in the path towards obtaining better performances in terms of robustness, precision, efficiency, preferably while maintaining simplicity in mechanical, mathematical, and computational terms.

In the last years of the XXth century, autonomous control for rotorcraft started to be introduced in the literature as a case of study for controlling some classes nonlinear

systems, for instance [8] proposed a nonlinear controller for *slightly non-minumum phase nonlinear* systems, the authors used the dynamic equations of a V/STOL aircraft to exemplify the behavior that a real system could have with this kind of controllers. Other examples are [9], which included the control of a quadrotor as an academic example for classical control system design along with other systems such as automotive engines, satelites, and biological systems, and [10], which proposed the use of neural networks for flight control.

However, at that time, the high cost of miniaturized power electronics, and the short capabilities of microprocessors and microcontrollers made it very difficult to experimentally validate and implement this kind of systems, therefore, the motivation of developing control strategies for miniaturized UAVs was limited, since it was considered to be just an interesting concept.

It was until the first decade of the *XXI*th century, when researchers started focusing on specifically controlling miniaturized quadrotor vehicles, one of the first works was [11], where the authors proposed to separate the rigid body dynamic equations, which were explained using Euler angles, from the motor dynamics, which were expressed using blade theory, this idea has been applied in many works that came afterwards. Then, in [12] and [13], visual sensors were included in the control loop of a quadrotor which was restricted such that only yaw and vertical movements could be performed. Restricting movements of experimental platforms was a common practice in the early development of UAVs, in [14], implemented classical Proportional-Integral-Derivative (PID) and Linear-Quadratic (LQ) controllers to control the orientation of a quadrotor which was fixed on a 3D universal joint, see Figure 1.5. As powerful embedded technology was still expensive and not widely available, most works at the time computed all of the numerical operations in a stationary personal computer, and then transmitted the control inputs to the motors using wired communications.

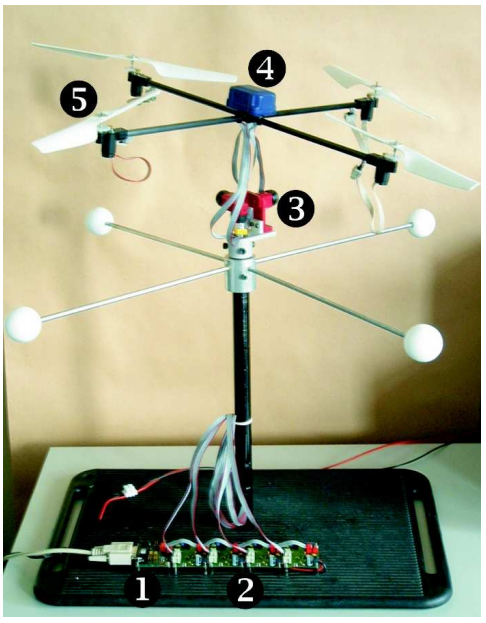


FIGURE 1.5: Restricted test-bench used in [14], 1)RS232 to I2C translator, 2)Motor modules, 3)3D universal joint, 4)Inertial Measurement Unit, 5)Propellers.

Some works from these years focused on developing controllers for a simplified rotorcraft model consisting on a 2-dimensional configuration named Planar Vertical Takeof and Landing (PVTOL), some examples can be found in [15]-[20]. The interest of studying this kind of vehicles is to better understand their 3-dimensional counterparts since the dynamics of a quadrotor can be seen as two combined PVTOL systems crossed at their centers.

Among some of the works that presented advances for quadrotors during this time, [21] and [22] proposed nonlinear control strategies to deal with the rotational and translational dynamics of quadrotors, their technique was based on separating the dynamic equations in three subsystems (altitude-yaw, y-axis-roll, and x-axis-pitch), the control algorithm was based on considering small values for the pitch and roll angles (quasi-stationary flight), such that the dynamic model can be expressed as a set of cascade integrators. In these works, experiments were performed in an unattached quadrotor, which was able to move in its three attitude angles and freely hover in a 3-dimensional space, however, a personal computer was required to acquire signals from sensors and compute control inputs, see Figure 1.6.

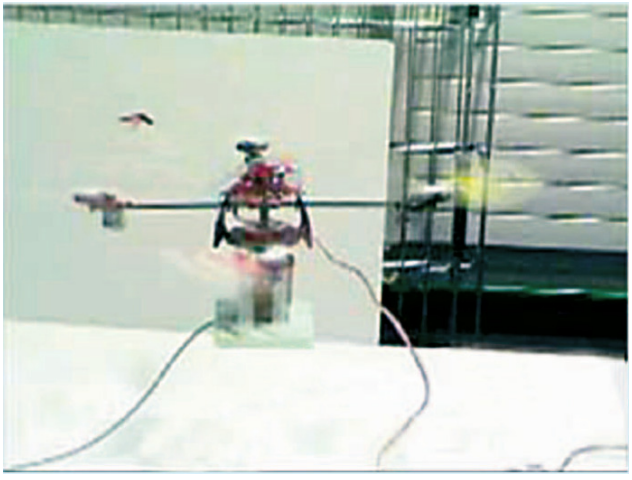


FIGURE 1.6: Quadrotor for experimental validation in [22], the vehicle is able to hover freely, however it is connected to a desktop computer for its stabilization.

At the second half of the 2000's decade, the availability and power of microprocessors increased at an accelerating pace while their price decreased, this greatly facilitated the access of embedded electronics for researchers, companies, and consumers, which sped up the development of controllers that could be tested on real miniaturized UAVs (the introduction of smartphones and other portable devices benefited from this same technology, therefore their success in the market happened at around the same time).

Although the availability of embedded electronics represented an impulse on UAV research, many of the proposed algorithms still considered small angles to simplify the quadrotor dynamic equations and facilitate the development of controllers. For instance, in [23], the position of a quadrotor was measured by a Motion-Capture system, and stabilized using an Integral-Backstepping controller programmed onboard, obtained by a linear small angles simplified dynamic model, other works that proposed similar approaches can be found in [24]-[29].

More recent contributions have further exploited the capabilities of modern electronics, and developed more advanced nonlinear control algorithms, for instance, [30] introduced an adaptive robust controler, however, the assumptions made by the authors require to consider small angle rotations and slow movements to ensure system stability due to the model complexity. Other works like [31]- [34] employed sliding mode algorithms to compensate external disturbances, but also consider conservative slow movements to design the control equations.

Among other nonlinear control techniques, authors have also used saturating function to stabilize quadrotors, for example [35] employed bounded functions combined with a backstepping technique to design an attitude controller for a quadrotor which proved to have robustness capabilities, then [36] combined a similar algorithm with a state predictor to improve the performance of UAVs in the presence of delayed measures.

Nowadays the capabilities of UAVs are increasing towards performing more impressive tasks that involve more complex maneuvers. For instance, [37] introduced an algorithm that tracks aggressive trajectories for obstacle avoidance, the authors employed differential flatness to execute complex navigation trajectories, however, the proposed algorithm's complexity such that, although embedded computers are nowadays accessible and available, it needs an external desk computer to calculate control trajectories. In [38] and [39], authors proposed algorithms for estimating the vehicle states and generating trajectories for flying through narrow gaps at different inclinations by computing the required quadrotor acceleration to pass through the gap at a plausible inclination, see Figure 1.7, however, the proposed approaches compute translational that are tracked by third-party libraries, thus do not directly deal with attitude signals and computing rotations.

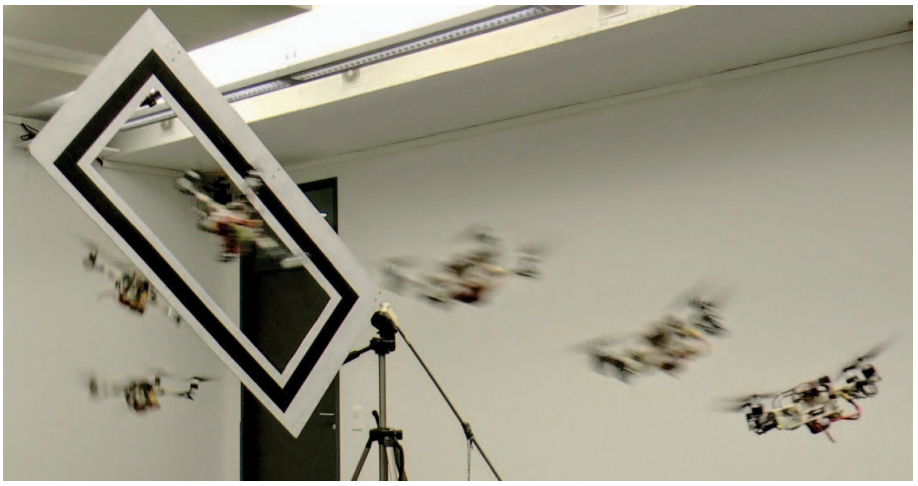


FIGURE 1.7: A quadrotor flying through narrow gaps in [39].

Other works like [40] and [41] addressed the problem of flying quadrotors in urban areas under wind gust conditions where turbulence may interfere with the vehicle operation, the authors compared several control algorithms based on integrating terms, backstepping techniques, and adaptive properties to deal with the effects of windy environments.

Until very recently, most of the controllers developed for UAVs use the classical Euler Angles representation to describe the quadrotor dynamics, resulting in complex nonlinear multi-variable equations that are hard to deal with. Many researchers have often made considerations like small-angle movements, single-axis displacements, or hovering conditions, other times, complex algorithms have been proposed which are hard to implement.

An alternative to Euler angles is designing controllers based on quaternions, which provide a simpler singularity-free description of the vehicle dynamics, making the task of designing robust and precise controllers easier.

1.2.3 Quaternion-based approaches for quadrotors

Quaternions are a kind of numbers proposed by Sir William Rowan Hamilton in the nineteenth century [42], as tool for describing 3-dimensional rotations with a set of *hypercomplex* numbers (symbolized by one real plus three imaginary parts). These numbers have been widely used in theoretical and experimental physics, but more recently have been applied to other fields such as computer graphics animation and robotics. Some works that explain an introduction to quaternion notions, operations and some applications are [43] - [45].

The application of quaternions for controlling aircraft systems was introduced in [46], where the authors used quaternion products to rotate control vectors between different reference frames, see Figure 1.8. It was remarked that quaternions are not affected by some undesired effects that are inherent to Euler angles such as Gimball locks, singularities, and discontinuities, the authors also emphasized that quaternion operations require less computational resources than computing rotation matrices from other approaches.

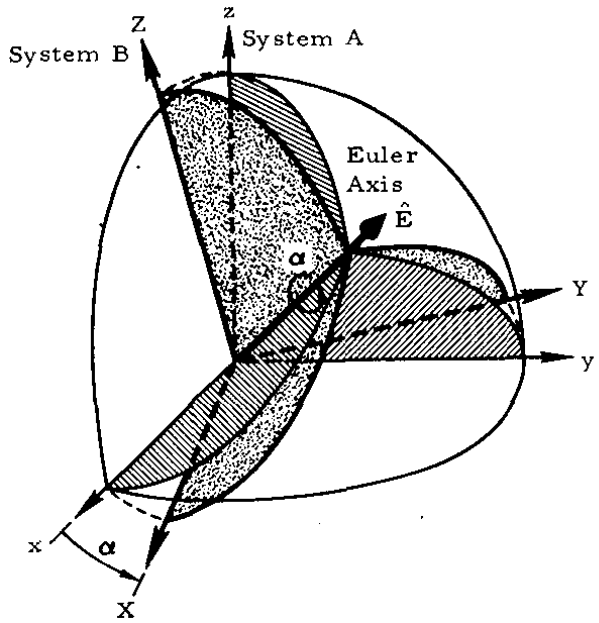


FIGURE 1.8: Rotation around an axis, following Euler's theorem to formulate a quaternion according to Hamilton, as illustrated in [46].

Later on, [47] proposed a state-feedback algorithm based on quaternions to perform large maneuvers in spacecraft involving satellite deployment with space shuttles, the authors highlighted that a relatively simple quaternion-based control algorithm could stabilize spacecraft without requiring to maintain small angles, and avoiding discontinuities, such properties were further studied by the same authors in [48]. Some other works were also presented, where sliding mode theory was applied quaternion-based dynamic models, such as [49] -[51], however, most of these contributions were limited to generic rigid spacecraft. Applications of quaternion theory to UAVs, were explored some years later.

One of the first works that used quaternions for controlling quadrotors was introduced in [52], where the authors applied quaternion-based dynamic equations to derive the quadrotor attitude model, then used a feedback controller (previously developed for spacecraft) to stabilize the quadrotor orientation, the same authors validated this idea on an experimental platform on [53] (using a similar configuration as Figure 1.5). Later on, [54] applied this control strategy on a detached quadrotor (which was able to unrestrictedly fly), and analyzed the effects on communication and network faults and proposed observers and fault-tolerant controllers for the attitude dynamics, while [55] proposed a strategy to stabilize the quadrotor position partly using quaternion operations, but only employed simulations to validate their proposal.

On the second half of the 2000's decade, progress was made by some authors on developing more quaternion-based control strategies. For instance [56]-[58] developed a bounded attitude controller using saturation functions on a sliding manifold defined by the vehicle angular velocity and the imaginary parts of the quaternion attitude, the authors validated this control strategy on a prototype illustrated in Figure 1.9.

FIGURE 1.9: Quadrotor experimental setup for testing quaternion-based control algorithms proposed in [57].

Some other quaternion-based techniques were also applied to quadrotor vehicles such as [59], where the authors proposed a quaternion high-order sliding mode algorithm to control attitude trajectory on spacecraft, while [60] also introduced a quaternion sliding mode control for spacecraft, but with adaptive properties that consider actuator saturation scenarios. A comparison of several quaternion-based control algorithms for quadrotors was introduced by [61], where proportional-derivative (PD), linear quadratic regulartor LQR, and backstepping position and attitude controllers were compared with simulations.

In another work, a quaternion-based nonlinear P^2 controller, for solving the attitude problem of a quad-rotor was proposed in [62]. In [63], a quaternion-based feedback controller is developed for the attitude stabilization of a quad-rotor. The control design takes into account a priori input bounds and is based on nested saturations. The authors forced the closed-loop trajectories to enter a fixed neighborhood around the origin in a finite time and remain thereafter. A quaternion-based nonlinear robust output feedback tracking controller was developed in [64] to address the attitude and altitude tracking problem of a quadrotor, here, approximation components based on neural networks are introduced to estimate model uncertainties and robust feedback components are designed to compensate for external disturbances.

More recent contributions have explored other properties of quaternion-based dynamic systems. For example, [65] and [66] proposed fractional sliding-mode controllers using quaternions to stabilize the quadrotor attitude dynamics, while also considering finite-time convergence of the system dynamics based on a fractional-order system. In [67], researchers estimated quadrotor position and attitude information during aggressive trajectories (with high accelerations and angular velocities) by only employing cameras and inertial sensors, quaternion operations were used in the proposed algorithm.

In this thesis, quaternion-based controllers for quadrotors were proposed with the objective of robustly tracking translational and rotational dynamic trajectories, part of the contributions presented during this PhD lie on using quaternions to deal with the underactuated nonlinear nature of quadrotors in the development of controllers and navigation algorithms, as it will be explained in the following chapters.

1.2.4 Target tracking using UAVs

As scientific and technological advancements have made UAVs more powerful, miniaturized, and affordable, many applications have been emerging at an increasing pace. One of such topics is the application of aerial vehicles to track moving targets, which have been attracting the attention of researchers in recent years.

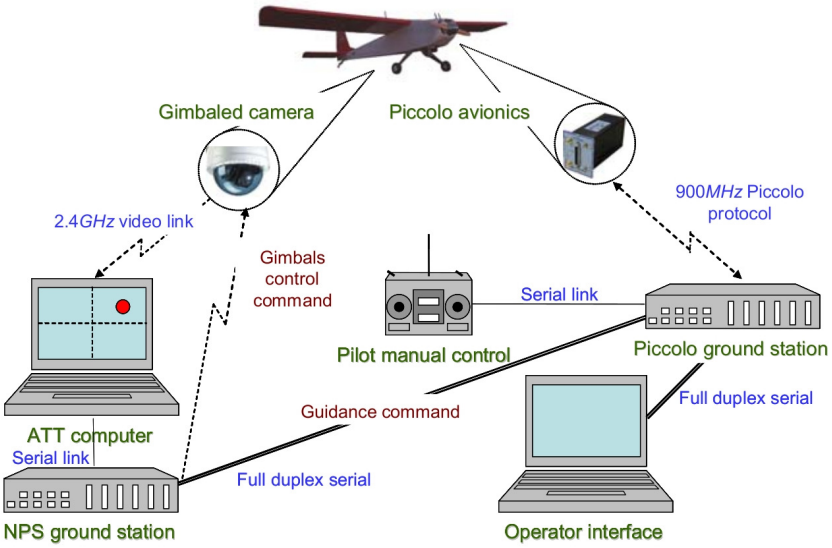


FIGURE 1.10: Flight control scheme used in [68] to track a moving target using fixed wing UAVs.

The first works that addressed this topic began in the second half of the 2010's decade, for instance, [68] proposed an algorithm to fly an autonomous airplane with a spiral trajectory that tracks a moving target in its center, the proposed approach required two connected ground stations, one to compute the autonomous path, and another to serve as an operator interface. In [69], the authors proposed a cooperative scheme, based on attracting vector fields, to track a moving target using two fixed-wing UAVs that describe circles while respecting a safety distance.

Other works have proposed heterogeneous robotic systems where multiple UAVs and Unmanned Ground Vehicles (UGVs) cooperate to achieve a common goal. However, this kind of approaches are often very demanding on computational and communication resources, therefore, many authors only validate their proposals in simulations. For instance [70] proposed a control scheme to coordinate groups of UAVs and UGVs using a decentralized flocking algorithm, which was validated on a simulation that does not consider the nonlinear UAV dynamics. Other examples are [71] where the challenge of tracking a ground target using UAVs in dense obstacle areas was formulated as an optimization problem, which is solved on a simulation environment, and [72] where the optimization problem is solved using dynamic programming, also in simulations.

Some researchers have directed their focus on the task of detecting the moving target, for instance [73] employed a vision algorithm to detect a moving ground vehicle using pixel characteristics, a particle filter was used to predict the target position when obstacles hinder its visibility, correcting the data once its sight is recovered. An alternative approach was presented in [74], where an occupancy grid is used to take into account fixed obstacle positions to model the target states, updating vision measurements with a Bayesian filter. [75] introduced a collaborative control technique in which a UAV takes off from the initial location of a UGV, then flies to provide an aerial point of view of the mission while simultaneously tracking the UGV's trajectory, then it lands at the new position of the ground vehicle.

More recent works have been implementing nonlinear flight controllers to track targets, for instance [76] proposed a moving path strategy to track single or multiple targets using fixed wing UAVs, then, nonlinear controllers developed with Lyapunov theory were used to track the computed trajectories. In [77], target tracking and 3-dimensional obstacle avoidance for autonomous airplanes is proposed by using Lyapunov guidance vector fields and fluid dynamic properties.

Due to the increasing advantages and availability of quadrotors, some works have used them to track moving targets, for instance [78] employed a tag detection algorithm to detect a landing platform located on a moving ground vehicle. The proposed approach fuses visual information with GPS signals using a Kalman filter to better estimate the landing surface trajectory. The problem of landing on a moving platform was also addressed in [79], where the proposed solution added an Uncertainty and Disturbance Estimator to compensate wind disturbances generated when the vehicle approaches to the landing site, and also un-modeled dynamics.

Chapter 2

Modeling Approaches

The first step towards designing control and navigation algorithms is to state a mathematical model to work with. In this chapter, several modeling methodologies are explored such that robust algorithms can be developed in the following chapters. Most of the work is focused on aerial vehicles since they are in general, more challenging control and navigate compared to UGVs due to their faster dynamics and unstable nature, the selected configuration for the UAV in this thesis was a quadrotor multicopter.

Most researchers have relied on the classical Euler angles [80] to represent the dynamics of UAVs. This approach is intuitive and easy to implement, specially when simplifications are considered, e.g. maintaining small angles, slow movements, etc. However, when more arduous designs, tasks, or applications are involved, Euler angles representations encounter some problems, such as complicated non-linear mathematical expressions, singularities, and gimbal locks.

Quaternions provide an alternative for representing the rotation of rigid bodies and have great advantages compared to Euler angles, i.e, lack of discontinuities and gimbal lock, and provision of mathematical simplicity. When multiple rotations and translations are considered in more complex systems, dual quaternions also become very useful to describe the transformation of rigid bodies.

This chapter is organized as follows: Section 2.1 introduces the elemental forces of a quadrotor aircraft. In Section 2.2 some of the classical modeling methodologies based on Euler angles are presented for comparison purposes. An explanation of the quaternion-based modeling methodology is detailed in Section 2.3, while Section 2.4 introduces a modeling approach based on dual quaternions. Finally some conclusions are discussed in Section 2.5.

2.1 Force and moment in a rotor

According to blade element theory, which is used to model airfoil and rotor performances, the forces and moment developed on a uniform wing are determined by the lift and drag forces and a pitching torque [81]. For a given rotor i with angular velocity ω_i, the linear velocity at each point along the propeller is proportional to the radial distance from the rotor shaft. Thus the following equations [82] can be stated:

$$f_i = C_T \rho A_p \, r^2 \omega_i^2, \tag{2.1}$$

$$\tau_i = C_Q \rho A_p r^3 \omega_i^2, \tag{2.2}$$

where f_i represents the total thrust produced by rotor $i = 1, ..., 4$ acting perpendicularly to the rotor plane neglecting blade flapping effect, τ_i describes the rotor torque, r denotes the rotor radius, ρ symbolizes the air density and $A_p = \pi r^2$. C_T and C_Q are non-dimensionalised thrust and rotor torque coefficients, which can be computed using blade element theory [83], see Figure 2.1.

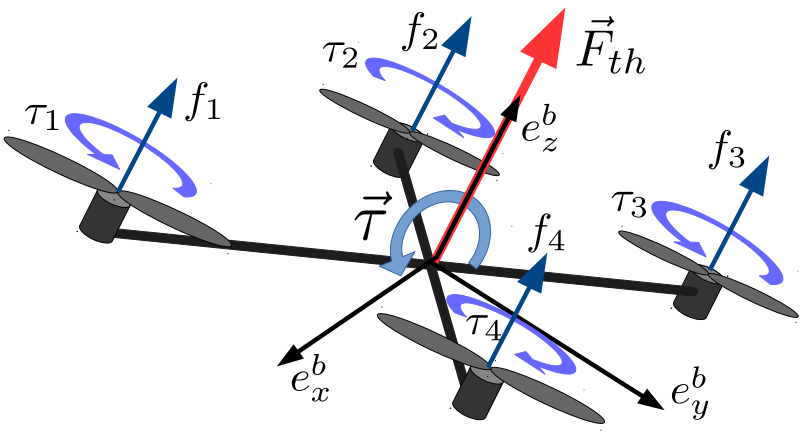


FIGURE 2.1: Propeller forces and torques acting on a quadrotor.

It is a common practice to consider the aerodynamic parameters from (2.1) and (2.2) as constants $k_T \approxeq C_T \rho A_p \, r^2$, $k_Q \approxeq C_Q \rho A_p r^3$. Taking into account a quadrotor symmetrical configuration, the total torque and thrust force produced on the vehicle by the propellers is computed as

$$\vec{F}_{th} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} k_T \, \omega_i^2 \end{bmatrix}, \quad \vec{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} l \left(k_T \, \omega_1^2 - k_T \, \omega_2^2 - k_T \, \omega_3^2 + k_T \, \omega_4^2 \right) \\ l \left(k_T \, \omega_1^2 + k_T \, \omega_2^2 - k_T \, \omega_3^2 - k_T \, \omega_4^2 \right) \\ \sum_{i=1}^{4} k_Q \omega_i^2 \, (-1)^i \end{bmatrix}, \tag{2.3}$$

where τ_x, τ_y, and τ_z denote the total torque components produced in each axis of the body reference frame, and $\vec{F}_{th}$ represents the total thrust force, acting in the vertical axis of the quadrotor.

2.2 Classical Modeling Methods

The two most popular techniques for modeling aerial vehicles will be presented in this section, the Euler-Lagrange and Newton-Euler approaches, both introduced for an ideal case, i.e. without perturbations and/or uncertainties in the model.

2.2.1 Euler-Lagrange Approach

Considering the representation of an aerial vehicle as a solid body evolving in a three dimensional space and subject to the main thrust and three torques. In this work, letters with arrows over them represent vectors in 3D space $\vec{(\cdot)} \in \mathbb{R}^3$. The generalized coordinates of the vehicle are defined by $\mathbf{x}_{quad} = (\vec{p}, \vec{\eta}) \in \mathbb{R}^6$, where $\vec{p} = (x, y, z) \in \mathbb{R}^3$ denotes the position vector of the center of mass relative to a fixed inertial frame $\mathcal{I}$, and $\vec{\eta} = (\psi, \theta, \phi) \in \mathbb{R}^3$ defines the quadrototor attitude in its Euler angles (yaw, pitch and roll angles, respectively) notation. The Lagrangian equation is defined as

$$L(\mathbf{x}_{quad}, \dot{\mathbf{x}}_{quad}) = T_{trans} + T_{rot} - U,$$

where $T_{trans} = \frac{m}{2}\dot{\vec{p}}^T\dot{\vec{p}}$ describes the quadrotor translational kinetic energy, $T_{rot} = \frac{1}{2}\vec{\Omega}^T J\vec{\Omega}$ denotes the rotational kinetic energy, $U = -m\vec{g} \cdot \vec{p}$ represents its potential energy, m denotes its mass, $\vec{\Omega} = [\omega_x \ \omega_y \ \omega_z]^T$ introduces the angular velocity vector, J defines the inertia matrix, and $\vec{g} = [0 \ 0 \ -g]^T$ means the acceleration vector due to gravity. The angular velocity vector $\vec{\Omega}$ resolved in the body frame $\mathcal{B}$ is related to the generalized velocities $\dot{\vec{\eta}}$ (in the region where the Euler angles are valid) by means of the standard kinematic relationship $\vec{\Omega} = W_\eta\dot{\vec{\eta}}$, [80]. Therefore, $T_{rot} = \frac{1}{2}\dot{\vec{\eta}}^T\mathbb{J}\dot{\vec{\eta}}$ with $\mathbb{J} = \mathbb{J}(\vec{\eta}) = W_\eta^T J \ W_\eta$ and

$$W_\eta = \begin{bmatrix} -\sin\theta & 0 & 1 \\ \cos\theta\sin\phi & \cos\phi & 0 \\ \cos\theta\cos\phi & -\sin\phi & 0 \end{bmatrix}, \qquad J = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix},$$

where J_{ii} denotes the moment of inertia with respect to the $i = x, y, z$ axes.

Note $\mathbb{J}(\vec{\eta})$ can be used as the inertia matrix for the vehicle rotational dynamics in terms of the Euler angles $\vec{\eta}$. Then, the mathematical equations that represent the dynamics of the aerial vehicle are obtained using the following equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{x}}_{quad}} - \frac{\partial L}{\partial \mathbf{x}_{quad}} = \begin{bmatrix} \vec{F}_p \\ \vec{\tau} \end{bmatrix}, \tag{2.4}$$

where $\vec{F}_p \in \mathbb{R}^3$ symbolizes the total force acting on the vehicle, and $\tau \in \mathbb{R}^3$ represents the torques.

2.2.2 Quadrotor dynamic model

From Figure 2.2 and considering that the thrust force acts only in the z-axis, it yields $\vec{F}_{th} = \vec{n}_z \vec{F}_{th}$, where $\vec{n}_z = [0 \quad 0 \quad 1]^T$ and $F_{th} = ||\vec{F}_{th}||$ represents the thrust directed out of the top of the vehicle. This vector force is represented in the inertial frame using a rotation matrix R derived from the Euler angles as $\vec{F}_{th}^{\mathcal{I}} = R\vec{F}_{th}$, where

$$R = R_{xyz}(\psi, \theta, \phi) \in SO(3) = \begin{bmatrix} C_\psi C_\theta & -S_\psi C_\theta & S_\theta \\ S_\psi C_\phi + C_\psi S_\theta S_\phi & C_\psi C_\phi - S_\psi S_\theta S_\phi & -C_\theta S_\phi \\ S_\psi S_\phi - C_\psi S_\theta C_\phi & C_\psi S_\phi + S_\psi S_\theta C_\phi & C_\theta C_\phi \end{bmatrix}, \quad (2.5)$$

where S_θ and C_θ stand for $\sin(\theta)$ and $\cos(\theta)$ respectively.

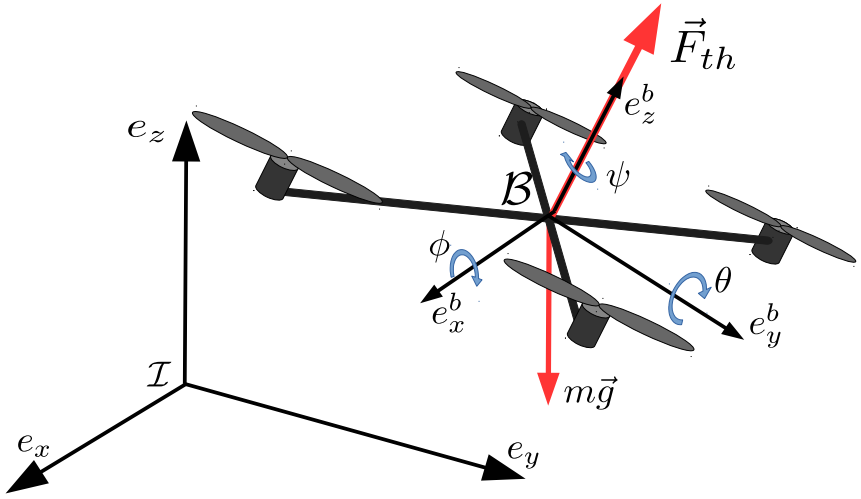


FIGURE 2.2: Quadcopter scheme in an inertial frame.

Developing (2.4), the Euler-Lagrange equation for the translation motion can be written as

$$m\ddot{\vec{p}} - m\vec{g} = \vec{F}_{th}^{\mathcal{I}} \ ,$$

and for the $\vec{\eta}$ coordinates

$$\mathbb{J}\ddot{\vec{\eta}} + \left(\dot{\mathbb{J}} - \frac{1}{2}\frac{\partial}{\partial \vec{\eta}} \left(\dot{\vec{\eta}}^T \mathbb{J} \right) \right) \dot{\vec{\eta}} = \tau \ ,$$

Therefore, rewriting the two previous equations results in

$$m\ddot{\vec{p}} = \vec{F}_{th}^{\mathcal{I}} - mg\vec{n}_z, \quad (2.6)$$

$$\mathbb{J}\ddot{\vec{\eta}} = \vec{\tau} - C(\vec{\eta}, \dot{\vec{\eta}})\dot{\vec{\eta}}. \quad (2.7)$$

where $C(\vec{\eta}, \dot{\vec{\eta}}) = \dot{\mathbb{J}} - \frac{1}{2}\frac{\partial}{\partial \vec{\eta}} \left(\dot{\vec{\eta}}^T \mathbb{J} \right)$ refers to the Coriolis term and contains the gyroscopic and centrifugal terms. Expanding equation (2.7) is not an easy task and in several works the full inertia matrix $\mathbb{J}$ is considered as diagonal and the Coriolis matrix is, in general, neglected.

The Coriolis and the inertial matrix can be obtained from (2.4) for the $\vec{\eta}$ dynamics. Therefore, rewriting the attitude dynamics yields

$$\frac{d}{dt}\left[\vec{\Omega}^T J \frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}}\right] - \vec{\Omega}^T J \frac{\partial \vec{\Omega}}{\partial \vec{\eta}} = \tau,$$

then $\frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}} = W_\eta$, thus, $\vec{\Omega}^T J \frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$ with

$$\begin{aligned}
b_1 &= -J_{xx}(\dot{\phi} S_\theta - \dot{\psi} S_\theta^2) + J_{yy}(\dot{\theta} C_\theta S_\phi C_\phi + \dot{\psi} C_\theta^2 S_\phi^2) + J_{zz}(\dot{\psi} C_\theta^2 C_\phi^2 - \dot{\theta} C_\theta S_\phi C_\phi), \\
b_2 &= J_{yy}(\dot{\theta} C_\phi^2 + \dot{\psi} C_\theta S_\phi C_\phi) - J_{zz}(\dot{\psi} C_\theta S_\phi C_\phi - \dot{\theta} S_\phi^2), \\
b_3 &= J_{xx}(\dot{\phi} - \dot{\psi} S_\theta).
\end{aligned}$$

Applying $\frac{d}{dt}\left(\vec{\Omega}^T J \frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}}\right)$,

$$\begin{aligned}
\dot{b}_1 &= -J_{xx}(\ddot{\phi} S_\theta + \dot{\phi}\dot{\theta} C_\theta - \ddot{\psi} S_\theta^2 - 2\dot{\psi}\dot{\theta} S_\theta C_\theta) + J_{yy}(\ddot{\theta} C_\theta S_\phi C_\phi - \dot{\theta}^2 S_\theta S_\phi C_\phi \\
&\quad -\dot{\theta}\dot{\phi} C_\theta S_\phi^2 + \dot{\theta}\dot{\phi} C_\theta C_\phi^2 + \ddot{\psi} C_\theta^2 S_\phi^2 - 2\dot{\psi}\dot{\theta} S_\theta C_\theta S_\phi^2 + 2\dot{\psi}\dot{\phi} C_\theta^2 S_\phi C_\phi) + J_{zz}(\ddot{\psi} C_\theta^2 C_\phi^2 \\
&\quad -2\dot{\psi}\dot{\theta} S_\theta C_\theta C_\phi^2 - 2\dot{\psi}\dot{\phi} C_\theta^2 S_\phi C_\phi - \ddot{\theta} C_\theta s\phi C_\phi + \dot{\theta}^2 S_\theta S_\phi C_\phi + \dot{\theta}\dot{\phi} C_\theta C_\phi^2 - \dot{\theta}\dot{\phi} C_\theta C_\phi^2), \\
\dot{b}_2 &= J_{yy}(\ddot{\theta} C_\phi^2 - 2\dot{\theta}\dot{\phi} S_\phi C_\phi + \ddot{\psi} C_\theta S_\phi C_\phi - \dot{\psi}\dot{\theta} S_\theta S_\phi C_\phi + \dot{\psi}\dot{\phi} C_\theta C_\phi^2 - \dot{\psi}\dot{\phi} C_\theta S_\phi^2) \\
&\quad -J_{zz}(\ddot{\psi} C_\theta S_\phi C_\phi - \dot{\psi}\dot{\theta} S_\theta S_\phi C_\phi - \dot{\psi}\dot{\phi} C_\theta S_\phi^2 + \dot{\psi}\dot{\phi} C_\theta C_\phi^2 - \ddot{\theta} S_\phi^2 - 2\dot{\theta}\dot{\phi} S_\phi C_\phi), \\
\dot{b}_3 &= J_{xx}(\ddot{\phi} - \ddot{\psi} S_\theta - \dot{\psi}\dot{\theta} C_\theta).
\end{aligned}$$

Similarly,

$$\frac{\partial \vec{\Omega}}{\partial \vec{\eta}} = \begin{bmatrix} 0 & -\dot{\psi} C_\theta & 0 \\ 0 & -\dot{\psi} S_\theta S_\phi & -\dot{\theta} S_\phi + \dot{\psi} C_\theta C_\phi \\ 0 & -\dot{\psi} S_\theta C_\phi & -\dot{\psi} C_\theta S_\phi - \dot{\theta} C_\phi \end{bmatrix},$$

then $\vec{\Omega}^T J \frac{\partial \vec{\Omega}}{\partial \vec{\eta}} = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$, where

$$\begin{aligned}
h_1 &= 0 \ , \\
h_2 &= -J_{xx}(\dot{\psi}\dot{\phi} C_\theta - \dot{\psi}^2 S_\theta C_\theta) - J_{yy}(\dot{\psi}\dot{\theta} S_\theta S_\phi C_\phi + \dot{\psi}^2 S_\theta C_\theta S_\phi^2) \ , \\
&\quad -J_{zz}(\dot{\psi}^2 S_\theta C_\theta C_\phi^2 - \dot{\psi}\dot{\theta} S_\theta S_\phi C_\phi) \\
h_3 &= J_{yy}(-\dot{\theta}^2 S_\phi C_\phi - \dot{\psi}\dot{\theta} C_\theta S_\phi^2 + \dot{\psi}\dot{\theta} C_\theta C_\phi^2 + \dot{\psi}^2 C_\theta^2 S_\phi C_\phi) \\
&\quad J_{zz}(-\dot{\psi}^2 C_\theta^2 S_\phi C_\phi + \dot{\psi}\dot{\theta} C_\theta S_\phi^2 - \dot{\psi}\dot{\theta} C_\theta C_\phi^2 + \dot{\theta}^2 S_\phi C_\phi) \ ,
\end{aligned}$$

such that

$$\begin{bmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} \dot{b}_1 - h_1 \\ \dot{b}_2 - h_2 \\ \dot{b}_3 - h_3 \end{bmatrix}.$$

Thus, grouping terms and using (2.7), it follows that for $\mathbb{J}$

$$\mathbb{J}(\vec{\eta}) = \begin{bmatrix} J_{xx}s^2\theta + J_{yy}c^2\theta s^2\phi + J_{zz}c^2\theta c^2\phi & c\theta c\phi s\phi(J_{yy} - J_{zz}) & -J_{xx}s\theta \\ c\theta c\phi s\phi(J_{yy} - J_{zz}) & J_{yy}c^2\phi + J_{zz}s^2\phi & 0 \\ -J_{xx}s\theta & 0 & J_{xx} \end{bmatrix}, \quad (2.8)$$

and

$$C(\vec{\eta}, \dot{\vec{\eta}}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix},$$

where

$$\begin{aligned}
c_{11} &= J_{xx}\dot{\theta}S_\theta C_\theta + J_{yy}(-\dot{\theta}S_\theta C_\theta S_\phi^2 + \dot{\phi}C_\theta^2 S_\phi C_\phi) - J_{zz}(\dot{\theta}S_\theta C_\theta C_\phi^2 + \dot{\phi}C_\theta^2 S_\phi C_\phi), \\
c_{12} &= J_{xx}\dot{\psi}S_\theta C_\theta - J_{yy}(\dot{\theta}S_\theta S_\phi C_\phi + \dot{\phi}C_\theta S_\phi^2 - \dot{\phi}C_\theta C_\phi^2 + \dot{\psi}S_\theta C_\theta S_\phi^2) \\
&\quad + J_{zz}(\dot{\phi}C_\theta S_\phi^2 - \dot{\phi}C_\theta C_\phi^2 - \dot{\psi}S_\theta C_\theta C_\phi^2 + \dot{\theta}S_\theta S_\phi C_\phi), \\
c_{13} &= -J_{xx}\dot{\theta}C_\theta + J_{yy}\dot{\psi}C_\theta^2 S_\phi C_\phi - J_{zz}\dot{\psi}C_\theta^2 S_\phi C_\phi, \\
c_{21} &= -J_{xx}\dot{\psi}S_\theta C_\theta + J_{yy}\dot{\psi}S_\theta C_\theta S_\phi^2 + J_{zz}\dot{\psi}S_\theta C_\theta C_\phi^2, \\
c_{22} &= -J_{yy}\dot{\phi}S_\phi C_\phi + J_{zz}\dot{\phi}S_\phi C_\phi, \\
c_{23} &= J_{xx}\dot{\psi}C_\theta + J_{yy}(-\dot{\theta}S_\phi C_\phi + \dot{\psi}C_\theta C_\phi^2 - \dot{\psi}C_\theta S_\phi^2) + J_{zz}(\dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2 + \dot{\theta}S_\phi C_\phi), \\
c_{31} &= -J_{yy}\dot{\psi}C_\theta^2 S_\phi C_\phi + J_{zz}\dot{\psi}C_\theta^2 S_\phi C_\phi, \\
c_{32} &= -J_{xx}\dot{\psi}C_\theta + J_{yy}(\dot{\theta}S_\phi C_\phi + \dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2) - J_{zz}(\dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2 + \dot{\theta}S_\phi C_\phi), \\
c_{33} &= 0.
\end{aligned}$$

Although (2.6) and (2.7) were developed taking in mind a multicopter with four rotors, these equations are also valid for other aerial configurations as long as the forces and torques are rewritten.

2.2.3 Newton-Euler Methodology

The general mathematical model describing the dynamics of an aircraft evolving in a three-dimensional space is obtained by representing the aircraft as a solid body, which is subject to non-conservative forces $\vec{F}_{\mathcal{I}} \in \mathbb{R}^3$ expressed in an inertial frame $\mathcal{I}$, and torques $\vec{\tau} \in \mathbb{R}^3$ applied to its center of mass and specified with respect to the body frame $\mathcal{B}$, and by using the Newton-Euler approach [22, 84, 85, 86]

$$m\ddot{\vec{p}} = \vec{F}_{\mathcal{I}}, \quad (2.9)$$

$$\dot{R} = R\hat{\Omega}, \quad (2.10)$$

$$J\dot{\vec{\Omega}} = -\vec{\Omega} \times J\vec{\Omega} + \vec{\tau}, \quad (2.11)$$

where $\hat{\Omega}$ describes the anti-symmetric matrix of $\vec{\Omega}$ and J represents the constant inertia matrix around the center of mass.

2.2.4 Quadrotor dynamic model

Consider the aerial vehicle from Figure 2.2, it can be concluded that

$$\vec{F}_{\mathcal{I}} = \begin{bmatrix} C_\psi C_\theta & -S_\psi C_\theta & S_\theta \\ S_\psi C_\phi + C_\psi S_\theta S_\phi & C_\psi C_\phi - S_\psi S_\theta S_\phi & -C_\theta S_\phi \\ S_\psi S_\phi - C_\psi S_\theta C_\phi & C_\psi S_\phi + S_\psi S_\theta C_\phi & C_\theta C_\phi \end{bmatrix} \vec{F}_{th} + \vec{g}, \tag{2.12}$$

where $\vec{g} = [0\ 0\ \ -g]^T$ denotes the gravity acceleration vector. The main vector force produced by the rotors is considered as $\vec{F}_{th} = [0\ 0\ F_{th}]^T$.

Considering the total forces and torques from (2.3), and introducing them into (2.9) and into (2.11), it follows that

$$\begin{aligned} m\ddot{x} &= -\sin\theta F_{th}, \\ m\ddot{y} &= \cos\theta\sin\phi F_{th}, \\ m\ddot{z} &= \cos\theta\cos\phi F_{th} - mg, \end{aligned} \tag{2.13}$$

$$\begin{bmatrix} \ddot{\psi} \\ \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = J^{-1} \left(\begin{bmatrix} \tau_z \\ \tau_y \\ \tau_x \end{bmatrix} - \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} \right). \tag{2.14}$$

Equation (2.13) represents the translational dynamics of the aerial vehicle while (2.14) introduces its rotational dynamics.

2.3 Quadrotor Quaternion Modeling

Quaternions were proposed by Hamilton in the nineteenth century as a three-dimensional version of complex numbers (represented as one real and one imaginary part) [42]. They are also known as "hypercomplex" numbers (the hypercomplex space is noted as $\mathbb{H}$) since they can be represented as one real plus three imaginary numbers as $\boldsymbol{q} \triangleq q_0 + q_1\hat{\imath} + q_2\hat{\jmath} + q_3\hat{k}$, where $\hat{\imath}, \hat{\jmath}, \hat{k} \in \mathbb{I}$, such that $\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = \hat{\imath}\hat{\jmath}\hat{k} = -1$, and $q_0, \cdots, q_3 \in \mathbb{R}$. Another common representation of a quaternion is using one scalar number and a vector as $\boldsymbol{q} \triangleq q_0 + \vec{q}$, with $\vec{q} = [q_1\ q_2\ q_3]^T$.

Due to its mathematical and geometrical advantages, this notation will be used throughout this chapter. Since the space of three dimensional vectors is included in the quaternion space, then vectors can be treated as quaternions with a scalar part equal to zero in all of the quaternion operations.

Consider the rotation illustrated in Figure 2.3 as vector $\vec{\vartheta} = [\vartheta_x\ \vartheta_y\ \vartheta_z]^T$ with magnitude $\vartheta = ||\vec{\vartheta}||$ in radians, acting on an axis represented as a unitary vector $\vec{u} = \vec{\vartheta}/||\vec{\vartheta}||$, the axis-angle representation of this rotation is denoted as $\vec{\vartheta} = \vartheta\vec{u}$.

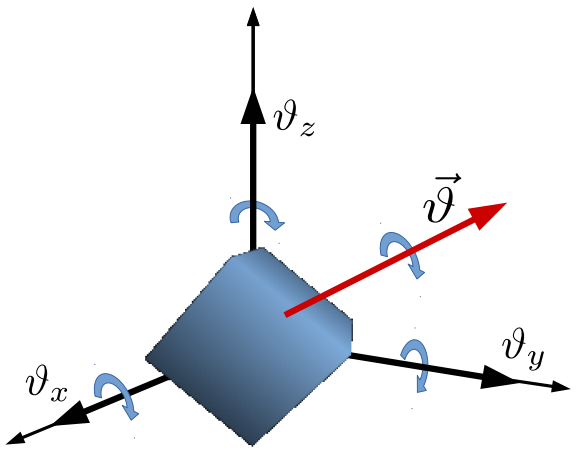


FIGURE 2.3: Axis-angle representation of a rigid body rotation

It is widely known that a simple rotation, with magnitude φ in radians, over a plane can be represented using the Euler Formula as $e^{\hat{\imath}\varphi} = \cos\varphi + \hat{\imath}\sin\varphi$ [43]. In the 19th century, a French banker named Olinde Rodrigues expanded the Euler Formula to include 3-dimensional rotations using quaternions. This expression, known as the Euler-Rodrigues formula is the exponential mapping of the axis-angle representation of a rotation, defined as

$$\boldsymbol{q} = e^{\frac{1}{2}\vartheta\vec{u}} = \cos(\vartheta/2) + \vec{u}\sin(\vartheta/2), \tag{2.15}$$

notice that $||\boldsymbol{q}|| = 1$, thus $\boldsymbol{q}$ can be called a *unit quaternion*.

Inversely, the axis-angle representation of a rotation can be derived from a quaternion using the logarithmic mapping

$$\vec{\vartheta} = 2\ln\boldsymbol{q}, \tag{2.16}$$

with

$$\ln\boldsymbol{q} := \begin{cases} [0\ 0\ 0]^T & \text{, if} \quad ||\vec{q}|| = 0, \\ \dfrac{\vec{q}}{||\vec{q}||}\arccos q_0 & \text{, if} \quad ||\vec{q}|| \neq 0. \end{cases} \tag{2.17}$$

2.3.1 Quaternion algebra

Product: Because of its significance, historically as well as in the definition of the quaternion space, the main operation of quaternions is the multiplication. Other operations and properties arise from this definition, like the conjugate and the norm. Considering $\boldsymbol{q}, \boldsymbol{r} \in \mathbb{H}$, the quaternion operations are defined as:

$$\boldsymbol{q} \otimes \boldsymbol{r} := (q_0 r_0 - \vec{q}\cdot\vec{r}) + (q_0\,\vec{r} + r_0\,\vec{q} + \vec{q}\times\vec{r})\ . \tag{2.18}$$

It is noteworthy that quaternion product is not commutative, which means that $\boldsymbol{q}\otimes\boldsymbol{r} \neq \boldsymbol{r}\otimes\boldsymbol{q}$. This is because of the same non-commutativity property of the cross product used in the definition.

Quaternion double cover: Due to the geometrical properties of the Euler-Rodrigues formula, every possible quaternion has a negative counterpart, which corresponds to an equivalent rotation, but adding a full 2π turn through the same axis. Let $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ be two quaternions such that

$$\begin{aligned} \boldsymbol{q}_1 &:= \cos(\vartheta/2) + \vec{u}\sin(\vartheta/2), \\ \boldsymbol{q}_2 &:= \cos((\vartheta + 2\pi)/2) + \vec{u}\sin((\vartheta + 2\pi)/2), \end{aligned} \tag{2.19}$$

then, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ represent the same orientation, with the difference that $\boldsymbol{q}_2$ takes an additional 2π rotation around $\vec{u}$, furthermore, $\boldsymbol{q}_1 = -\boldsymbol{q}_2$.

Sum: The sum of quaternions is simply defined as the sum of each of its elements,

$$\boldsymbol{q} + \boldsymbol{r} := q_0 + r_0 + \vec{q} + \vec{r}. \tag{2.20}$$

The set of all quaternions with operations addition and multiplication defines a non-commutative division ring. See [45] for more information on this matter.

Conjugate: The conjugate of a unit quaternion expresses an inverse rotation over the same axis, and is defined as

$$\boldsymbol{q}^* := q_0 - \vec{q}\ , \tag{2.21}$$

while the conjugate of a product of quaternions is

$$(\boldsymbol{q} \otimes \boldsymbol{r})^* = \boldsymbol{r}^* \otimes \boldsymbol{q}^*, \tag{2.22}$$

which can be proved by expanding the corresponding products.

Norm: The norm of a quaternion is defined by

$$||\boldsymbol{q}||^2 := \boldsymbol{q} \otimes \boldsymbol{q}^* = q_0^2 + q_1^2 + q_2^2 + q_3^2\ . \tag{2.23}$$

Inverse: The quaternion product forms a closed-loop group. That is, the product of two non-null quaternions is another quaternion. This means that for any non-null quaternion there exists an inverse quaternion such that

$$\begin{aligned} &\boldsymbol{q}^{-1} := \frac{\boldsymbol{q}^*}{||\boldsymbol{q}||}, \\ &\boldsymbol{q} \otimes \boldsymbol{q}^{-1} = \boldsymbol{q}^{-1} \otimes \boldsymbol{q} = 1 + [\,0\ \ 0\ \ 0\,]^T. \end{aligned} \tag{2.24}$$

Vector Rotation: Considering $\vec{p} \in \mathbb{R}^3$ as a 3D vector in a first reference frame (the earth coordinates example), and $\vec{p}'$ as the same vector but now with respect to a new reference frame (for example, a vehicle's moving coordinates), then $\vec{p}$ can be transformed into $\vec{p}'$ using a double quaternion product as

$$\vec{p}' = \boldsymbol{q}^{-1} \otimes \vec{p} \otimes \boldsymbol{q} = \boldsymbol{q}^* \otimes \vec{p} \otimes \boldsymbol{q}, \tag{2.25}$$

where the quaternion $\boldsymbol{q}$ represents the rotation of the second reference frame with respect to the first one. Note this rotation does not affect the vector's magnitude.

Derivative: The derivative of any unit quaternion can be obtained by differentiating (2.25) as

$$\dot{\vec{p}}' = \dot{\boldsymbol{q}}^{-1} \otimes \vec{p} \otimes \boldsymbol{q} + \boldsymbol{q}^{-1} \otimes \vec{p} \otimes \dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q} \otimes \vec{p}' + \vec{p}' \otimes \boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}}. \tag{2.26}$$

Since $\boldsymbol{q}$ is an unit quaternion, then $\boldsymbol{q}^{-1} \otimes \boldsymbol{q} = 1$ and $\dot{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q} + \boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}}$, which yields

$$\dot{\vec{p}}' = \vec{p}' \otimes \boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}} - \boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}} \otimes \vec{p}' = 2(\boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}}) \times \vec{p}'. \tag{2.27}$$

Note $\dot{\vec{p}}'$ is the translational velocity of the vector, $\dot{\vec{p}}' = \vec{\Omega} \times \vec{p}'$, where $\vec{\Omega}$ is the rotational velocity of $\vec{p}'$, thus

$$\vec{\Omega} \times \vec{p}' = 2(\boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}}) \times \vec{p}', \tag{2.28}$$

which can be reduced to

$$\vec{\Omega} = 2(\boldsymbol{q}^{-1} \otimes \dot{\boldsymbol{q}}) \Rightarrow \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega}. \tag{2.29}$$

2.3.2 Rigid Body Dynamic Modeling

The translational and rotational state of any given rigid body can also be expressed as a tuple $\mathbf{x} := \begin{bmatrix}\vec{p} & \dot{\vec{p}} & \boldsymbol{q} & \vec{\Omega}\end{bmatrix}^T$ where $\vec{p} \in \mathrm{R}^3$ symbolizes the body position in the inertial frame $\mathcal{I}$, $\dot{\vec{p}}$ its velocity, $\boldsymbol{q} = q_0 + \begin{bmatrix}q_1 & q_2 & q_3\end{bmatrix}^T$ defines the vehicle orientation with respect to $\mathcal{I}$, represented as a unit quaternion and $\vec{\Omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ represents the rotational velocity in the moving reference frame $\mathcal{B}$ located the body's center of mass. Therefore, following Newton's equations of motion, the dynamic model of any rigid body expressed with unit quaternions is

$$\dot{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix}\vec{p} \\ \dot{\vec{p}} \\ \boldsymbol{q} \\ \vec{\Omega}\end{bmatrix} = \begin{bmatrix}\dot{\vec{p}} \\ m^{-1}\vec{F}_{\mathcal{I}} \\ \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} \\ J^{-1}\left(\vec{\tau} - \vec{\Omega} \times J\vec{\Omega}\right)\end{bmatrix}, \tag{2.30}$$

where J is the inertia matrix, $\vec{\tau}$ represents the total torque, both with respect to $\mathcal{B}$, and $\vec{F}_{\mathcal{I}}$ defines the total force applied to the body in the $\mathcal{I}$. Equation (2.30) can be used to describe any mechanical system including aerial vehicles. In the next subsection the dynamics of a quadrotor will be represented using this approach.

2.3.3 Quadrotor quaternion dynamical model

If the mechanical configuration is considered to be symmetric, and some effects as blade flapping and the misalignment on the motors' axes could be considered small enough, it can then be assumed that the forces and torques which act on the vehicle are only the ones illustrated on Figure 2.4.

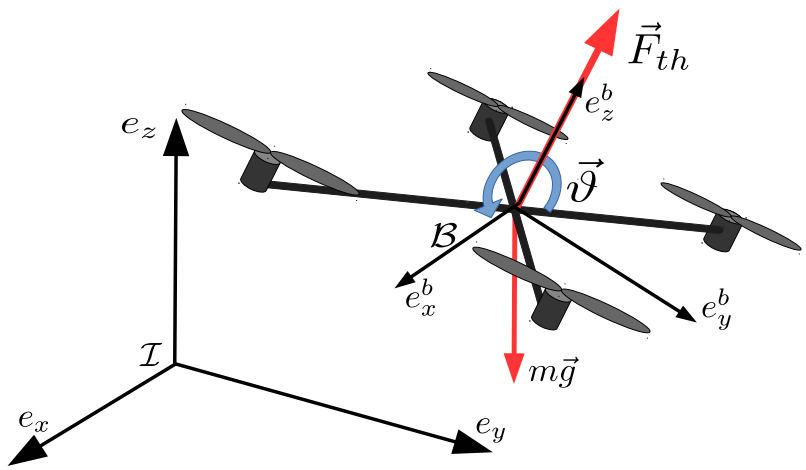


FIGURE 2.4: Quadrotor free body diagram

Here, two reference frames are considered. $\mathcal{I} = [\ e_x \quad e_y \quad e_z\ ]^T$ defines the fixed inertial coordinates, and $\mathcal{B} = [\ e_x^b \quad e_y^b \quad e_z^b\ ]^T$ represents the moving body frame located on the vehicle's center of mass, and $\boldsymbol{q} = e^{\frac{1}{2}\vec{\vartheta}}$ denotes a quaternion rotation from $\mathcal{I}$ to $\mathcal{B}$.

Let $\vec{F}_{th}$ and $\vec{\tau}$ symbolize the total thrust force, and torque from (2.3), since $\vec{F}_{th}$ and $\vec{\tau}$ act on $\mathcal{B}$, a quaternion rotation using (2.25) is applied to express the vehicle dynamics as

$$\dot{\mathbf{x}}_{quad} = \frac{d}{dt}\begin{bmatrix} \vec{p} \\ \dot{\vec{p}} \\ \boldsymbol{q} \\ \vec{\Omega} \end{bmatrix} = \begin{bmatrix} \dot{\vec{p}} \\ \boldsymbol{q} \otimes \frac{\vec{F}_{th}}{m} \otimes \boldsymbol{q}^* + \vec{g} \\ \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} \\ J^{-1}\left(\vec{\tau} - \vec{\Omega} \times J\,\vec{\Omega}\right) \end{bmatrix}. \tag{2.31}$$

Note from (2.31) that the quadrotor's rotational and translational dynamics are coupled, due to the orientation of $\vec{F}_{th}^{\mathcal{I}}$ depending on the vehicle's attitude $\boldsymbol{q}$. Nevertheless, using an appropriate approach and some properties of unit quaternions, the quadrotor can be easily stabilized despite its underactuated nature.

2.3.4 Decoupling the vehicle dynamics

Since the attitude sub-system of the quadrotor is completely actuated, we address it in this subsection.

2.3.4.1 Quaternion Rotational Model

From (2.31), the rotational dynamics of a quadrotor can be written as

$$\dot{\mathbf{x}}_r = \frac{d}{dt}\begin{bmatrix} \boldsymbol{q} \\ \vec{\Omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} \\ J^{-1}\left(\vec{\tau} - \vec{\Omega} \times J\,\vec{\Omega}\right) \end{bmatrix}, \tag{2.32}$$

If (2.32) is stabilized using a control action applied by $\vec{\tau}$, then the quaternion attitude will converge to $\boldsymbol{q}_0 = 1 + [0\ 0\ 0]^T$ while the axis-angle orientation $\vec{\vartheta}$ and its angular velocity $\vec{\Omega}$ will converge to zero.

Given a desired attitude trajectory defined by a quaternion $\boldsymbol{q}_d$ and its angular velocity $\vec{\Omega}_d$, (2.32) can be defined in terms of the error quaternion $\boldsymbol{q}_e \triangleq \boldsymbol{q}_d^* \otimes \boldsymbol{q}$ as

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{q}_e \\ \vec{\Omega}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\boldsymbol{q}_e \otimes \vec{\Omega}_e \\ J^{-1}\left(\vec{\tau} - \vec{\Omega}_e \times J\,\vec{\Omega}_e\right) \end{bmatrix}, \tag{2.33}$$

if τ_u is correctly designed in terms of the attitude error. Then $\boldsymbol{q}_e \to \boldsymbol{q}_0$, implying $\boldsymbol{q} \to \boldsymbol{q}_d$.

2.3.4.2 Quadrotor Translational Model

From (2.31) and defining $\vec{F}_{th}^{\mathcal{I}} = \boldsymbol{q} \otimes \vec{F}_{th} \otimes \boldsymbol{q}^*$, the translational dynamics are given by

$$\dot{\mathbf{x}}_t = \frac{d}{dt}\begin{bmatrix} \vec{p} \\ \dot{\vec{p}} \end{bmatrix} = \begin{bmatrix} \dot{\vec{p}} \\ m^{-1}\vec{F}_{th}^{\mathcal{I}} + \vec{g} \end{bmatrix}. \tag{2.34}$$

From (2.34), a desired force $\vec{F}_{th}^{\mathcal{I}}$ can easily be designed such that $\mathbf{x}_t$ and $\dot{\mathbf{x}}_t$ converge to zero. If a position error is defined as $\vec{p}_e = \vec{p} - \vec{p}_d$, where $\vec{p}_d$ represents a desired position for the UAV, then the translational error dynamics can be written as

$$\frac{d}{dt}\begin{bmatrix} \vec{p}_e \\ \dot{\vec{p}}_e \end{bmatrix} = \begin{bmatrix} \dot{\vec{p}}_e \\ m^{-1}\vec{F}_{th}^{\mathcal{I}} + \vec{g} \end{bmatrix}. \tag{2.35}$$

Consequently, if an adequate controller is designed for $\vec{F}_{th}^{\mathcal{I}}$ the position error will converge to zero, meaning the quadrotor can be stabilized in any desired position.

2.3.5 Coupled Dynamics

Analyzing (2.35), it yields that the translational model can be seen as a fully actuated system, in which $\vec{F}_{th}^{\mathcal{I}}$ can be designed to point at any direction. But considering the complete model of the quadrotor, the force that the propellers really exert depends on the attitude subsystem as seen in (2.31).

Define a desired force $\vec{F}_u \in \mathbb{R}^3$ w.r.t. $\mathcal{I}$ which stabilizes system (2.35), given the direction and magnitude of such force, the attitude can be controlled using τ such that the direction of $\vec{F}_{th}^{\mathcal{I}}$ is aligned with $\vec{F}_u$, thus orientating the quadrotor thrust in the direction required to control the translational dynamics.

This quaternion is derived from the shortest rotation between both vectors, and represented by $\boldsymbol{q}_d$. Two equivalent methods can be used to compute it.

Quaternion trajectory (option 1): Recalling the Euler-Rodrigues formula from (2.15), $\boldsymbol{q}_d$ is defined as

$$\boldsymbol{q}_d = e^{\frac{1}{2}\vartheta_d \vec{u}_d} = \cos\left(\frac{\vartheta_d}{2}\right) + \vec{u}_d \sin\left(\frac{\vartheta_d}{2}\right), \tag{2.36}$$

where $\vec{u}_d$ and ϑ_d denote respectively the axis and the angle of the shortest rotation between $\vec{F}_{th}$ and $\vec{F}_u$. Defining $\vec{n}_z$ and $\vec{n}_u$ as the normalized vectors of $\vec{F}_{th}$ and $\vec{F}_u$ respectively (note $\vec{n}_z = [0\ 0\ 1]^T$ is constant), the cross product between these vectors is defined as

$$\vec{n}_u \times \vec{n}_z = \vec{u}_d \sin(\vartheta_d), \tag{2.37}$$

while the scalar product is given by

$$\vec{n}_u \cdot \vec{n}_z = \cos(\vartheta_d). \tag{2.38}$$

Some known trigonometric functions can be used for cutting the rotation in half as

$$\cos\left(\frac{\vartheta_d}{2}\right) = \pm\sqrt{\frac{1+\cos(\vartheta_d)}{2}} \quad , \quad \sin\left(\frac{\vartheta_d}{2}\right) = \pm\sqrt{\frac{1-\cos(\vartheta_d)}{2}}. \tag{2.39}$$

Then the attitude that aligns the thrust to the controller direction can be designed using trigonometric identities and the Euler-Rodrigues formula as

$$\begin{aligned} \boldsymbol{q}_t &:= \pm\left(\sqrt{\frac{1+\vec{n}_u\cdot\vec{n}_z}{2}} + \frac{\vec{n}_u\times\vec{n}_z}{||\vec{n}_u\times\vec{n}_z||}\sqrt{\frac{1-\vec{n}_u\cdot\vec{n}_z}{2}}\right) \\ F_{th} &:= \left|\left|\vec{F}_u\right|\right|. \end{aligned} \tag{2.40}$$

Since $\vec{n}_z$ is always aligned with the vertical axis of the body frame $\mathcal{B}$, and the direction of (2.40) is defined by a cross product, then $\boldsymbol{q}_t$ will only rotate the vehicle in the xy plane.

An additional rotation around the z axis can be added by introducing

$$\boldsymbol{q}_d := \boldsymbol{q_t} \otimes \boldsymbol{q}_z, \tag{2.41}$$

where $\boldsymbol{q}_z$ denotes the desired rotation over the z axis. Figure 2.5 illustrates the behavior of the aforementioned rotations.

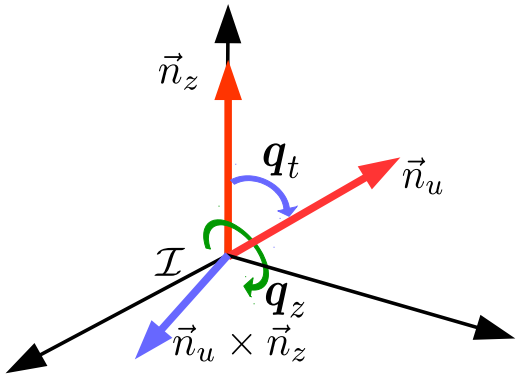


FIGURE 2.5: Rotation that aligns the thrust vector with the desired position control force, with an additional rotation in the z axis.

Quaternion trajectory (option 2): From the definition of the quaternion product, and treating $\vec{n}_u$ and $\vec{n}_z$ as quaternions with zero-value scalar parts, (2.37) and (2.38) can be combined as

$$\vec{n}_u \otimes \vec{n}_z^* = -\vec{n}_u \cdot \vec{n}_z^* + \vec{n}_u \times \vec{n}_z^* = \vec{n}_z \cdot \vec{n}_u + \vec{n}_z \times \vec{n}_u = \cos(\vartheta_d) + \vec{u}_d \sin(\vartheta_d) \ . \quad (2.42)$$

Since (2.42) expresses twice the desired rotation needed in (2.36), exponential and logarithmic properties are then applied, thus resulting in

$$\boldsymbol{q}_t = e^{\dfrac{\ln(\vec{n}_u \otimes \vec{n}_z^*)}{2}} \ . \quad (2.43)$$

Note since $\vec{n}_z$ only acts in the vertical axis of the quadrotor, then $\boldsymbol{q}_d$ will only compute rotations around the xy plane of the inertial frame. Considering ψ_d as a desired rotation around the z axis of the vehicle's body frame, the desired quaternion can be enhanced as

$$\boldsymbol{q}_d = e^{\dfrac{\ln(\vec{n}_u \otimes \vec{n}_z^*)}{2}} \otimes \boldsymbol{q}_z = e^{\dfrac{\ln(\vec{n}_u \otimes \vec{n}_z^*)}{2}} \otimes e^{\dfrac{[0\ 0\ \psi_d]^T}{2}} \ . \quad (2.44)$$

Introducing (2.44) into the rotational error dynamic model from (2.33), and if τ is designed such that $\boldsymbol{q} \to \boldsymbol{q}_d$, then it implies that $\vec{F}_{th}^{\mathcal{I}} \to \vec{F}_u$ such that system (2.35) can be stabilized if $\vec{F}_u$ is correctly designed.

2.4 Dual Quaternion Modeling

The study of translational and rotational dynamics using simple unit quaternions usually do so by separating the position and rotation states as in the previous sections. This is very common when dealing with relatively simple systems such as quadrotors and other similar types of aircraft, however, mathematical models can get very complicated when multiple rotations and translations are involved. Such case could occur, for example, in digital animations, multi-agent robotic systems, and autonomous aerial manipulators.

Over many years, researchers came with the idea of expanding the concept of the quaternion to include more complex geometrical transformations. Such ideas started developing in the realm of non-euclidean geometry, analysis, and topology. In 1873, an English geometer named William Kingdon Clifford made a first sketch of an expansion of Hamilton's quaternions, by adding a second quaternion which is multiplied by a new imaginary term, he gave this concept the name of *biquaternion* [87], [88]. Later on, Alexander McAulay introduced in 1898 a *nullipotent term* ϵ to the biquaternion. This term can be defined in any way, as long as it respects the property

$$\epsilon \neq 0 \quad | \quad \epsilon^2 = 0, \tag{2.45}$$

with this component, McAulay developed dual quaternion algebra, using the term of *octonions*. Finally, in 1985, Aleksandr Kotelnikov studied the applications of octonions and biquaternions on kinematics [89], concluding that the properties of unit quaternions are also valid for those numbers, hence, they were eventually known as *Dual Quaternions*, which are defined as

$$\hat{\boldsymbol{q}} = \boldsymbol{q}_R + \boldsymbol{q}_D\epsilon, \quad \epsilon \neq 0, \epsilon^2 = 0, \tag{2.46}$$

where $\boldsymbol{q}_R$, $\boldsymbol{q}_D \in \mathbb{H}$ are known as the real and dual parts of $\hat{\boldsymbol{q}}$ respectively.

2.4.1 Dual quaternion operations

Some of the most important operations for dual quaternions are:

Sum: Let $\hat{\boldsymbol{q}}_1$ and $\hat{\boldsymbol{q}}_2$ be dual quaternions, then:

$$\hat{\boldsymbol{q}}_1 + \hat{\boldsymbol{q}}_2 = \boldsymbol{q}_{1R} + \boldsymbol{q}_{2R} + [\boldsymbol{q}_{1D} + \boldsymbol{q}_{2D}]\,\epsilon. \tag{2.47}$$

Product: The multiplication between dual quaternions is defined as:

$$\hat{\boldsymbol{q}}_1 \otimes \hat{\boldsymbol{q}}_2 = \boldsymbol{q}_{1R} \otimes \boldsymbol{q}_{2R} + [\boldsymbol{q}_{1R} \otimes \boldsymbol{q}_{2D} + \boldsymbol{q}_{1D} \otimes \boldsymbol{q}_{2R}]\,\epsilon. \tag{2.48}$$

Norm: The norm of a dual quaternion is defined as:

$$||\hat{\boldsymbol{q}}||^2 = \hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}}^*. \tag{2.49}$$

Note that if $||\hat{\boldsymbol{q}}||^2 = 1 + 0\epsilon$, then $\hat{\boldsymbol{q}}$ is called a *unit dual quaternion*.

Conjugation: The conjugation of a dual quaternion is defined as:

$$\hat{\boldsymbol{q}}^* = \boldsymbol{q}_R^* - \boldsymbol{q}_D^*\epsilon. \tag{2.50}$$

Since in this work we are dealing only with unit dual quaternions, then we can say that $\hat{\boldsymbol{q}}^* = \hat{\boldsymbol{q}}^{-1}$

Dual quaternion logarithm: A dual quaternion can be transformed by $\ln \hat{\boldsymbol{q}} = \frac{1}{2}(\vec{\vartheta} + \vec{p}_{\mathcal{B}}\epsilon)$, where $\vec{\vartheta} = 2 \ln \boldsymbol{q}$ represents the body's rotation given by a unit quaternion logarithmic mapping and $\vec{p}_{\mathcal{B}}$ denotes the position vector in the body frame.

This yields a relationship between a dual quaternion and the screw representation of simultaneous rotation and translation, illustrated in Figure 2.6, and defined as

$$2 \ln \hat{\boldsymbol{q}} = \vec{\vartheta} + \vec{p}_{\mathcal{B}}\, \epsilon \ . \tag{2.51}$$

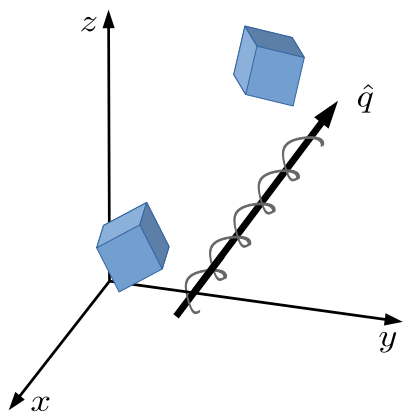


FIGURE 2.6: Simultaneous rotation and translation of a rigid body.

Multiple Dual Quaternion Transformation Some robots involve successions of rotations and translations, such as aerial manipulators (a quadrotor provided with an attached robotic arm). Dual quaternions become useful in this case. For example, in Figure 2.7, $\hat{\boldsymbol{q}}_1$ represents the transformation from reference frame R_1 to R_2, $\hat{\boldsymbol{q}}_2$ denotes the relative transformation from R_2 to R_3, and $\hat{\boldsymbol{q}}_3$ defines the total transformation from R_1 to R_3.

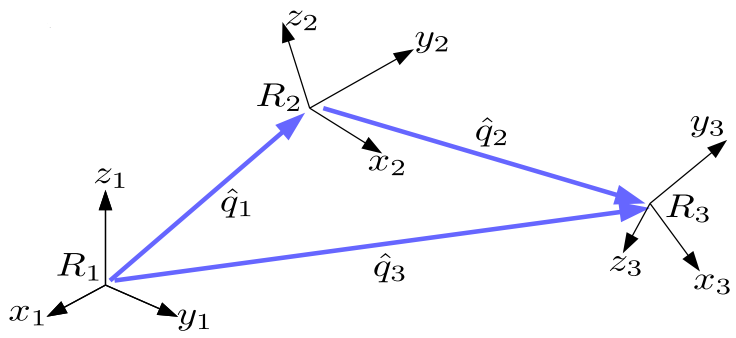


FIGURE 2.7: Simultaneous dual quaternion transformations can be expressed as a one, using the dual quaternion product.

The total transformation from R_1 to R_3 can be computed using a dual quaternion product as

$$\hat{\boldsymbol{q}}_3 = \hat{\boldsymbol{q}}_1 \otimes \hat{\boldsymbol{q}}_2. \tag{2.52}$$

If more transformations are required, this expression can be extended by adding as much dual quaternions as needed.

2.4.1.1 Dual Quaternion Inverse Transformations

From (2.52), inverse kinematics can be easily obtained just by multiplying by the conjugate of any required transformation. For example, if $\hat{\boldsymbol{q}}_2$ and $\hat{\boldsymbol{q}}_3$ are given, $\hat{\boldsymbol{q}}_1$ can be computed as

$$\hat{\boldsymbol{q}}_3 \otimes \hat{\boldsymbol{q}}_2^* = \hat{\boldsymbol{q}}_1 \otimes \hat{\boldsymbol{q}}_2 \otimes \hat{\boldsymbol{q}}_2^* \Rightarrow \hat{\boldsymbol{q}}_1 = \hat{\boldsymbol{q}}_3 \otimes \hat{\boldsymbol{q}}_2^*. \tag{2.53}$$

Similarly, if $\hat{\boldsymbol{q}}_1$ and $\hat{\boldsymbol{q}}_3$ are known, $\hat{\boldsymbol{q}}_2$ can be computed as

$$\hat{\boldsymbol{q}}_1^* \otimes \hat{\boldsymbol{q}}_3 = \hat{\boldsymbol{q}}_1^* \otimes \hat{\boldsymbol{q}}_1 \otimes \hat{\boldsymbol{q}}_2 \Rightarrow \hat{\boldsymbol{q}}_2 = \hat{\boldsymbol{q}}_1^* \otimes \hat{\boldsymbol{q}}_3. \tag{2.54}$$

2.4.2 Dual Quaternion Kinematics

Supposing a rigid body is subject to a translation $\vec{p}$ expressed in the inertial reference frame (or $\vec{p}_{\mathcal{B}} \in \mathbb{R}^3$ if its represented with respect to the body's reference frame), followed by a rotation represented by a quaternion $\boldsymbol{q}$, then its dual quaternion transformation can be expressed as

$$\hat{\boldsymbol{q}} = \boldsymbol{q} + \frac{1}{2}\vec{p} \otimes \boldsymbol{q}\epsilon = \boldsymbol{q} + \frac{1}{2}\boldsymbol{q} \otimes \vec{p}_{\mathcal{B}}\epsilon, \tag{2.55}$$

where $\otimes$ is a quaternion product. Since $||\boldsymbol{q}|| = 1$, then $\hat{\boldsymbol{q}}$ is considered to be a unit dual quaternion.

The derivate of the previous equation is obtained by differentiating (2.55) as

$$\begin{aligned}
\dot{\hat{\boldsymbol{q}}} &= \dot{\boldsymbol{q}} + \frac{1}{2}\left[\dot{\boldsymbol{q}} \otimes \vec{p}_{\mathcal{B}} + \boldsymbol{q} \otimes \dot{\vec{p}}_{\mathcal{B}}\right]\epsilon \\
&= \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} + \left[\frac{1}{4}\boldsymbol{q} \otimes \vec{\Omega} \otimes \vec{p}_{\mathcal{B}} + \frac{1}{2}\boldsymbol{q} \otimes \dot{\vec{p}}_{\mathcal{B}}\right]\epsilon \\
&= \frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} + \left[\frac{1}{2}\boldsymbol{q} \otimes (\vec{\Omega} \times \vec{p}_{\mathcal{B}}) + \frac{1}{4}\boldsymbol{q} \otimes \vec{p}_{\mathcal{B}} \otimes \vec{\Omega} + \frac{1}{2}\boldsymbol{q} \otimes \dot{\vec{p}}_{\mathcal{B}}\right]\epsilon \\
&= \frac{1}{2}\left(\boldsymbol{q} + \frac{\boldsymbol{q} \otimes \vec{p}_{\mathcal{B}}}{2}\epsilon\right) \otimes \left(\vec{\Omega} + [\vec{\Omega} \times \vec{p}_{\mathcal{B}} + \dot{\vec{p}}_{\mathcal{B}}]\epsilon\right).
\end{aligned} \tag{2.56}$$

Define

$$\hat{\xi} \triangleq \vec{\Omega} + [\vec{\Omega} \times \vec{p}_{\mathcal{B}} + \dot{\vec{p}}_{\mathcal{B}}]\epsilon, \tag{2.57}$$

where $\hat{\xi}$ is the *twist* dual vector (combination of angular and translational velocities). Finally, the expression for the derivate of a dual quaternion is written as

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes \hat{\xi}. \tag{2.58}$$

This can be interpreted as a screw; containing a translation vector along its length, and a rotation that around itself, see Figure 2.6.

2.4.3 Quadrotor Dual Quaternion Model

Consider the quadrotor as a rigid body, and describing its rotation and translation with respect to the body's reference frame as a dual quaternion $\hat{\boldsymbol{q}}_v$. Its dynamic model is given by

$$\begin{bmatrix} \dot{\hat{\boldsymbol{q}}}_v \\ \dot{\hat{\xi}}_v \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\hat{\boldsymbol{q}}_v \otimes \hat{\xi}_v \\ \hat{F}_v + \hat{u}_v \end{bmatrix}, \tag{2.59}$$

with

$$\hat{\xi}_v = \vec{\Omega}_v + [\vec{\Omega}_v \times \vec{p}_v + \dot{\vec{p}}_v]\epsilon,$$

where $\vec{\Omega}_v$ expresses the vehicle angular velocity and $\vec{p}_v$ defines its position with respect to its local frame.

The control and external forces are included in the terms $\vec{F}_v$ and $\hat{u}_v$ as

$$\begin{aligned} \hat{F}_v &= \vec{a}_v + (\vec{a}_v \times \vec{p}_v + \vec{\Omega}_v \times \dot{\vec{p}}_v)\epsilon, \\ \hat{u}_v &= J_v^{-1}\vec{\tau}_v + [J_v^{-1}\vec{\tau}_v \times \vec{p}_v + m_v^{-1}\vec{F}_v]\epsilon, \\ \vec{a}_v &= -J_v^{-1}(\vec{\Omega}_v \times J_v\vec{\Omega}_v). \end{aligned} \tag{2.60}$$

where J_v denotes the quadrotor's inertia matrix, m_v represents its mass, $\vec{\tau}_v \in \mathbb{R}^3$ corresponds to the total torque, and $\vec{F}_v = \vec{F}_{th} + \boldsymbol{q}^* \otimes m\vec{g} \otimes \boldsymbol{q}$ is the total force containing the thrust and gravity vectors in the body reference frame.

In the case of a symmetric vehicle, it is considered that the center of mass is located in the structure's geometric center. The effects of the combination of $\vec{\Omega}_v$ and $\dot{\vec{p}}_v$ can be considered to be nonexistent, thus simplifying the quadrotor model, which can be rewritten as

$$\begin{bmatrix} \dot{\hat{\boldsymbol{q}}}_v \\ \dot{\hat{\xi}}_v \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\hat{\boldsymbol{q}}_v \otimes \hat{\xi}_v \\ \hat{u}_v \end{bmatrix}, \tag{2.61}$$

with $\hat{\xi}_v = \vec{\Omega}_v + \dot{\vec{p}}_v\epsilon$, and

$$\hat{u}_v = -J_v^{-1}(\vec{\Omega}_v \times J_v\vec{\Omega}_v - \vec{\tau}_v) + [m_v^{-1}\vec{F}_v]\epsilon. \tag{2.62}$$

Defining $\vec{\tau}_v$ as a function of a quaternion reference computed by (2.40) or (2.44), then the control action, given by $\hat{u}_v$ can be designed to stabilize system (2.61) using many methodologies.

2.5 Modeling approaches conclusions

The mathematical concepts and properties of quaternions have been known for many years. However, until very recently, most research works in robotics, more specifically in UAVs has been based on more conservative approaches such as Euler angles. Nevertheless, the presence of important non-linearities, undesired effects such as the Gimball-Lock, and an inherent complexity when multiple rotations and translations are present, hinder the development of bolder algorithms and applications.

The aim of this chapter is to give an introduction to dynamic modeling of quadrotors using unit and dual quaternions. Even if this kind of UAVs are inherently underactuated, an approach was introduced such that its dynamic equations can be analyzed and treated as a fully actuated system. Although his approach can appear to be less intuitive and difficult to conceptualize, the application of quaternions can really simplify dynamic models, and help in the design of better controllers.

Chapter 3

Control approaches for aerial vehicles

3.1 Euler angles based controllers

In this section, two of the classical control methodologies will be presented to exemplify the complexity and drawbacks of the Euler angles formulation of the quadrotor model, exposing phenomena such as singularities and simplifications commonly made when using this approach.

A common practice with these techniques is to consider separately on one hand the vehicle vertical dynamics and on the other hand the longitudinal and lateral equations, following this methodology, an altitude controller using a sliding mode approach will be introduced, then a backstepping algorithm will be developed for the x and y translational equations.

3.1.1 Sliding mode altitude control

Firstly, the quadrotor altitude subsystem is considered from (2.13) as

$$m\ddot{z} = \cos\theta\cos\phi F_{th} - mg, \tag{3.1}$$

then a sliding manifold is proposed as

$$s_z = \dot{z} + k_z z\,, \;\; \dot{s}_z = \ddot{z} + k_z \dot{z}\,, \;\; k_z > 0\,, \tag{3.2}$$

define a positive definite function

$$V_z = \frac{1}{2}s_z^2\,, \tag{3.3}$$

such that its derivative is given by

$$\dot{V}_z = s_z \dot{s}_z = s_z(\ddot{z} + k_z \dot{z}) = s_z \left(\frac{F_{th}}{m} \cos\theta \cos\phi - g + k_z \dot{z} \right). \tag{3.4}$$

Proposing a control thrust force as

$$F_{th} = \frac{m(-k_z \dot{z} + g - \text{sign}(s_z))}{\cos\theta \cos\phi}, \tag{3.5}$$

then (3.4) becomes

$$\dot{V}_z = s_z \left(-\text{sign}(s_z)\right). \tag{3.6}$$

Since $V_z > 0$ and $\dot{V}_z < 0$ for all $\dot{z} + k_z z \neq 0$, the vehicle altitude is asymptotically stabilized by controller (3.5).

Note that this control approach has two drawbacks. Firstly, it is easy to remark that this controller is only valid if $\cos\theta$, $\cos\phi \neq 0$, meaning it reaches a singularity when the vehicle inclines at 90° in either pitch or roll angles. Secondly, since (3.5) does not take the lateral states into account, the magnitude of the force can not be regulated to take into account the vehicle x and y position.

3.1.2 Backstepping control

Since the quadrotor model from (2.13) and (2.14) includes highly nonlinear components, it is a common practice to simplify the equations in order to design controllers.

Considering the altitude is regulated by (3.5), and only small angle movements are performed by the vehicle ($\cos\theta, \cos\phi \approx 1$, $\sin\theta \approx \theta$, $\sin\phi \approx \phi$), and neglecting the Coriolis gyroscopic effects, then the quadrotor longitudinal and lateral dynamics can be written as

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -\frac{F_{th}}{m}\theta \\ \dot{\theta} \\ \tau_\theta \end{bmatrix}, \qquad \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \phi \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ \frac{F_{th}}{m}\phi \\ \dot{\phi} \\ \tau_\phi \end{bmatrix}. \tag{3.7}$$

For design purposes, consider the following dynamic system, which has a similar structure as (3.7):

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ bx_3 \\ x_4 \\ u \end{bmatrix}. \tag{3.8}$$

Define a positive definite function as

$$V_1 = \frac{1}{2} e_1^2, \tag{3.9}$$

where $e_1 = x_1 - x_1^d$ represents the error between x_1 and its desired value x_1^d, $\dot{x}_1^d = x_2^d$ such that differentiating (3.9) yields

$$\dot{V}_1 = e_1\dot{e}_1 = e_1(\dot{x}_1 - \dot{x}_1^d) = e_1(x_2 - x_2^d). \tag{3.10}$$

In order to achieve a decreasing V_1, a virtual variable is proposed as

$$x_2^v = x_2^d - e_1, \tag{3.11}$$

such that

$$\dot{V}_1 = e_1(x_2 - x_2^v - e_1) = -e_1^2 + e_1(x_2 - x_2^v) = -e_1^2 + e_1e_2, \tag{3.12}$$

where $e_2 = x_2 - x_2^v$ defines a speed error.

Introducing a second positive definite function as

$$V_2 = \frac{1}{2}e_2^2\ , \quad \dot{V}_2 = e_2\dot{e}_2 = e_2(\dot{x}_2 - \dot{x}_2^v) = e_2(bx_3 - \dot{x}_2^v), \tag{3.13}$$

defining a new virtual variable $\delta_1^v = -\dot{x}_2^v + e_2 + e_1$, and an error as $e_3 = \delta_1^v + bx_3$, such that (3.13) yields

$$\dot{V}_2 = -e_2^2 - e_1e_2 + e_2e_3, \tag{3.14}$$

Proposing a third positive definite function as

$$V_3 = \frac{1}{2}e_3^2\ , \quad \dot{V}_3 = e_3\dot{e}_3 = e_3(\dot{\delta}_1^v + bx_4). \tag{3.15}$$

Defining the last virtual variable and error as $\delta_2^v = \dot{\delta}_1^v + e_2 + e_3$ and $e_4 = \delta_2^v + x_4$, then (3.15) yields

$$\dot{V}_3 = -e_3^2 - e_2e_3 + e_3e_4. \tag{3.16}$$

Finally, the last positive definite function and its derivative are introduced as

$$V_4 = \frac{1}{2}e_4^2\ , \quad \dot{V}_4 = e_4\dot{e}_4 = e_4(\dot{\delta}_2^v + u). \tag{3.17}$$

In order to ensure asymptotic convergence, it is desired that $\dot{V}_4 = -e_4^2 - e_3e_4$, such that a Lyapunov function and its derivative can be defined as

$$V = V_1 + V_2 + V_3 + V_4 = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 + \frac{1}{2}e_3^2 + \frac{1}{2}e_4^2 \tag{3.18}$$

$$\dot{V} = -e_1^2 + e_1e_2 - e_2^2 - e_1e_2 + e_2e_3 - e_3^2 - e_2e_3 + e_3e_4 - e_4^2 - e_3e_4 \tag{3.19}$$

$$= -e_1^2 - e_2^2 - e_3^2 - e_4^2 \tag{3.20}$$

Therefore the controller that stabilizes system (3.8), according to the Lyapunov function (3.18) is given by

$$u = -e_4 - e_3 - \dot{\delta}_2^v, \tag{3.21}$$

such that $V > 0, \dot{V} < 0$ for all $e_i \neq 0, i = 1, \cdots, 4$. Therefore the errors converge asymptotically to zero.

Substituting errors and virtual variables as:

$$\delta_1^v = 2(x_1 - x_1^d) + 2(x_2 - x_2^d) - bx_3^d, \tag{3.22}$$
$$\dot{\delta}_1^v = 2(x_2 - x_2^d) + 2b(x_3 - x_3^d) - bx_4^d, \tag{3.23}$$
$$\delta_2^v = 3(x_1 - x_1^d) + 5(x_2 - x_2^d) + 3b(x_3 - x_3^d) - bx_4^d, \tag{3.24}$$
$$\dot{\delta}_2^v = 3(x_2 - x_2^d) + 5b(x_3 - x_3^d) + 3b(x_4 - x_4^d) - b\dot{x}_4^d, \tag{3.25}$$

such that the controller yields

$$u = -(\delta_2^v + x_4) - (\delta_1^v + bx_3) - \dot{\delta}_2^v, \tag{3.26}$$
$$= -5(x_1 - x_1^d) - 10(x_2 - x_2^d) - 9b(x_3 - x_3^d) - \left((3b+1)x_4 - 4bx_4^d\right) + b\dot{x}_4^d \tag{3.27}$$

Since the equilibrium point of the quadrotor model lies in hovering state, it is desired that $\phi_d, \theta_d, \dot{\phi}_d, \dot{\theta}_d = 0$, thus, the final position controllers for the vehicle are derived from (3.7) and (3.27)

$$\tau_\theta = -5(x - x^d) - 10(\dot{x} - \dot{x}_d) + 9\frac{F_{th}}{m}\theta - \left(3\frac{F_{th}}{m} + 1\right)\dot{\theta}, \tag{3.28}$$
$$\tau_\phi = -5(y - y^d) - 10(\dot{y} - \dot{y}_d) - 9\frac{F_{th}}{m}\phi - \left(3\frac{F_{th}}{m} + 1\right)\dot{\phi}, \tag{3.29}$$

where x_d, y_d represent the vehicle desired position.

It is easy to note the drawbacks of these controllers, first, in this example only small angle movements were considered, thus limiting the quadrotor real capabilities, then a consideration of the complete vehicle dynamics would be a difficult task, that could reveal nonlinearities and uncertainties that might be difficult to deal with. Another alternative for controlling the quadrotor will be discussed in the following.

3.2 Quaternion state feedback controller

In this section, a simple quaternion-based feedback controller is designed as an example using Lyapunov theory based on the proposed model from Section 2.3.2. This controller, consists on a force in the 3-dimensional space defined as

$$\vec{F}_u = -K_{pt}\left(\vec{p} - \vec{p}_d\right) - K_{dt}\left(\dot{\vec{p}} - \dot{\vec{p}}_d\right) - m\vec{g}, \tag{3.30}$$

where $K_{pt}, K_{dt} \in \mathbb{R}^{3\times 3}$ denote positive control gains, and $\vec{p}_d$ symbolizes the desired position for the quadrotor.

Then, the force computed by (3.30) is used to determine the attitude control action by introducing

$$\vec{\tau} = -2K_{pr}\ln\left(\boldsymbol{q}_z^* \otimes \left(\pm\left(\sqrt{\frac{1+\frac{\vec{F}_u}{||\vec{F}_u||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}} - \frac{\frac{\vec{F}_u}{||\vec{F}_u||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}}{\left\|\frac{\vec{F}_u}{||\vec{F}_u||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}\right\|}\sqrt{\frac{1-\frac{\vec{F}_u}{||\vec{F}_u||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}}\right)\right)\otimes\boldsymbol{q}\right)$$
$$-K_{dr}\vec{\Omega} + \vec{\Omega}\times J\vec{\Omega}, \tag{3.31}$$

where $\boldsymbol{q}_z = \cos(\psi_d/2) + [0\ 0\ 1]^T\sin(\psi_d/2)$ represents the quaternion transformation of the desired yaw angle, and control gains are denoted by positive matrices $K_{pr}, K_{dr} \in \mathbb{R}^{3\times3}$.

The development and stability proof of this algorithm will be explained in the following subsections.

3.2.1 Translational Controller

First, from (2.35), the linear translational subsystem can be written as

$$\frac{d}{dt}\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix} = \begin{bmatrix}0^{3\times3} & I^{3\times3}\\ 0^{3\times3} & 0^{3\times3}\end{bmatrix}\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix} + \begin{bmatrix}0^{3\times3}\\ m^{-1}I^{3\times3}\end{bmatrix}\left(\vec{F}_u + m\vec{g}\right). \tag{3.32}$$

Propose the following positive definite function and its derivative as

$$V_t = \frac{1}{2}\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix}\cdot\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix}, \tag{3.33}$$

$$\dot{V}_t = \begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix}\cdot\left[\begin{bmatrix}0^{3\times3} & I^{3\times3}\\ 0^{3\times3} & 0^{3\times3}\end{bmatrix}\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix} + \begin{bmatrix}0^{3\times3}\\ m^{-1}I^{3\times3}\end{bmatrix}\left(\vec{F}_u + m\vec{g}\right)\right]. \tag{3.34}$$

Proposing

$$\vec{F}_u = -\begin{bmatrix}K_{pt} & K_{pt}\end{bmatrix}\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix} - m\vec{g}, \tag{3.35}$$

where K_{pt}, $K_{dt} \in \mathbb{R}^{3\times3}$ contain control gains and are defined as

$$K_{pt} = \begin{bmatrix}k_{ptx} & 0 & 0\\ 0 & k_{pty} & 0\\ 0 & 0 & k_{ptz}\end{bmatrix}, \quad K_{dt} = \begin{bmatrix}k_{dtx} & 0 & 0\\ 0 & k_{dty} & 0\\ 0 & 0 & k_{dtz}\end{bmatrix}, \tag{3.36}$$

then (3.33) can be rewritten as

$$\dot{V}_t = \begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix}\cdot\left[\begin{bmatrix}0^{3\times3} & I^{3\times3}\\ 0^{3\times3} & 0^{3\times3}\end{bmatrix} - \begin{bmatrix}0^{3\times3}\\ m^{-1}I^{3\times3}\end{bmatrix}\begin{bmatrix}K_{pt} & K_{pt}\end{bmatrix}\right]\begin{bmatrix}\vec{p}_e\\ \dot{\vec{p}}_e\end{bmatrix}. \tag{3.37}$$

In order to asymptotically stabilize the sub-system, K_{pt} and K_{dt} must be chosen such that the real parts of

$$\operatorname{eig}\left(\begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ 0^{3\times3} & 0^{3\times3} \end{bmatrix} + \begin{bmatrix} 0^{3\times3} \\ m^{-1}I^{3\times3} \end{bmatrix} \begin{bmatrix} K_{pt} & K_{dt} \end{bmatrix}\right) \tag{3.38}$$

are negative definite.

If the condition satisfied, then asymptotic stability is ensured for system (3.32) since

$$V_t > 0, \quad \dot{V}_t < 0 \;\; \forall \;\; \begin{bmatrix} \vec{p}_e \\ \dot{\vec{p}}_e \end{bmatrix} \neq 0. \tag{3.39}$$

3.2.2 Rotational Controller

Considering $\boldsymbol{q}_e \triangleq \boldsymbol{q}_d^* \otimes \boldsymbol{q}$ and $\vec{\vartheta}_e = 2\ln(\boldsymbol{q}_e)$, $\dot{\vec{\vartheta}}_e = \vec{\Omega} - 2\frac{d}{dt}\ln(\boldsymbol{q}_d)$, the same methodology from Section 3.2.1 is now applied to the rotational error model in its axis-angle representation by introducing

$$\frac{d}{dt}\begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} = \begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ 0^{3\times3} & 0^{3\times3} \end{bmatrix} \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} + \begin{bmatrix} 0^{3\times3} \\ J^{-1} \end{bmatrix} \left(\vec{\tau} - \dot{\vec{\vartheta}}_e \times J\,\dot{\vec{\vartheta}}_e\right). \tag{3.40}$$

Proposing a positive-definite function with its derivative as

$$V_r = \frac{1}{2}\begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} \cdot \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix}, \tag{3.41}$$

$$\dot{V}_r = \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} \cdot \left[\begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ 0^{3\times3} & 0^{3\times3} \end{bmatrix} \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} + \begin{bmatrix} 0^{3\times3} \\ J^{-1} \end{bmatrix} \left(\vec{\tau} - \dot{\vec{\vartheta}}_e \times J\,\dot{\vec{\vartheta}}_e\right)\right]. \tag{3.42}$$

The controller can be defined as

$$\vec{\tau} = -\begin{bmatrix} K_{pr} & K_{dr} \end{bmatrix} \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} + \dot{\vec{\vartheta}}_e \times J\,\dot{\vec{\vartheta}}_e, \tag{3.43}$$

with control gains given by

$$K_{pr} = \begin{bmatrix} k_{prx} & 0 & 0 \\ 0 & k_{pry} & 0 \\ 0 & 0 & k_{prz} \end{bmatrix}, \quad K_{dr} = \begin{bmatrix} k_{drx} & 0 & 0 \\ 0 & k_{dry} & 0 \\ 0 & 0 & k_{drz} \end{bmatrix}. \tag{3.44}$$

Therefore, asymptotic stability for the rotational sub-system is ensured as long the real parts of

$$\operatorname{eig}\left(\begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ 0^{3\times3} & 0^{3\times3} \end{bmatrix} + \begin{bmatrix} 0^{3\times3} \\ J^{-1} \end{bmatrix} \begin{bmatrix} K_{pr} & K_{dr} \end{bmatrix}\right) \tag{3.45}$$

are negative such that

$$V_r > 0, \quad \dot{V}_r < 0 \;\; \forall \;\; \begin{bmatrix} \vec{\vartheta}_e \\ \dot{\vec{\vartheta}}_e \end{bmatrix} \neq 0. \tag{3.46}$$

Introducing (3.35) into (2.40), a desired attitude is defined to compute $\boldsymbol{q}_e$ and $\vec{\vartheta}_e$ such that the final controller expression yields (3.30) and (3.31).

3.2.3 Simulation example

In this subsection, numerical simulations of the model presented in (2.31) and the controllers from (3.35) and (3.43) are illustrated. The model parameters were taken from a custom-made quadrotor, and estimated using computer assisted design and finite-element techniques such that

$$m = 1.3\,\mathrm{kg} \quad , \quad J = \begin{bmatrix} 0.177 & 0 & 0 \\ 0 & 0.177 & 0 \\ 0 & 0 & 0.354 \end{bmatrix} \mathrm{kg\,m^2} \; . \tag{3.47}$$

For the controllers, their gains were empirically selected as

$$K_{pt} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 8 \end{bmatrix}, \; K_{dt} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \; K_r = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \; K_r = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \tag{3.48}$$

such that the eigenvalues of the closed-loop system ensure its stability.

A flight scenario is considered in which the quadrotor takes off from the origin to an altitude of 1m, then a $r = 2$m circular path is followed while a rotation around the vehicle's z axis is tracked such that the quadrotor's front is always pointed towards the center of the circle. This is achieved by designing a circular path p_d as

$$\vec{p}_d = \begin{cases} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T & \forall \quad t < 5\mathrm{s} \\ \begin{bmatrix} -r\cos(t-5) + r \\ -r\sin(t-5) \\ 0 \end{bmatrix} & \forall \quad t > 5\mathrm{s} \end{cases}, \tag{3.49}$$

and a complement to $\boldsymbol{q}_z$ from (2.41) by adapting (2.43) such that

$$\delta_{xy} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T - p \quad , \quad \boldsymbol{q}_z = e^{\dfrac{\ln\left(\delta_{xy} \otimes (-[1\;0\;0]^T)\right)}{2}} \; . \tag{3.50}$$

The controllers from (3.35) and (3.43) compute the forces and torques required to stabilize the vehicle in the desired references, shown in Figures 3.1 and 3.2 respectively.

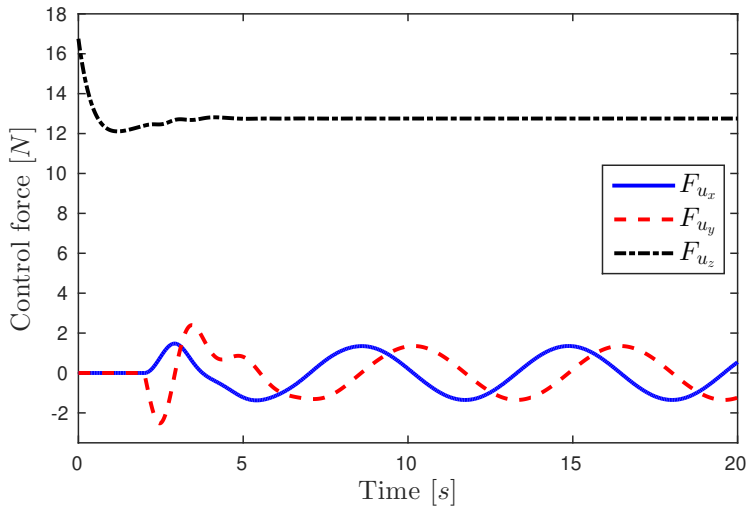


FIGURE 3.1: Translational controller

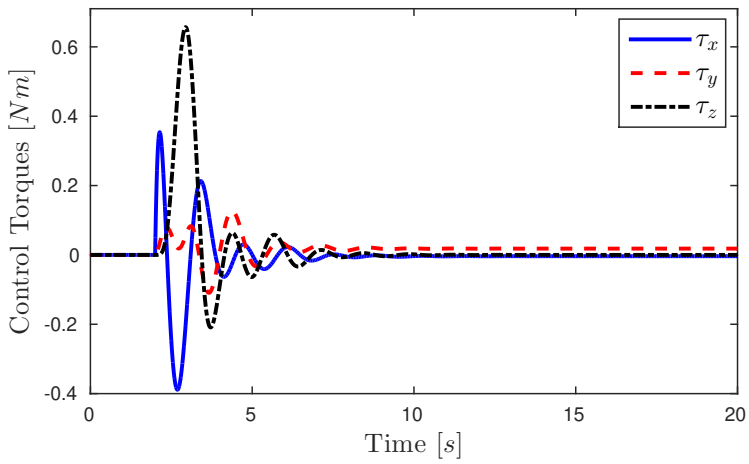


FIGURE 3.2: Rotational controller

The desired position is tracked by the effects of the control force as depicted on Figure 3.3. The quadrotor's front is symbolized by an arrow. Note its direction is pointed towards the center of the circle, following (3.50).

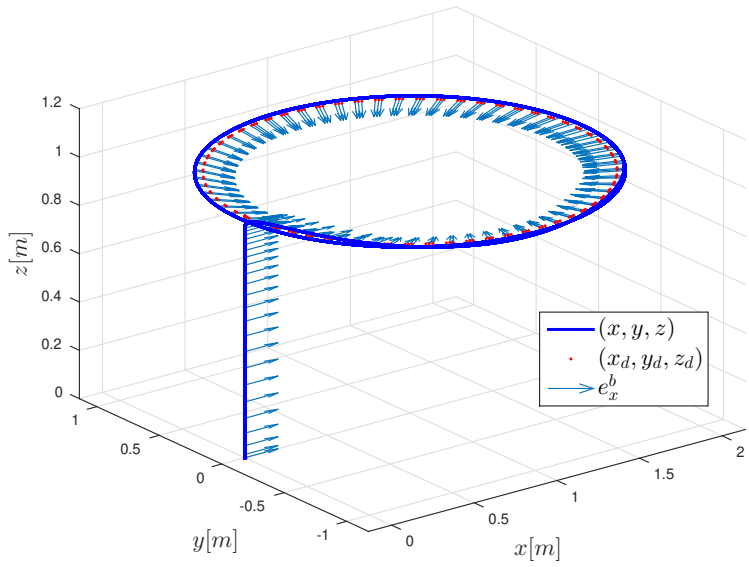
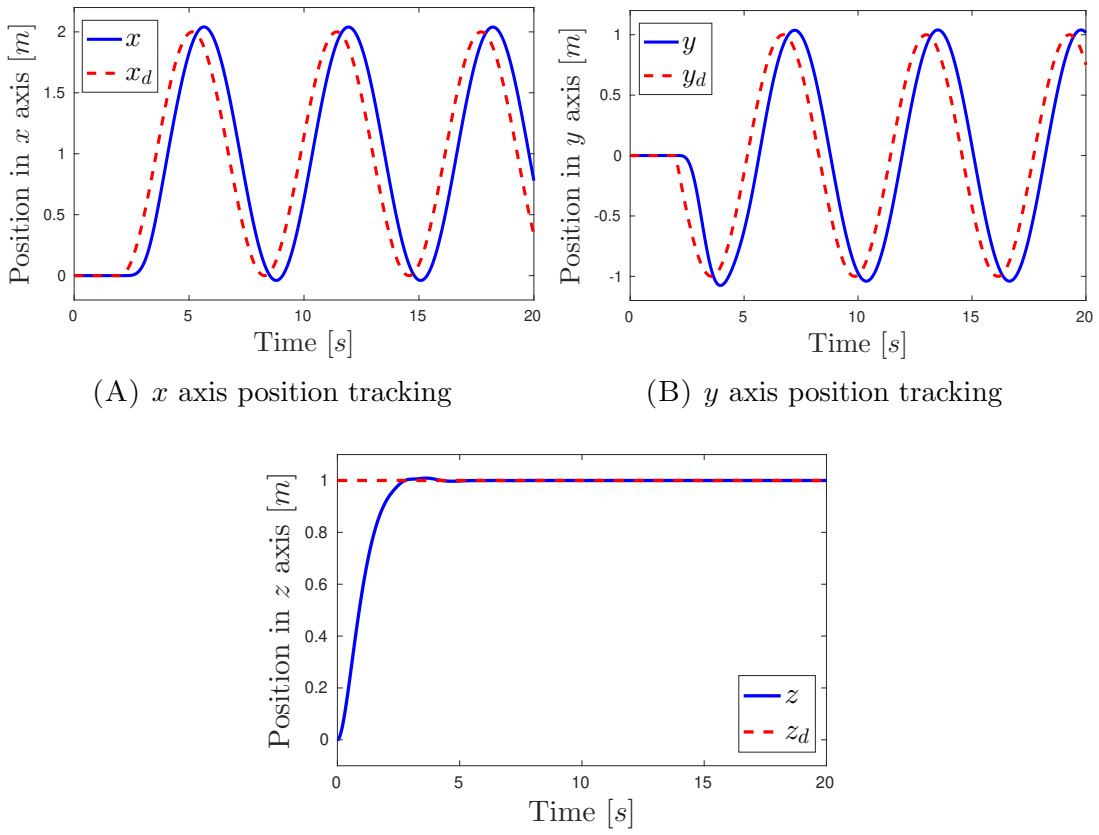


FIGURE 3.3: Quadrotor 3D flight

Figure 3.4 illustrates the translational response on each axis, the sinusoidal oscillations correspond to the vehicle circular movement.



(A) x axis position tracking

(B) y axis position tracking

(C) Quadrotor altitude regulation

FIGURE 3.4: Quadrotor position tracking

The attitude reference computed by (2.40) and (3.50) is represented in Figure 3.5. It is important to remark that, although the rotation of the vehicle completes more than one complete tour around the z axis, the attitude is continuous. This lack of discontinuity points is one of the main advantages of using quaternion approaches.

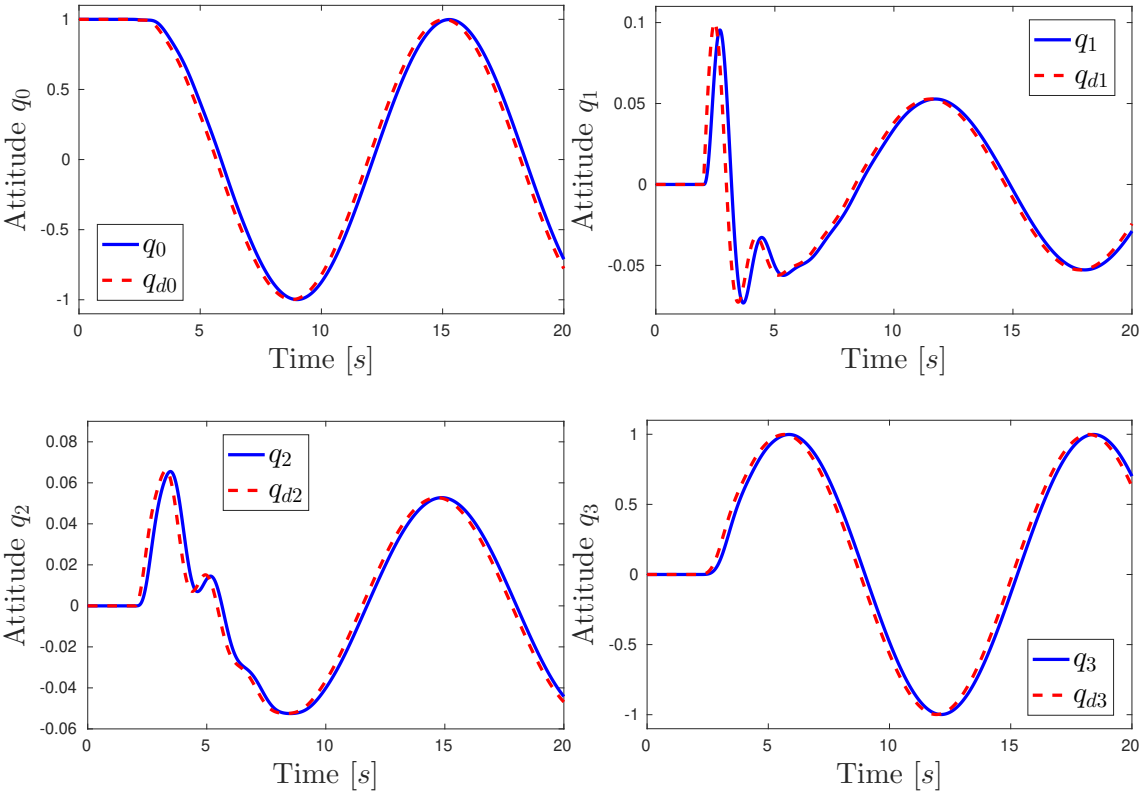
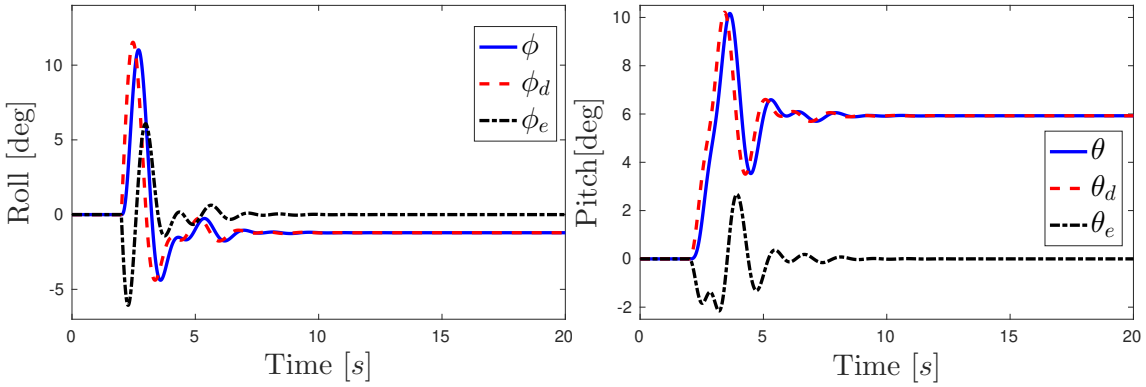


FIGURE 3.5: Quadrotor attitude quaternion trajectory tracking

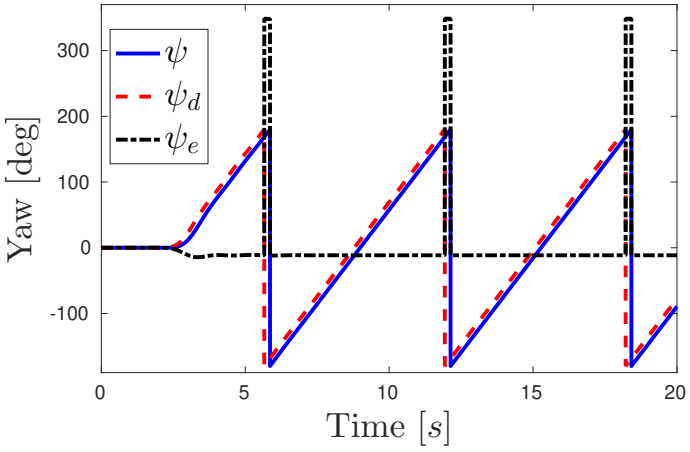
Finally, to better illustrate the vehicle's rotational behavior, the equivalent Euler angles are illustrated on Figure 3.6, which were computed by following

$$\begin{aligned}
\phi &= \tan^{-1}\left(\frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1q_1 + q_2q_2)}\right), \\
\theta &= \sin^{-1}\left(2(q_0q_2 - q_1q_3)\right), \\
\psi &= \tan^{-1}\left(\frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2q_2 + q_3q_3)}\right).
\end{aligned} \tag{3.51}$$

Note this conversion uncovers the discontinuities that are present when completing full rotations ($\pm 180°$), also note that the yaw angle error displays sudden jumps when such rotations are reached, this may cause undesired behaviors in experiments if left unfixed.



(A) Quadrotor angle and error in x axis (B) Quadrotor angle and error in y axis



(C) Quadrotor angle and error in z axis

FIGURE 3.6: Quadrotor attitude axis-angle representation

3.3 Passivity-based quaternion control

Passivity is a fundamental property of many physical systems which involves energy dissipation and transformation. Passivity-based control (PBC) methodology relies on synthesizing control laws which render the closed loop system passive [90].

In most of the works found in literature which use passivity to stabilize quadrotors, it is impossible to directly control the vehicle full dynamics because certain passivity properties are not satisfied, see [91], [92]. Nevertheless, controllers based on PBC can be used if the dynamics are modified, this requires the use of more complex algorithms and high computational cost implementations for just a partial stabilization.

By analyzing the quadrotor dynamic model from (2.32) and (2.34), the system can be analyzed as a fully actuated systems, linked by a quaternion rotation defined by (2.40), therefore, meeting the passivity conditions and enabling the use of the PBC methodology for all the vehicle states.

3.3.1 Classical PBC methodology

Typical passivity-based control methodology starts with a dynamic system defined as

$$\begin{cases} \dot{x}(t) = f(x) + g(x)u(t) \\ \tilde{y}(t) = h(x) \end{cases} \tag{3.52}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $\tilde{y} \in \mathbb{R}^m$, $f(x)$, $g(x)$ and $h(x)$ are smooth functions. There is a function $w\left[u, \tilde{y}\right]$ called supply rate, that is locally integrable, i.e.,

$$\int_{t_0}^{t_1} w\left[u\left(t\right), \tilde{y}\left(t\right)\right] dt < \infty, \forall t_0 \leq t_1$$

If there exists a function $H(x) \geq 0$, $H(0) = 0$, such that

$$H(x(t_1)) - H\left(x(t_0)\right) = \int_{t_0}^{t_1} w\left[u\left(t\right), \tilde{y}\left(t\right)\right] dt - d(t) \tag{3.53}$$

then, (3.52) is said to be a dissipative system. When $w\left[u\left(t\right), \tilde{y}\left(t\right)\right] = \tilde{y}^T\left(t\right) u\left(t\right)$, (3.52) represents a passive system. Here, $H(x)$ represents the storage function and $d(t)$ is the dissipated energy function. For the system (3.52), passivity is equivalent to the existence of a scalar $H(x)$ such that,

$$\left(\nabla_x H\right)^T f(x) \leq 0, \;\; h(x) = g^T(x) \nabla_x H,$$

where $\nabla_x H$ symbolizes the gradient of H with respect to x.

3.3.2 PBC Methodology for a Quad-rotor using Quaternions

The quadrotor dynamic model can be expressed in matrix form as

$$\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \begin{bmatrix} \ddot{\vec{p}} \\ \ddot{\vec{\vartheta}} \end{bmatrix} + \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix} = \mathcal{U}, \tag{3.54}$$

where $\vec{0}$ represents a zero vector, $\vec{F}_u$ and $\vec{\tau}$ denote the desired control force acting in the 3-dimensional space, and $\vec{\tau}_u = \vec{\tau} - \vec{\Omega} \times J\vec{\Omega}$ denotes the quadrotor control torque.

Considering $\frac{d}{dt} 2 \ln \boldsymbol{q} = \dot{\vec{\vartheta}} = \vec{\Omega}$, the total energy of system (3.54) is

$$H = \frac{1}{2} \begin{bmatrix} \dot{\vec{p}} \\ \vec{\Omega} \end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \begin{bmatrix} \dot{\vec{p}} \\ \vec{\Omega} \end{bmatrix} + \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix}^T \begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix}. \tag{3.55}$$

In PBC methodology, the control input is decomposed in two terms

$$\mathcal{U} = \mathcal{U}_{st}(\vec{p}, \vec{\vartheta}) + \mathcal{U}_{dy}(\dot{\vec{p}}, \vec{\Omega}), \tag{3.56}$$

where the first term is designed to achieve energy-shaping and the second one injects damping.

From (3.53), if there exists a $\mathcal{U}_{st}$ such that

$$-\int_0^t \mathcal{U}_{st}^T \begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} \tilde{y}(t)dt = H_a \begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} + k, \tag{3.57}$$

for some H_a and a constant k, then the energy-shaping term $\mathcal{U}_{st}$ will ensure that the map $\mathcal{U}_{st} \to \tilde{y}$ is passive with the following form for the desired energy function

$$H_d = H + H_a, \tag{3.58}$$

where H_a represents the supplied energy by the controller. We will require that the function H_d has an isolated minimum at ξ_*, that is

$$\xi_* = \arg\min H_d. \tag{3.59}$$

The passive outputs for this system are the generalized velocities, that is $\tilde{y} = [\dot{\vec{p}}, \vec{\Omega}]$. The easiest way to shape the energy is by following [93]

$$\mathcal{U}_{st} = \nabla H - K_p \left(\begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} - \xi_* \right), \tag{3.60}$$

where $\mathcal{U}_{st}$ stabilizes ξ_* with a Lyapunov function as the difference between the stored and supplied energies.

Then, introducing (3.60) and $\tilde{y} = \dot{\xi}$ into (3.57) we can obtain

$$H_a(\xi) = -\begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix}^T \begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} + \frac{1}{2} \left(\begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} - \xi_* \right) K_p \left(\begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} - \xi_* \right) + k, \tag{3.61}$$

where $K_p = K_p^T > 0$ contains design parameters. Substituting (3.55) and (3.61) into (3.58), it follows that

$$H_d = \frac{1}{2} \begin{bmatrix} \dot{\vec{p}} \\ \vec{\Omega} \end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \begin{bmatrix} \dot{\vec{p}} \\ \vec{\Omega} \end{bmatrix} + \frac{1}{2} \left(\begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} - \xi_* \right) K_p \left(\begin{bmatrix} \vec{p} \\ \vec{\vartheta} \end{bmatrix} - \xi_* \right). \tag{3.62}$$

Controller (3.60) can be rewritten as

$$\mathcal{U}_{st}(\vec{p}, \vec{\vartheta}) = \nabla H - \nabla H_d. \tag{3.63}$$

From (3.63), it is ensured that the map $\mathcal{U}_{st} \to \tilde{y}$ is passive with (3.62) as the desired energy function.

The damping injection term is then given by

$$\mathcal{U}_{dy}(\dot{\vec{p}}, \vec{\Omega}) = -K_d \begin{bmatrix} \dot{\vec{p}} \\ \vec{\Omega} \end{bmatrix}, \tag{3.64}$$

where $K_d = K_d^T > 0$ contains design parameters.

To compute the final controller, consider $K_p = \begin{bmatrix} K_{pt} & 0_{3\times3} \\ 0_{3\times3} & K_{pr} \end{bmatrix}$ and $K_d = \begin{bmatrix} K_{dt} & 0_{3\times3} \\ 0_{3\times3} & K_{dr} \end{bmatrix}$, the energy-shaping term $\mathcal{U}_{st}$ from (3.63) then takes the form

$$\mathcal{U}_{st}(\vec{p}, \vec{\vartheta}) = \begin{bmatrix} -m\vec{g} - K_{pt}\left(\vec{p} - \vec{p}_d\right) \\ -2K_{pr} \ln\left(\boldsymbol{q}_e\right) \end{bmatrix}.$$

Here $\vec{p}_d$ denotes the desired position, note that the term $\boldsymbol{q}_e = \boldsymbol{q}_d^* \otimes \boldsymbol{q}$ is the quaternion error between the actual orientation $\boldsymbol{q}$ and the desired reference $\boldsymbol{q}_d$, which is computed following (2.40) and (2.41). If the control law is such that $\ln\left(\boldsymbol{q}_e\right) \to [0 \quad 0 \quad 0]^T$, then $\boldsymbol{q}_e \to 1 + [0 \quad 0 \quad 0]^T$, which implies that the orientation of the quadrotor converges to the desired reference $\boldsymbol{q} \to \boldsymbol{q}_d^*$.

Hence, the damping injection term $\mathcal{U}_{dy}$ from (3.64) is determined as

$$\mathcal{U}_{dy}(\dot{\vec{p}}, \vec{\Omega}) = \begin{bmatrix} -K_{dt}\left(\dot{\vec{p}} - \dot{\vec{p}}_d\right) \\ -K_{dr}\left(\vec{\Omega} - \vec{\Omega}_d\right) \end{bmatrix},$$

where $\dot{\vec{\vartheta}}_d = 2\ln \boldsymbol{q}_d$, and matrices $K_i > 0$, $i : pt, pr, dt, dr$ are composed by tuning parameters. From (3.56) the control law is applied to the model (3.54) according to the following equivalence:

$$\begin{bmatrix} \vec{F}_u \\ \vec{\tau} \end{bmatrix} = \begin{bmatrix} -K_{pt}\left(\vec{p} - \vec{p}_d\right) - K_{dt}\left(\dot{\vec{p}} - \dot{\vec{p}}_d\right) - m\vec{g} \\ -2K_{pr} \ln\left(\boldsymbol{q}_z^* \otimes \boldsymbol{q}_t^* \otimes \boldsymbol{q}\right) - K_{dr}\left(\vec{\Omega} - \vec{\Omega}_d\right) + \vec{\Omega} \times J\vec{\Omega} \end{bmatrix}. \tag{3.65}$$

The obtained control law in quaternion space guarantees the stabilization of all the system states without any change in the model or in the PBC strategy.

3.4 Energy-based quaternion controllers

As a continuation of the passivity-based quaternion control explained in the previous section, other energy-based approaches were explored in a similar avenue, also taking the advantages of quaternion algorithms to develop equations, and to avoid undesired effects such as singularities and gumball locks.

From (3.54), the quadrotor total energy in terms of errors can be expressed as

$$\bar{H} = \frac{1}{2}\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & J\end{bmatrix}\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix} + \begin{bmatrix} m\vec{g}\\ \vec{0}\end{bmatrix}^T \begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}\otimes\boldsymbol{q}_d^*)\end{bmatrix}, \tag{3.66}$$

where $\vec{p}_d, \dot{\vec{p}}_d$ and $\vec{\Omega}_d$ denote the desired vehicle position, speed and angular velocity respectively. Differentiating the above along the trajectories of the system

$$\dot{\bar{H}} = \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & J\end{bmatrix}\begin{bmatrix}\ddot{\vec{p}}\\ \dot{\vec{\Omega}}\end{bmatrix} + \begin{bmatrix} m\vec{g}\\ \vec{0}\end{bmatrix}^T \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}. \tag{3.67}$$

Substituting model (3.54) into the above, it follows

$$\dot{\bar{H}} = \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \mathcal{U}. \tag{3.68}$$

Two energy-based control schemes are synthesized in the following subsections.

3.4.1 Energy feedback controller

Now, consider the following positive candidate Lyapunov function

$$V = \frac{1}{2}K_E\bar{H}^2 + \frac{1}{2}\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}\otimes\boldsymbol{q}_d^*)\end{bmatrix}^T K_p \begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}\otimes\boldsymbol{q}_d^*)\end{bmatrix}, \tag{3.69}$$

where $K_p, K_m \in \mathbb{R}^{6\times6}$ and $K_E \in \mathbb{R}$ denote positive control gains, and $\boldsymbol{q}_d$ symbolizes the desired quadrotor attitude quaternion. Differentiating (3.69) with respect to time

$$\dot{V} = K_E\bar{H}\dot{\bar{H}} + \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m \begin{bmatrix}\ddot{\vec{p}}\\ \dot{\vec{\Omega}}\end{bmatrix} + \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_p \begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix}.$$

Introducing (3.68), it yields

$$\dot{V} = K_E\bar{H}\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \mathcal{U} + \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m \begin{bmatrix}\ddot{\vec{p}}\\ \dot{\vec{\Omega}}\end{bmatrix} + \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_p \begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix}$$

Notice from (3.54) that $\begin{bmatrix}\ddot{\vec{p}}\\ \dot{\vec{\Omega}}\end{bmatrix} = \begin{bmatrix} mI_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & J\end{bmatrix}^{-1}\left(\mathcal{U} - \begin{bmatrix} m\vec{g}\\ \vec{0}\end{bmatrix}^T\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}\right)$, then

$$\dot{V} = \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m \left(\begin{bmatrix} mI_{3\times3} & 0_{3\times3}\\ 0_{3\times3} & J\end{bmatrix}^{-1}\left(\mathcal{U} - \begin{bmatrix} m\vec{g}\\ \vec{0}\end{bmatrix}^T\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}\right)\right) \tag{3.70}$$

$$+ \begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_p \begin{bmatrix}\vec{p}-\vec{p}_d\\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix} + K_E\bar{H}\begin{bmatrix}\dot{\vec{p}}-\dot{\vec{p}}_d\\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \mathcal{U}. \tag{3.71}$$

Factoring terms, it follows that

$$\dot{V} = \begin{bmatrix} \dot{\vec{p}} - \dot{\vec{p}}_d \\ \vec{\Omega} - \vec{\Omega}_d \end{bmatrix}^T \left(\left[K_E \bar{H} + K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \right] \mathcal{U} - K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} \right.$$
$$\left. + K_p \begin{bmatrix} \vec{p} - \vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix} \right). \tag{3.72}$$

Therefore, the first control law is defined such that:

$$-K_d \begin{bmatrix} \dot{\vec{p}} - \dot{\vec{p}}_d \\ \vec{\Omega} - \vec{\Omega}_d \end{bmatrix} = \left[K_E \bar{H} + K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \right] \mathcal{U} \tag{3.73}$$
$$- K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} + K_p \begin{bmatrix} \vec{p} - \vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix},$$

where $K_d = K_d^T > 0$. This leads to

$$\dot{V} = -\begin{bmatrix} \dot{\vec{p}} - \dot{\vec{p}}_d \\ \vec{\Omega} - \vec{\Omega}_d \end{bmatrix}^T K_d \begin{bmatrix} \dot{\vec{p}} - \dot{\vec{p}}_d \\ \vec{\Omega} - \vec{\Omega}_d \end{bmatrix}.$$

From (3.73) we can obtain

$$\mathcal{U} = [E]^{-1} \left[-K_p \begin{bmatrix} \vec{p} - \vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix} - K_d \begin{bmatrix} \dot{\vec{p}} - \dot{\vec{p}}_d \\ \vec{\Omega} - \vec{\Omega}_d \end{bmatrix} + K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} \right],$$

where $E = K_E \bar{H} + K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1}$, this ensures that E always has inverse and that U does not have singularities. The final control law can be rewritten, as follows

$$\mathcal{U} = \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix} = \left[K_E \bar{H} + K_m \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \right]^{-1} \begin{bmatrix} -K_{pt}(\vec{p} - \vec{p}_d) - K_{dt}(\dot{\vec{p}} - \dot{\vec{a}}_d) - K_{mt}\vec{g} \\ -2K_{pr}\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) - K_{dr}(\vec{\Omega} - \vec{\Omega}_d) \end{bmatrix}, \tag{3.74}$$

where $K_{pt} > 0$, $K_{pr} > 0$, $K_{dt} > 0$, $K_{dr} > 0$ and $K_{mt} > 0$ contain design parameters, p_d denotes the equilibrium configuration, and $\boldsymbol{q}_d$ is computed by (2.40).

The controller ensures $\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \to [0 \quad 0 \quad 0]^T$, then $\boldsymbol{q}_d^* \otimes \boldsymbol{q} \to 1 + [0 \quad 0 \quad 0]^T$, which implies that the orientation of the vehicle converges to the desired reference $\boldsymbol{q} \to \boldsymbol{q}_d$.

From (2.40) $\boldsymbol{q}_d$ is used to close the loop such that the quadrotor trust force is rotated to coincide with $\vec{F}_u$, thus the position is stabilized in the desired reference.

3.4.2 Energy-based optimal control

The second algorithm is derived from a performance cost function which is to be minimized, it is defined as follows:

$$C = \frac{1}{2}\int_0^\infty \left(\begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \\ \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T Q \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \\ \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix} + \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix}^T R \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix} \right) dt \quad (3.75)$$

where the state and input weighting matrices are assumed such that $Q = Q^T$, $Q > 0$ and $R = R^T$, $R \geqslant 0$.

System (3.54) can be optimally stabilized solving:

$$\frac{dV_o}{dt} + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \\ \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T Q \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \\ \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix} + \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix}^T R \begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u \end{bmatrix} = 0 \quad (3.76)$$

Then, consider the following Lyapunov candidate function based on the total energy

$$\begin{aligned} V_o =& \frac{1}{2}K_E\bar{H}^2 + \frac{1}{2}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T K_m \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix} \\ &+ \frac{1}{2}\begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix}^T K_p \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix} + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix}^T K_T \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix} \end{aligned} \quad (3.77)$$

where $K_T = {K_T}^T > 0$. Differentiating (3.77) along the trajectories of the system, and introducing (3.67) and (3.54) in the above, it yields

$$\begin{aligned} \dot{V}_o =& \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T K_T \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix} + K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} \\ &+ \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T K_p \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix} + \left(K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d \end{bmatrix}^T K_m + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q}) \end{bmatrix}^T K_T \right) \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix}^{-1} \left(\mathcal{U} - \begin{bmatrix} m\vec{g} \\ \vec{0} \end{bmatrix} \right). \end{aligned} \quad (3.78)$$

Finally, introducing (3.78) into (3.76) and applying dynamic programming, it follows that

$$
\begin{aligned}
0 =& \frac{\partial}{\partial\left(\mathcal{U}-\begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}\right)}\Bigg[\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_T \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix} + K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix} \\
&+ \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_p \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix} + \Bigg(K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J\end{bmatrix} \\
&+ \begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix}^T K_T\Bigg)\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J\end{bmatrix}^{-1}\left(\mathcal{U}-\begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}\right) \\
&+ \left(\mathcal{U}-\begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}\right)^T R\left(\mathcal{U}-\begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}\right)\Bigg].
\end{aligned}
\tag{3.79}
$$

Then,

$$
\begin{aligned}
0 =& K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T + \left(\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix}^T K_T\right)\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J\end{bmatrix}^{-1} \\
&+ R\left(\begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u\end{bmatrix} - \begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}\right).
\end{aligned}
\tag{3.80}
$$

Therefore, the control law can be represented as

$$
\begin{aligned}
\begin{bmatrix} \vec{F}_u \\ \vec{\tau}_u\end{bmatrix} =& -R^{-1}\Bigg[\left(\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}^T K_m + \begin{bmatrix} \vec{p}-\vec{p}_d \\ 2\ln(\boldsymbol{q}_d^*\otimes\boldsymbol{q})\end{bmatrix}^T K_T\right)\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & J\end{bmatrix}^{-1} \\
&+K_E\bar{H}\begin{bmatrix} \dot{\vec{p}}-\dot{\vec{p}}_d \\ \vec{\Omega}-\vec{\Omega}_d\end{bmatrix}\Bigg] + \begin{bmatrix} m\vec{g} \\ \vec{0}\end{bmatrix}.
\end{aligned}
\tag{3.81}
$$

Remembering that $\vec{\Omega}_d = 2\frac{d}{dt}\ln \boldsymbol{q}_d$ and $\boldsymbol{q}_d$ is designed to track the direction of $\vec{F}_u$ as defined by (2.40).

3.5 Geometrical bounding control

The previous controllers were capable of performing robust and precise flights in most cases, however, if sudden changes in the reference occur, or if extremely large disturbances interfere with the quadrotor navigation, some controllers might not be able to recover the vehicle and continue the mission.

A controller for robustly tracking trajectories was developed by introducing a function which bounds the magnitude of any given vector inside of a cylinder. The idea is that the final controllers limit the action of the control forces and torques.

Define a *cylindrical bounded function* $\sigma_b(\vec{a}) : \mathbb{R}^3 \to \mathbb{R}^3$, with arguments $\vec{a} = [a_x, a_y, a_z]^T$ and bounding limits $b = [b_{xy}, b_z]^T$, as

$$\begin{aligned}
\sigma_b(\vec{a}) &:= \begin{bmatrix} \sigma_{xy} \\ \sigma_z \end{bmatrix}, \ \sigma_{xy} \in \mathbb{R}^2, \sigma_z \in \mathbb{R} \ ; \\
\sigma_{xy} &:= \begin{cases} [a_x, a_y]^T & \text{for} \quad ||[a_x, a_y]|| < b_{xy} \, , \\ b_{xy} \frac{[a_x, a_y]^T}{||[a_x, a_y]^T||} & \text{for} \quad ||[a_x, a_y]|| \geq b_{xy} \, , \end{cases} \\
\sigma_z &:= \begin{cases} a_z & \text{for} \quad |a_z| < b_z \, , \\ b_z \, \text{sign}(a_z) & \text{for} \quad |a_z| \geq b_z \, . \end{cases}
\end{aligned} \tag{3.82}$$

Note that vector $\sigma_b(\vec{a})$ is contained inside a cylinder centered in the origin with radius b_{xy} and height $2b_z$, this ensures a symmetrical behavior in the xy plane, and independent bounds in the z axis, as Figure 3.7 illustrates.

The control strategy will be based on a function which bounds a vector inside of a cylinder such that a symmetrical behavior is ensured in any direction of the $x - y$ plane while setting a different bound for the vertical axis.

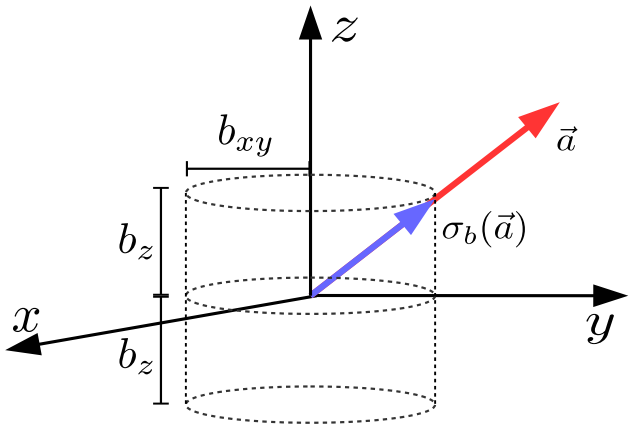


FIGURE 3.7: Bounding a vector inside a cylinder.

The following controllers asymptotically stabilize a quadrotor by bounding the control action inside a cylinder:

$$\vec{F}_u = -mK_t\sigma_c\left(K_c\left(\dot{\vec{p}} - \dot{\vec{p}}_d\right) + \sigma_d\left(K_dK_c\left(\vec{p} - \vec{p}_d\right) + K_d\left(\dot{\vec{p}} - \dot{\vec{p}}_d\right)\right)\right) - m\vec{g}, \tag{3.83}$$

$$\vec{\tau} = -JK_r\sigma_a\left(K_a\vec{\Omega} + \sigma_b\left(2K_bK_a\ln\left(\boldsymbol{q}_z^* \otimes \boldsymbol{q}_t^* \otimes \boldsymbol{q}\right) + K_b\vec{\Omega}\right)\right) + \vec{\Omega} \times J\,\vec{\Omega}, \tag{3.84}$$

where $\boldsymbol{q}_t$ is computed by introducing (3.83) into (2.40), $\boldsymbol{q}_z = \cos(\psi_d/2) + [0\ 0\ 1]^T \sin(\psi_d/2)$ represents the desired yaw rotation, $K_t, K_r, K_a, K_b, K_c, K_d$ represent constant diagonal matrices, and $\sigma_a, \sigma_b, \sigma_c, \sigma_d$ are bounding functions with bounds a, b, c, and d. The detailed development of this control approach will be stated in the rest of this section.

3.5.1 Rotational Bounded Algorithm

The goal is to propose a rotational controller to stabilize the drone at a desired attitude that will be computed from the direction of a control force. This controller will be developed in the following.

Lemma 3.1. *Attitude subsystem (2.33) converges asymptotically to the origin by the effects of a bounded controller defined as*

$$\vec{\tau} = -JK_r\sigma_a\left(K_a\vec{\Omega} + \sigma_b\left(2K_bK_a\ln\left(\boldsymbol{q}_z^* \otimes \boldsymbol{q}_t^* \otimes \boldsymbol{q}\right) + K_b\vec{\Omega}\right)\right) + \vec{\Omega} \times J\,\vec{\Omega}\ . \tag{3.85}$$

where $K_r, K_a, K_b \in \mathbb{R}^{3\times3}$ denote positive diagonal constant matrices.

Proof. A positive-definite function and its derivative are defined for the attitude system as $V_1 : \mathbb{R}^3 \to \mathbb{R}^+$

$$V_1 := \frac{1}{2}\vec{\Omega}\cdot\vec{\Omega} \quad ; \quad \dot{V}_1 = \vec{\Omega}\cdot J^{-1}(\vec{\tau} - \vec{\Omega}\times J\,\vec{\Omega})\ . \tag{3.86}$$

Define

$$\vec{\tau} = -J^{-1}K_r\sigma_a(K_a\vec{\Omega} + \vec{\varphi}_b) + \vec{\Omega}\times J\,\vec{\Omega}\ , \tag{3.87}$$

where $\vec{\varphi}_b = [\varphi_{bx}, \varphi_{by}, \varphi_{bz}]^T$ denotes a vectorial bounded function with limits b_{xy} and b_z such that $||[\varphi_{bx}, \varphi_{by}]|| < b_{xy}$, $|\varphi_{bz}| < b_z$. $K_r, K_a \in \mathbb{R}^{3\times3}$ are constant positive-definite diagonal matrices, therefore (3.86) becomes

$$\dot{V}_1 = -\vec{\Omega}\cdot J^{-1}K_r\sigma_a(K_a\vec{\Omega} + \vec{\varphi}_b)\ . \tag{3.88}$$

(3.88) can be analyzed in two cases as follows. Since $\sigma_a(K_a\vec{\Omega} + \vec{\varphi}_b)$ is bounded, then $\dot{V}_1 < 0$ if

$$\text{sign}(k_{ai}\Omega_i + \varphi_{bi}) = \text{sign}(\Omega_i) \Rightarrow |k_{ai}\Omega_i| > b_i\ , \tag{3.89}$$

where k_{ai} , $i : x, y, z$ represent the diagonal entries of K_a, implying that there exists T_0, such that for any time $t > T_0$, $|k_{ai}\Omega_i + \varphi_{bi}| \leq 2b_i$. Therefore, bounds have to be chosen as $a_{xy} > 2b_{xy}$, and $a_z > 2b_z$ to ensure that $V_1 < 0$.

Notice that for all $t > T_0$, a second case arises where $\sigma_a(K_a\vec{\Omega} + \vec{\varphi}_b) = K_a\vec{\Omega} + \vec{\varphi}_b$, and (3.87) can be considered as

$$\vec{\tau} = -J^{-1}K_r(K_a\vec{\Omega} + \vec{\varphi}_b) + \vec{\Omega}\times J\,\vec{\Omega}\ . \tag{3.90}$$

Considering $\frac{d}{dt}2\ln\boldsymbol{q}_e = \dot{\vec{\vartheta}} = \vec{\Omega}$ to be the angular speed, $\vec{\nu}_1 \in \mathbb{R}^3$ is defined as

$$\vec{\nu}_1 := 2J^{-1}K_rK_a\ln\boldsymbol{q}_e + \vec{\Omega}\ \ ;\ \ \dot{\vec{\nu}}_1 = J^{-1}K_rK_a\vec{\Omega} + \dot{\vec{\Omega}}\ , \tag{3.91}$$

introducing (3.90) into (3.91) yields,

$$\dot{\vec{\nu}}_1 = J^{-1}K_rK_a\vec{\Omega} - J^{-1}K_r(K_a\vec{\Omega} + \vec{\varphi}_b) = -J^{-1}K_r\vec{\varphi}_b\ . \tag{3.92}$$

A second positive definite function $V_2 : \mathbb{R}^3 \to \mathbb{R}^+$ is then introduced as

$$V_2 = \frac{1}{2}\vec{\nu}_1 \cdot \vec{\nu}_1 \quad ; \quad \dot{V}_2 = \vec{\nu}_1 \cdot \dot{\vec{\nu}}_1 = -\vec{\nu}_1 \cdot J^{-1} K_r \vec{\varphi}_b \,. \tag{3.93}$$

Propose $\vec{\varphi}_b$ as

$$\vec{\varphi}_b = \sigma_b(K_b \vec{\nu}_1) \ , \tag{3.94}$$

where K_b is a diagonal matrix containing positive gains, therefore $\dot{V}_2 = -\vec{\nu}_1 \cdot J^{-1} K_r \sigma_b(K_b \vec{\nu}_1) < 0$.

From (3.93) and (3.94) it is implied that $\vec{\nu}_1 \to [0,0,0]^T$ and $\dot{\vec{\nu}}_1 \to [0,0,0]^T$, then, from (3.92), it follows that $\vec{\varphi}_b \to [0,0,0]^T$, from (3.88), it yields $\vec{\Omega} \to [0,0,0]^T$ and $\ln \boldsymbol{q}_e \to [0,0,0]^T$, hence ensuring stability for the attitude subsystem. □

Finally, $\vec{\tau}$ is rewritten as

$$\vec{\tau} = -J^{-1} K_r \sigma_a (K_a \vec{\Omega} + \sigma_b (2 K_b K_a \ln (\boldsymbol{q}_z^* \otimes \boldsymbol{q}_t^* \otimes \boldsymbol{q}) + K_b \vec{\Omega})) + \vec{\Omega} \times J \vec{\Omega} \ . \tag{3.95}$$

Following (2.40), the attitude reference for controller (3.95) is computed as

$$\boldsymbol{q}_t = \pm \left(\sqrt{\frac{1 + \frac{\vec{F}_u}{||\vec{F}_u||} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{2}} + \frac{\frac{\vec{F}_u}{||\vec{F}_u||} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{\left|\left| \frac{\vec{F}_u}{||\vec{F}_u||} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right|\right|} \sqrt{\frac{1 - \frac{\vec{F}_u}{||\vec{F}_u||} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}{2}} \right) , \tag{3.96}$$

where $\vec{F}_u$ will be defined in the next section by a cylindrical bounded control force which will stabilize the vehicle translational dynamics.

3.5.2 Translational Bounded Control Law

The next step is to define a bounded control force which stabilizes the quadrotor translational dynamics. This controller is developed by following a methodology, similar to the rotational case.

Lemma 3.2. *The position subsystem (2.35) converges asymptotically to zero by the effects of a desired control force defined as*

$$\vec{F}_u = -m K_t \sigma_c \left(K_c \left(\dot{\vec{p}} - \dot{\vec{p}}_d \right) + \sigma_d \left(K_d K_c \left(\vec{p} - \vec{p}_d \right) + K_d \left(\dot{\vec{p}} - \dot{\vec{p}}_d \right) \right) \right) - m\vec{g} \,, \tag{3.97}$$

where $K_t, K_c, K_d \in \mathbb{R}^{3\times 3}$ are positive diagonal matrices.

Proof. Proposing two positive-definite functions $V_3, V_4 : \mathbb{R}^3 \to \mathbb{R}^+$ and their derivatives as

$$\begin{aligned} V_3 &:= \tfrac{1}{2}\dot{\vec{p}} \cdot \dot{\vec{p}} \quad ; \quad \dot{V}_3 = \dot{\vec{p}} \cdot \left(\tfrac{1}{m}\vec{F}_u + \vec{g}\right) \,, \\ V_4 &:= \tfrac{1}{2}\vec{\nu}_2 \cdot \vec{\nu}_2 \quad ; \quad \dot{V}_4 = \vec{\nu}_2 \cdot \dot{\vec{\nu}}_2 \,, \end{aligned} \tag{3.98}$$

with $\vec{\nu}_2 := mK_tK_c\vec{p} + \dot{\vec{p}}$. Following the same procedure as in the attitude case, it straightforwardly yields the controller given by (3.97). □

The proposed cylindrical bounded controller is capable of robustly tracking trajectories which can be introduced into $\vec{p}_d$ and $\dot{\vec{p}}_d$. One example is an autonomous coordinated circular target tracking algorithm, which will be further explained in the next chapter.

3.6 Spherical chattering-free sliding mode control

With the motivation of achieving faster convergence of the quadrotor states, with the aim of performing aggressive navigation scenarios which requiring fast movements and robustness on the system, a controller was designed using a modified sliding-mode technique, enhanced for 3-dimensional functions.

Define a function $\Gamma : \mathbb{R}^3 \to \mathbb{R}^3$, which bounds any vector into a sphere (see Figure 3.8), respecting its original direction and modifying its magnitude in terms of a continuous arctangent function as

$$\Gamma(\vec{\boldsymbol{k}}) := \hat{\boldsymbol{k}}\ \frac{2}{\pi}\tan^{-1}\left(||\vec{\boldsymbol{k}}||\right) , \tag{3.99}$$

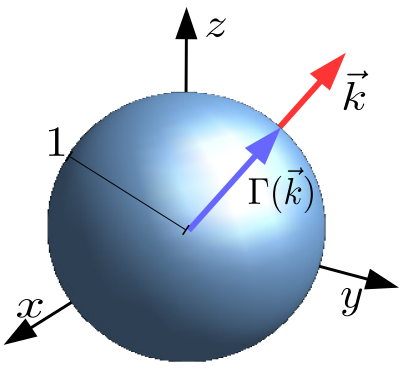


FIGURE 3.8: Spherical bounding of a 3-D vector, its bounded magnitude is defined by an arctangent function.

where $\vec{\boldsymbol{k}} \in \mathbb{R}^3$ represents any vector, and $\hat{\boldsymbol{k}}$ symbolizes its direction, computed as

$$\hat{\boldsymbol{k}} := \begin{cases} \dfrac{\vec{\boldsymbol{k}}}{||\vec{\boldsymbol{k}}||} & \text{for} \quad ||\vec{\boldsymbol{k}}|| \neq 0 \,, \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T & \text{for} \quad ||\vec{\boldsymbol{k}}|| = 0 \,. \end{cases} \tag{3.100}$$

Lemma 3.3. *Given any vector $\vec{\boldsymbol{k}}$, $\Gamma(\vec{\boldsymbol{k}})$ is bounded by a unitary sphere and it holds*

$$\text{sign}(\Gamma(\vec{\boldsymbol{k}})_i) = \text{sign}(\vec{\boldsymbol{k}}_i), \tag{3.101}$$

for every component $i = x, y, z$ of both vectors.

Proof. Computing the normalized vector of $\Gamma(\vec{\boldsymbol{k}})$, it yields

$$\frac{\Gamma(\vec{\boldsymbol{k}})}{||\Gamma(\vec{\boldsymbol{k}})||} = \frac{\hat{\boldsymbol{k}}\ \frac{2}{\pi}\tan^{-1}\left(||\vec{\boldsymbol{k}}||\right)}{\frac{2}{\pi}\tan^{-1}\left(||\vec{\boldsymbol{k}}||\right)} = \hat{\boldsymbol{k}}, \tag{3.102}$$

therefore, $\vec{\boldsymbol{k}}$, and $\Gamma(\vec{\boldsymbol{k}})$ share the same direction, and since the magnitude of the former is defined by an arctangent function with a positive definite argument, it holds

$$0 < \frac{2}{\pi}\tan^{-1}\left(||\vec{\boldsymbol{k}}||\right) < 1. \tag{3.103}$$

□

Using the previous formulation, a sliding mode translational controller is proposed such that the control action is defined by a spherical arctangent function as

$$\vec{F}_u = -\frac{K_{tp}\vec{p}_e + K_{td2}\dot{\vec{p}}_e}{\left|\left|K_{tp}\vec{p}_e + K_{td2}\dot{\vec{p}}_e\right|\right|}\ \frac{2}{\pi}\tan^{-1}\left|\left|K_{tp}\vec{p}_e + K_{td2}\dot{\vec{p}}_e\right|\right| - K_{td1}\dot{\vec{p}}_e - m\vec{g}, \tag{3.104}$$

then, a similar approach yields a rotational algorithm based on the same principle, which yields a torque defined as

$$\tau_u = -\frac{K_{rp}\ln\left(\boldsymbol{q}_e\right) + K_{rd2}\vec{\Omega}}{\left|\left|K_{rp}\ln\left(\boldsymbol{q}_e\right) + K_{rd2}\vec{\Omega}\right|\right|}\ \frac{2}{\pi}\tan^{-1}\left|\left|K_{rp}\ln\left(\boldsymbol{q}_e\right) + K_{rd2}\vec{\Omega}\right|\right| - K_{rd1}\vec{\Omega} + \vec{\Omega}\times J\,\vec{\Omega}. \tag{3.105}$$

The development of both controllers will be detailed in the following subsections.

3.6.1 Attitude Control Formulation

Theorem 3.4. *The attitude subsystem (2.32) of a quadrotor converges asymptotically to the quaternion origin $\boldsymbol{q}_{\mathcal{O}} = 1 + [0\ 0\ 0]^T$ by means of*

$$\vec{\tau} = -K_{rd1}\vec{\Omega} - \Gamma\left(2K_{rp}\ln\boldsymbol{q}_e + K_{rd2}\vec{\Omega}\right) + \vec{\Omega}\times J\vec{\Omega}. \tag{3.106}$$

Proof. Propose a positive-definite function and its derivative as

$$V_1 = \frac{1}{2}\left(2\ln\boldsymbol{q}_e\right)\cdot\left(2\ln\boldsymbol{q}_e\right)\ \rightarrow\ \dot{V}_1 = 2\ln\boldsymbol{q}_e\cdot\vec{\Omega}. \tag{3.107}$$

Asymptotic convergence for $\vec{\vartheta}_e = 2\ln \boldsymbol{q}_e$ is achieved if $\dot{V}_1 < 0$ for all $\vec{\vartheta}_e \neq \vec{0}$, which is ensured if $\text{sign}(\Omega_i) = \text{sign}(\vartheta_i)$, for $i = x, y, z$, this can be achieved by proposing $K_1\vec{\Omega} = -2K_2 \ln \boldsymbol{q}_e$, where $K_j \in \mathbb{R}^{3\times 3}, j = 1, 2, ...,$ are positive diagonal matrices.

Define a second positive-definite function as

$$V_2 = \frac{1}{2}\varsigma_r \cdot \varsigma_r \;\rightarrow\; \dot{V}_2 = \varsigma_r \cdot \dot{\varsigma}_r, \tag{3.108}$$

where ς_r is a sliding manifold such that

$$\begin{aligned} \varsigma_r &= K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e\,, \\ \dot{\varsigma}_r &= K_1(J^{-1}\vec{\tau} - J^{-1}\vec{\Omega} \times J\vec{\Omega}) + K_2\vec{\Omega}\,, \end{aligned} \tag{3.109}$$

note that asymptotic stability for (3.108) is reached if

$$\dot{\varsigma}_r = -K_3\Gamma\left(K_4\varsigma_r\right). \tag{3.110}$$

Introducing (3.110) into (3.109), yields

$$\begin{aligned} \vec{\tau} \;=\; & -JK_1^{-1}K_3\Gamma\left(K_4K_1\vec{\Omega} + 2K_4K_2\ln \boldsymbol{q}_e\right) \\ & -JK_1^{-1}K_2\vec{\Omega} + \vec{\Omega} \times J\vec{\Omega}\,, \end{aligned} \tag{3.111}$$

propose $K_3 = K_1J^{-1}$, $K_p = K_4K_2$, $K_{d1} = K_3^{-1}K_2$, and $K_{d2} = K_4K_1$, (3.111) is finally expressed as

$$\vec{\tau} = -K_{d1}\vec{\Omega} - \Gamma\left(2K_p \ln \boldsymbol{q}_e + K_{d2}\vec{\Omega}\right) + \vec{\Omega} \times J\vec{\Omega}. \tag{3.112}$$

Finally, a Lyapunov candidate function is proposed as

$$V_r := \frac{1}{2}(K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e) \cdot (K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e), \tag{3.113}$$

being

$$\dot{V}_r = (K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e) \cdot (K_1(J^{-1}\vec{\tau} - J^{-1}\vec{\Omega} \times J\vec{\Omega}) + K_2\vec{\Omega}). \tag{3.114}$$

Introducing (3.112), it yields

$$\dot{V}_r = -\big(K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e\big) \cdot \Gamma\left(K_4\big(K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e\big)\right). \tag{3.115}$$

From Lemma 3.3, it is clear that $\dot{V} < 0$ for all $\varsigma_r \neq 0$ therefore forcing (3.113) to converge to the manifold $\varsigma_r = K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e$, if matrices K_1 and K_2 are chosen such that the sliding manifold is asymptotically stable, then $K_1\vec{\Omega} + 2K_2 \ln \boldsymbol{q}_e \rightarrow 0$ and $K_1\vec{\Omega} \rightarrow -2K_2 \ln \boldsymbol{q}_e$.

Since $\dot{\vec{\vartheta}} = \vec{\Omega}$, $2\ln \boldsymbol{q}_e$ asymptotically converges to zero and following (2.15), then $\boldsymbol{q} \rightarrow \boldsymbol{q}_{\mathcal{O}}$. □

3.6.2 Position Controller

Following the same procedure as in the attitude subsystem, the quadrotor translational dynamics can be controlled as follows:

Theorem 3.5. *The position subsystem (2.34) of a quadrotor converges asymptotically to a zero vector by means of*

$$\vec{F}_u = -K_{td1}\dot{\vec{p}}_e - \Gamma\left(K_{tp}\vec{p}_e + K_{td2}\dot{\vec{p}}_e\right) - m\vec{g}. \tag{3.116}$$

Proof. Proposing a Lyapunov function as

$$V_t := \frac{1}{2}(K_{t1}\dot{\vec{p}}_e + K_{t2}\vec{p}_e)\cdot(K_{p1}\dot{\vec{p}}_e + K_{p2}p), \tag{3.117}$$

and applying (3.116) its derivative yields

$$\dot{V}_t = -\big(K_{t1}\dot{\vec{p}}_e + K_{t2}\vec{p}_e\ \big)\cdot\Gamma\left(K_{t4}\big(K_{t1}\dot{\vec{p}}_e + K_{t2}\vec{p}_e\ \big)\right), \tag{3.118}$$

where $K_{tj} \in \mathbb{R}^{3\times 3}, j = 1, 2, ...,$ represent positive diagonal matrices with $K_{t3} = K_{t1}m^{-1}$, $K_{tp} = K_{t4}K_{t2}$, $K_{td1} = K_{t3}^{-1}K_{t2}$, and $K_{td2} = K_{t4}K_{t1}$.

As in the attitude case, Lemma 3.3 implies that (3.118) is negative definite for all $K_{t1}\dot{\vec{p}}_e \neq -K_{t2}\vec{p}_e$, therefore forcing the convergence of (3.117) to the manifold $\sigma_t = K_{t1}\dot{\vec{p}}_e + K_{t2}\vec{p}_e$, implying $K_{t1}\dot{\vec{p}}_e + K_{t2}\vec{p}_e \to 0$ and $K_{t1}\dot{\vec{p}}_e \to -K_{t2}\vec{p}_e$. □

Defining $\vec{p}_e = \vec{p} - \vec{p}_d$, and applying (2.40) then the quadrotor can be stabilized to follow any position reference and any desired angle in the z axis.

3.7 Control approaches conclusions

Quadrotors are known to be inherently unstable, nonlinear, and underactuated systems. In recent years, researchers have been developing algorithms and techniques to control these vehicles in order to accomplish different kinds of tasks. Nevertheless, as aerial vehicles became more advanced and popular, the interest on them shifted from simple stabilization to bolder and more complicated applications, which would require very complex approaches, that could even be impossible to accomplish using the classical modeling techniques.

In this chapter, some control techniques based on quaternion formulations were presented, starting from a simple state-feedback algorithm, followed by passivity, energy, and 3-dimensional saturation approaches, up to a nonlinear sliding-mode controller.

In the following section, some applications of the previous controllers will be presented, simulations and experiments will be detailed to validate each technique.

Chapter 4

Navigation techniques for quadrotors

In most works currently found in literature, quadrotor navigation relies on controllers based on Euler-angles models, however such approaches commonly ensure system stability only in a linear region defined by straight trajectories, slow movements, and small angle inclinations. Therefore paths and trajectories are commonly tracked slowly and carefully for avoiding to take the system out of its stability zone.

Taking advantage of the quaternion-based controllers detailed in the previous section, autonomous and semi-autonomous navigation algorithms were conceived for different quadrotor flight scenarios. The characteristics of quaternions simplify the task of merging navigation schemes with quadrotor controllers, enhancing system performance and robustness against disturbances and uncertainties. Numerical simulations and real-world experiments are performed to validate every proposal.

The contents of this chapter are organized as it follows. First, Section 4.1 introduces a semi-autonomous navigation scenario where a quadrotor tracks gesture-based user attitude commands, then, an autonomous navigation algorithm for a fleet of quadrotors using a distributed path planning approach is detailed in Section 4.2, finally, a novel aggressive deployment strategy for quadrotors is presented in Section 4.3, here, a combination of autonomous and semi-autonomous algorithms is proposed to autonomously recover and stabilize a quadrotor which is launched through the air under any initial conditions and with its motors turned off.

4.1 Safe quadrotor navigation using arm commands

Consider that a quadrotor is piloted using arm gestures from its pilot, measured using inertial sensors contained in a wearable armband, with the objective of providing an intuitive and safe flight experience. If the vehicle follows blindly all the input device signals, aggressive unintended rotations could arrive from an inexperienced user, which might be dangerous for the vehicle and people around.

4.1.1 Attitude gestures

Let four coordinate systems be represented as $\mathcal{I}$, $\mathcal{M}_0$, $\mathcal{M}$, and $\mathcal{B}$, which locate respectively at the global fixed frame, the initial bracelet pose, the rotating coordinates of the pilot's forearm, and the quadrotor body frame, see Figure 4.1, such that the rotation between $\mathcal{I}$ and $\mathcal{M}_0$ is given by a constant quaternion

$$\boldsymbol{q}_{\mathcal{M}_0} = \left(\cos\frac{\psi_0}{2} + \begin{bmatrix} 0 \\ 0 \\ \sin\frac{\psi_0}{2} \end{bmatrix}\right) \otimes \left(\cos\frac{\theta_0}{2} + \begin{bmatrix} 0 \\ \sin\frac{\theta_0}{2} \\ 0 \end{bmatrix}\right) \otimes \left(\cos\frac{\phi_0}{2} + \begin{bmatrix} \sin\frac{\phi_0}{2} \\ 0 \\ 0 \end{bmatrix}\right), \tag{4.1}$$

where θ_0, ϕ_0, and ψ_0 represent the forearm's initial pitch, roll, and yaw angles. Then, the rotation from $\mathcal{I}$ to $\mathcal{M}_0$ is defined by

$$\boldsymbol{q}_{\mathcal{M}}(t) = \left(\cos\frac{\psi_{\mathcal{M}}}{2} + \begin{bmatrix} 0 \\ 0 \\ \sin\frac{\psi_{\mathcal{M}}}{2} \end{bmatrix}\right) \otimes \left(\cos\frac{\theta_{\mathcal{M}}}{2} + \begin{bmatrix} 0 \\ \sin\frac{\theta_{\mathcal{M}}}{2} \\ 0 \end{bmatrix}\right) \otimes \left(\cos\frac{\phi_{\mathcal{M}}}{2} + \begin{bmatrix} \sin\frac{\phi_{\mathcal{M}}}{2} \\ 0 \\ 0 \end{bmatrix}\right), \tag{4.2}$$

where $\phi_{\mathcal{M}}, \theta_{\mathcal{M}}$, and $\psi_{\mathcal{M}}$ symbolize the time variant Euler angles from $\mathcal{M}_0$ to $\mathcal{M}$.

The quadrotor attitude commands are defined as a function of the angular variations experienced on the user's forearm $\boldsymbol{q}_{\mathcal{M}}(t)$, i.e., if the user tilts his arm, the quadcopter should undergo a correspondent motion along the same axis. This allows the user to get a more natural feel on how the drone will behave since it will essentially mimic the user's arm orientation.

It is, however, undesired to attain a direct correspondence between $\boldsymbol{q}_{\mathcal{M}}(t)$ and that one of the drone. For example; an inexperienced user might tilt his arm excessively, resulting in a vertical pose, by doing this, the drone would turn perpendicularly to the ground and crash.

Therefore, it is necessary to implement some safe movements restrictions to prevent the drone from performing risky rotations. Introducing a quaternion reference as

$$\boldsymbol{q}_{\mathcal{R}}(t) = \left(\cos\frac{\psi_{\mathcal{R}}}{2} + \begin{bmatrix} 0 \\ 0 \\ \sin\frac{\psi_{\mathcal{R}}}{2} \end{bmatrix}\right) \otimes \left(\cos\frac{\theta_{\mathcal{R}}}{2} + \begin{bmatrix} 0 \\ \sin\frac{\theta_{\mathcal{R}}}{2} \\ 0 \end{bmatrix}\right) \otimes \left(\cos\frac{\phi_{\mathcal{R}}}{2} + \begin{bmatrix} \sin\frac{\phi_{\mathcal{R}}}{2} \\ 0 \\ 0 \end{bmatrix}\right), \tag{4.3}$$

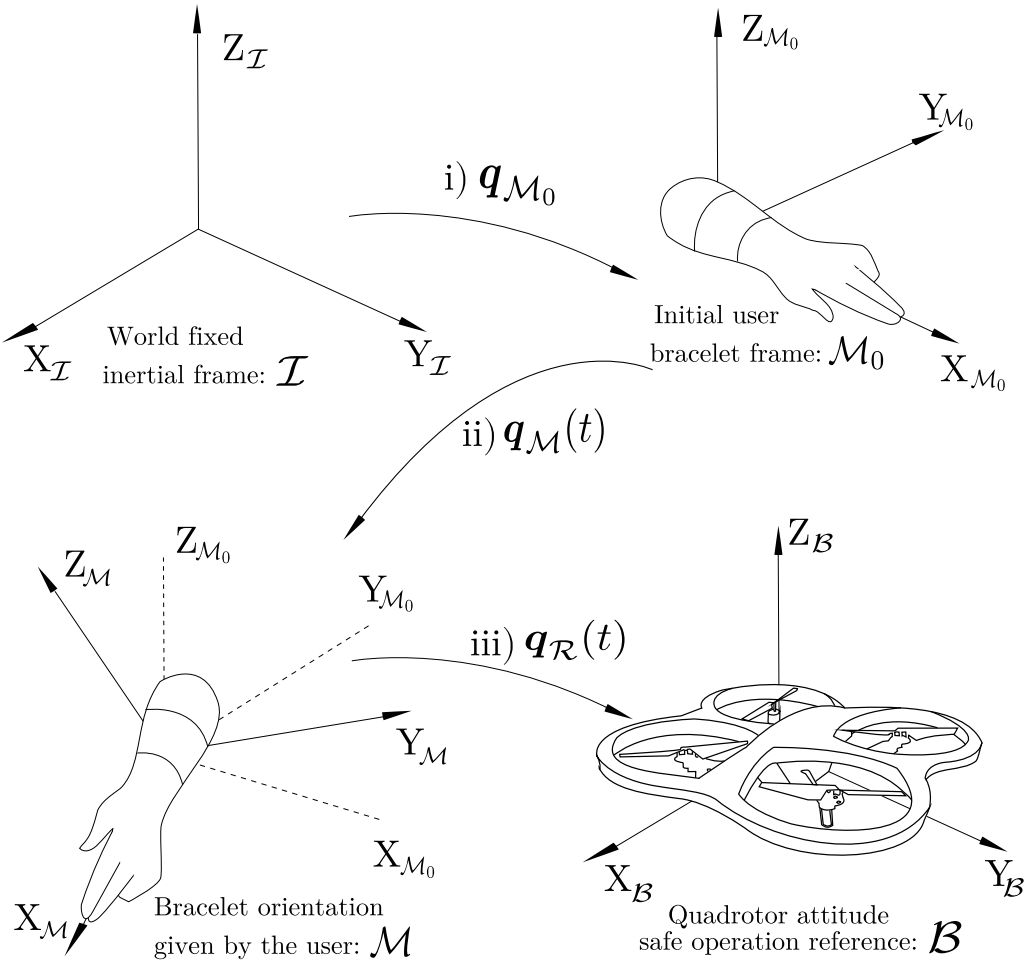


FIGURE 4.1: Reference frames for quadrotor attitude reference.

where

$$\theta_{\mathcal{R}}(\theta_{\mathcal{M}}) = \begin{cases} \theta_{\mathcal{M}}^3 S_\theta & ; \quad \theta_{\mathcal{M}} \leq \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} \\ \theta_{\max} & ; \quad \theta_{\mathcal{M}} > \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} \end{cases}, \tag{4.4}$$

$$\phi_{\mathcal{R}}(\phi_{\mathcal{M}}) = \begin{cases} \phi_{\mathcal{M}}^3 S_\phi & ; \quad \phi_{\mathcal{M}} \leq \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} \\ \phi_{\max} & ; \quad \phi_{\mathcal{M}} > \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} \end{cases}, \tag{4.5}$$

$$\psi_{\mathcal{R}}(\psi_{\mathcal{M}}) = \psi_{\mathcal{M}}, \tag{4.6}$$

being $\theta_{\max}$ and $\phi_{\max}$ the maximum tilt allowed on the reference angles (θ_R and ϕ_R) respectively, while S_θ, $S_\phi < 1$ are tunning parameters that improve the sensitivity of the command along them, and must be selected such that

$$\sqrt[3]{\frac{\phi_{\max}}{S_\phi}}, \ \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} < 1. \tag{4.7}$$

Note from (4.4) and (4.5) that cubic relationships are followed by the attitude reference with respect to the forearm rotation, nevertheless (4.7) implies that the

attitude gesture signals are cubically attenuated, see Figure 4.2.

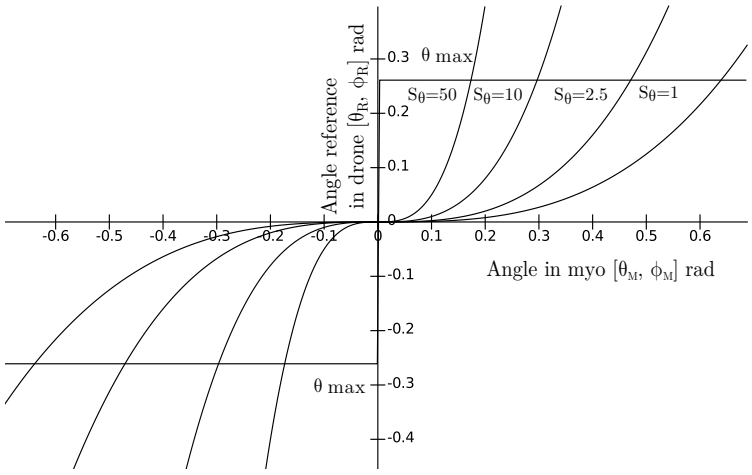


FIGURE 4.2: From left to right, greater values $\theta_{\mathcal{M}}$ and $\phi_{\mathcal{M}}$ are required to reach tilt limit a as sensitivity values S_θ and S_ϕ decrease.

These cubic expressions also mean that higher values for these parameters translate into the quadcopter performing more aggressive movements in response to the user's arm motion; while lower values will cause the quadcopter to perform slower in response to the same arm motion, the tilting will also be diminished as the sensitivity values approach zero until the drone remains totally unresponsive.

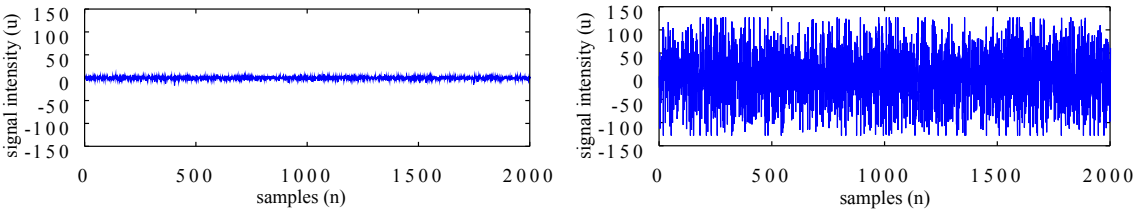
The range of rotation, as well as the velocity at which tilt limits $\theta_{\max}$ or $\phi_{\max}$ will be reached depend on the sensitivity parameters S_θ and S_ϕ, see Figure 4.2. These parameters are tuned so as to meet user's comfort and experience: advanced users might prefer higher sensitivity values for sharper, faster maneuvers while beginners may prefer lower values for slower, easier to follow movements.

4.1.2 Electrical references from skeletal muscles

For this work, a pose will be defined as any combination in the fingers and wrist disposition, see Figure 4.3.

FIGURE 4.3: Examples of poses, respectively: a fist, fingers spread, folded wrist. Image via https://support.getmyo.com

Consider the armband contains eight electromyographic sensors, such that each one detects electrical activity in the superficial layers of the arm muscles and translates its intensity into an 8-bit value ranging from -127 to 128, see Figure 4.4.



(A) A single EMG sensor for a resting arm. (B) EMG signal when the muscle is tighten.

FIGURE 4.4: Resting and tense muscular EMG measurements for one sensor

The combination and interpretation of these measurements can be used to detect five manufacturer predefined poses (wave in, wave out, double tap, fist, rest and spread fingers), or be used as raw data as well.

Some of the predefined poses can be, at times, triggered incorrectly, mainly when the armband is passed from one user to another. The *double-tap* predefined pose, which is performed by tapping the middle finger and thumb in quick succession, was deemed as the most reliable of these; since it was always triggered when the user intended to. A second pose was required for this proposal, having ran out of reliable predefined poses a *custom pose* was proposed.

Custom Gesture The custom pose in Figure 4.5 works by measuring and adding up the electrical raw muscle activity in the user's arm or fingers collected from all eight sensors of the array. Once it surpasses a certain threshold it is checked to ascertain the intentionality of the command and then the signal is sent. This is the command used to set the custom rotation reference, which is mostly used when the user drifts too much apart from his starting position.

FIGURE 4.5: Example of the custom pose, fingers are to be held tightly in place as if firmly holding a spherical object.

The raw measurement of the total muscular activity in the subject's arm can be computed by

$$\sigma_{EMG}(t) = \sum_{i=1}^{8} |EMG_i(t)|, \tag{4.8}$$

where $EMG_i(t)$ is the measure for each electrode.

Let b symbolize the muscular activity threshold and

$$f_1(t) = \frac{1 + \text{sign}(\sigma_{EMG}(t) - b)}{2} \tag{4.9}$$

be the function that verifies whether or not the raw muscular activity in the user's arm has surpassed threshold b.

Define c and d as parameters that will verify the intentionality of the gesture. c is defined as a whole number greater than zero, and d as a rational number such that $0 < d < 1$. by checking c number of samples from (4.9) at different instants of time, i.e.

$$f_2(t) = \sum_{i=0}^{c} f_1(t - i). \tag{4.10}$$

Besides, parameter d effectively refers to the least amount of times that threshold b has to be surpassed within c samples in order to be considered as a valid pose. This can be explained in the following:

$$f_3(t) = \text{sign}(f_2(t) - dc), \tag{4.11}$$

signifying $f_3(t) = 1$ that the pose has been successfully triggered and $f_3(t) = -1$ or $f_3(t) = 0$ means that the pose has not been triggered.

4.1.3 Input based on a gesture sequence

Once the bracelet signals are processed, the next goal is to apply them for controlling the UAV. The transition between the user's arm orientation and drone attitude reference is relatively natural; however, if the user wants to perform more specific tasks, it is necessary to device a different set of gestures.

At Heudiasyc lab, in collaboration with CINVESTAV Saltillo, a nonlinear controller was developed to perform quadrotor pirouettes in closed loop, see [94]. The aim in the arm command algorithm is to indicate the drone when to perform acrobatic maneuvers using the user gestures.

A sequence triggered by a *double-tap* pose was chosen, consisting in two stages: After the double-tap gesture, the user has L_{t1} milliseconds to either perform the follow up for a single, double or triple quadrotor pirouette, see Figure 4.6.

If the user begins a gesture sequence during L_{t1} an additional L_{t2} milliseconds are provided during which the rest of the sequence is to be performed, this amounts $L_{t1} + L_{t2}$ milliseconds during which the user has to perform the whole sequence.

The quadrotor will perform the intended mission immediately thereafter, or, it will remain hovering if the sequence fails to be performed correctly.

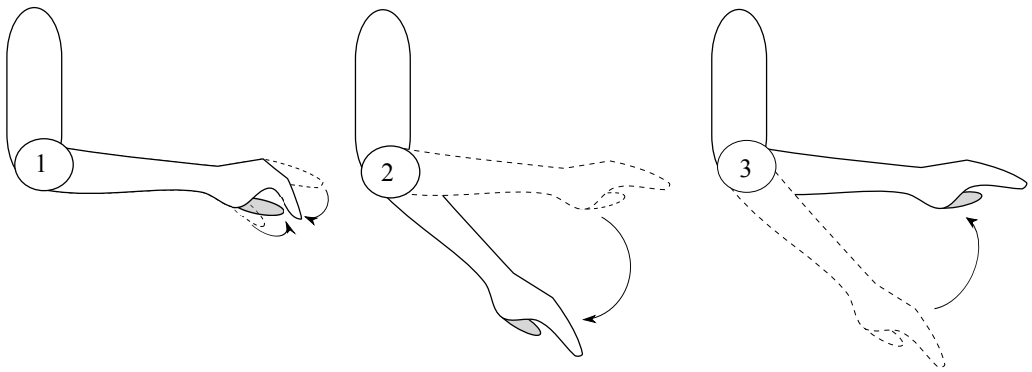


FIGURE 4.6: Example of the gesture sequence to be performed in order to trigger a triple loop command.

The method used to define each sequence is the following; first, since the gesture is performed by tilting the forearm upwards and downwards a vertical boundary $b_{\theta_{\mathcal{M}}}$ was set around the initial frame $\mathcal{M}_0$. Gestures can then be identified by the way the user's arm orientation transitions between these regions by following

$$f_{G1}(t) = \frac{1 + \text{sign}(\theta_{\mathcal{M}}(t) - b_{\theta_{\mathcal{M}}})}{2} - \frac{1 - \text{sign}(\theta_{\mathcal{M}}(t) + b_{\theta_{\mathcal{M}}})}{2}. \tag{4.12}$$

Define a scalar function, which depends on discrete evaluations of (4.12) as

$$\rho = \sum_{i=0}^{2} f_{G1}(t_i) + f_{G1}(t_{i-1}), \tag{4.13}$$

where t_i is the time instant where the value of $f_{G2}(t)$ changes, marking the transition between any two adjacent poses. The value of (4.13), is finally assigned to specific tasks for the drone according to the following table:

ρ **value**	**Gesture**	**Number of loops**
$\rho = 0$	No gesture detected	One Loop
$\rho = 2$	Up, down, up	Two loops
$\rho = -2$	Down, up, down	Three loops
otherwise	Unassigned gesture	One Loop

TABLE 4.1: Gesture assignment to detection function

4.1.4 Safe Human-UAV Interaction

Human error is always a possibility in semi-autonomous navigation. In order to decrease its impact, safety measures were developed to allow for more intuitive commands and a more enjoyable user experience. These also ensure safety by avoiding unintended maneuvers that might result in crashes.

IMU based safety measures

An additional orientation-based safety measure is implemented by defining two boundaries ϕ_{lim} and θ_{lim} in the event when the user would drop his arm. Should this happen, the orientation reference for $\theta_{\mathcal{R}}$ and $\phi_{\mathcal{R}}$ will be set to 0 to ensure safe hovering, this is achieved by enhancing (4.4), (4.5) and (4.6) as

$$\theta_{\mathcal{R}}(\theta_{\mathcal{M}}) = \begin{cases} \theta_{\mathcal{M}}^3 S_\theta & ; \quad \theta_{\mathcal{M}} \leq \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} \\ \theta_{\max} & ; \quad \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} < \theta_{\mathcal{M}} \leq \theta_{\text{lim}} \\ 0 & ; \quad \theta_{\text{lim}} < \theta_{\mathcal{M}} \end{cases}, \tag{4.14}$$

$$\phi_{\mathcal{R}}(\phi_{\mathcal{M}}) = \begin{cases} \phi_{\mathcal{M}}^3 S_\theta & ; \quad \phi_{\mathcal{M}} \leq \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} \\ \phi_{\max} & ; \quad \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} < \phi_{\mathcal{M}} \leq \phi_{\text{lim}} \\ 0 & ; \quad \phi_{\text{lim}} < \phi_{\mathcal{M}} \end{cases}, \tag{4.15}$$

$$\psi_{\mathcal{R}}(\psi_{\mathcal{M}}) = \psi_{\mathcal{M}}, \tag{4.16}$$

where ϕ_{lim} and θ_{lim} are the thresholds where the user's arm is considered to be beyond the acceptable, see Figures 4.7 and 4.8.

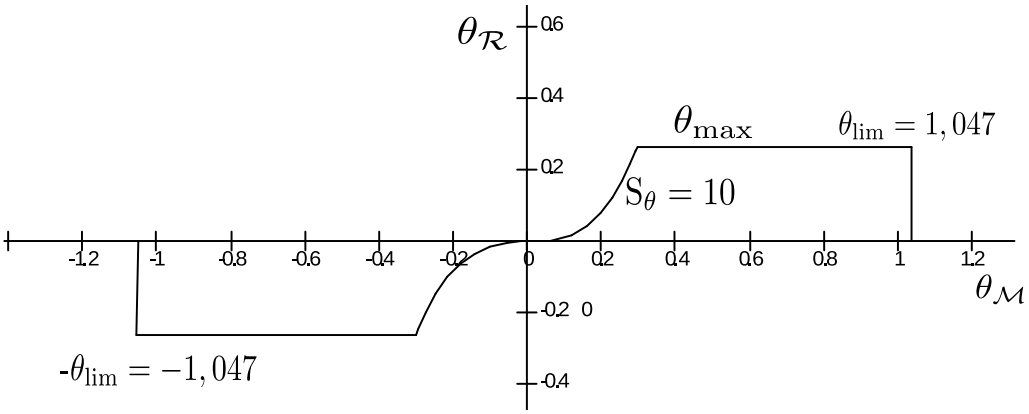


FIGURE 4.7: $\phi_{\mathcal{R}}$ with safety limit put in 1.047 rad (60°).

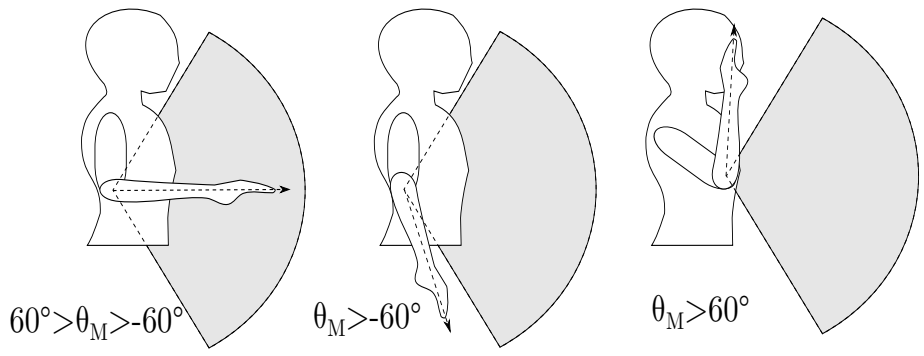


FIGURE 4.8: $\theta_{\mathcal{M}}$ orientation references considered within the accepted range.

EMG based safety measures

The electromyographic capabilities of the device were also used to make it safer. As previously mentioned, there are 7 pre programmed muscular poses p each assigned a default numeric value see Figure 4.9 and Table 4.2.

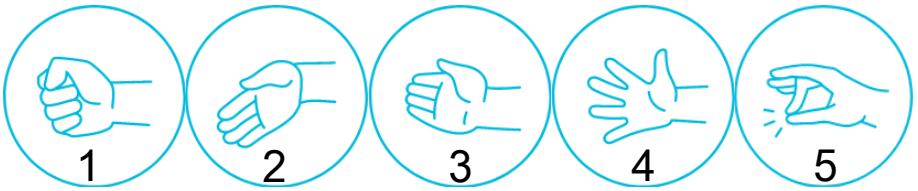


FIGURE 4.9: Default preprogrammed poses. From left to right poses 1 through 5 respectively. Image via https://support.getmyo.com

$\mathcal{P}$ numeric value	pose name	pose action
0	resting position	neutral reference
1	fist	unused
2	wave in	unused
3	wave out	unused
4	spread fingers	unused
5	double tap	unlock device, trigger loop
6	unknown	lock device if not worn

TABLE 4.2: Predefined poses and correspondent action.

Furthermore, the bracelet can detect whether or not is it being worn by a user, and it will remain locked $l(t)$ until the unlocking pose is performed (double tap pose while its locked). For this proposal: 6 (unknown) will determine whether or not the bracelet is being worn, 5 (double tap) will be used as the unlocking pose (as well as begin loop command), and 0 (resting position) will be used a neutral reference. Therefore the chosen poses for our application are: $\mathcal{P} = \{0, 5, 6\}$.

Once communication between the program and the bracelet is enabled, the bracelet must be unlocked by performing the double tap pose according to (4.17).

$$
\begin{aligned}
l(t_0) &= \begin{cases} 0 & \text{for } \mathcal{P} = 5 \\ 1 & \text{for } \mathcal{P} \neq 5 \end{cases}, \\
l(t) &= \begin{cases} 0 & \text{for } \mathcal{P} \neq 6 \\ l(t_0) & \text{for } \mathcal{P} = 6 \end{cases}.
\end{aligned}
\tag{4.17}
$$

If the user suddenly removes the bracelet data transmission is stopped, then θ_R and ϕ_R angles are set to zero while ψ_R remains in the last reference recorded such that

$$\theta_{\mathcal{R}}(t) = \begin{cases} l(t)\theta_{\mathcal{M}}^3 S_\theta & ; \quad \theta_{\mathcal{M}} \leq \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} \\ l(t)\theta_{\max} & ; \quad \sqrt[3]{\frac{\theta_{\max}}{S_\theta}} < \theta_{\mathcal{M}} \leq \theta_{\lim} \\ 0 & ; \quad \theta_{\lim} < \theta_{\mathcal{M}} \end{cases}, \tag{4.18}$$

$$\phi_{\mathcal{R}}(t) = \begin{cases} l(t)\phi_{\mathcal{M}}^3 S_\theta & ; \quad \phi_{\mathcal{M}} \leq \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} \\ l(t)\phi_{\max} & ; \quad \sqrt[3]{\frac{\phi_{\max}}{S_\phi}} < \phi_{\mathcal{M}} \leq \phi_{\lim} \\ 0 & ; \quad \phi_{\lim} < \phi_{\mathcal{M}} \end{cases}, \tag{4.19}$$

$$\psi_{\mathcal{R}}(t) = l(t)\psi_{\mathcal{M}} + (1 - l(t))\psi_{\mathcal{M}}(t-1). \tag{4.20}$$

Additional safety measures As a required safety measure to ensure the laboratory material integrity, a joystick command device is always kept at hand by a more experienced operator who can, at given moment, override the armband navigation commands in order to deliver the quadrotor from hazard.

This safety layer is included by defining the actual reference quaternion $\boldsymbol{q}_d$, which is introduced the attitude controller from (3.43), where the orientation error is redefined as $\vec{\vartheta}_e = 2\ln(\boldsymbol{q}_d^* \otimes \boldsymbol{q})$ with:

$$\boldsymbol{q}_d(t) = \begin{cases} \boldsymbol{q}_{\text{joy}} & , \quad \text{override on}, \\ \boldsymbol{q}_{\mathcal{R}} & , \quad \text{override off}, \end{cases} \tag{4.21}$$

where $\boldsymbol{q}_{\text{joy}}$ represents a quaternion computed from a dual-joystick device, following remote piloting conventions.

4.1.5 Experimental results

The safe control scheme for quadcopter navigation was programmed using the Fl-AIR (Framework libre AIR) libraries [95]. The attitude commands were inputted by a sensor-equipped bracelet [1]. This device is provided by three main sensors; a gyroscope, an accelerometer, and an array of eight electromyographic (EMG) sensors. Custom compatibility for Linux was based on the PyoConnect library[2].

The armband communicates its sensor data via bluetooth into a PC. This data is handled by a program which processes the attitude references and the EMG measures interpretation routines.

Data output from this algorithm is handed over to the internal UAV controller via UDP (User Datagram Protocol) using wireless communications. This program handles the commands for the drone attitude and contains the control laws needed for its operation, see Figure 4.10.

[1] https://www.myo.com/techspecs
[2] http://www.fernandocosentino.net/pyoconnect/

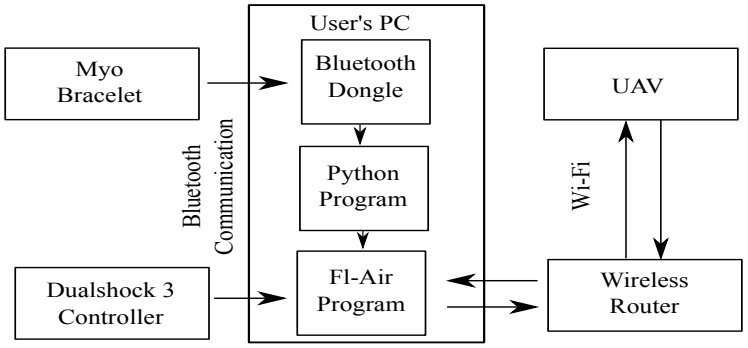
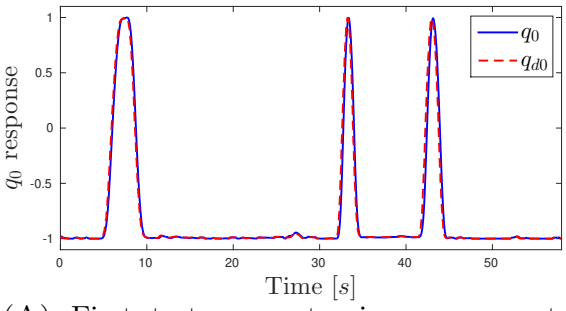


FIGURE 4.10: Communication process involved in the armband-UAV interface.

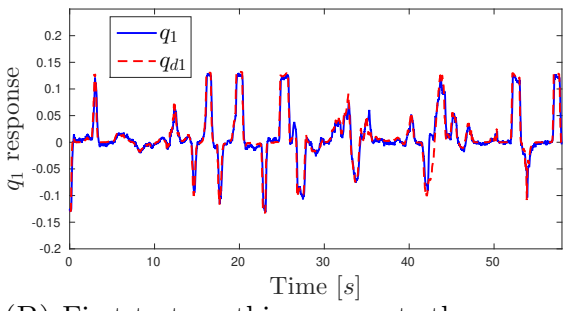
Attitude safety restrictions: The first flight test consisted on controlling the vehicle attitude using arm gestures, at some moments, the user performed high inclinations, which could result in dangerous movements if managed incorrectly. However, the proposed algorithm restricts such rotations by following (4.18) and (4.19), where the maximum angles were set at $\theta_{\text{lim}} = \phi_{\text{lim}} = 15°$.

At the beginning and end of the test, rotations around the user's body axis were performed, resulting in multiple 360° yaw rotations, since the user movements are not restricted over the z axis, the given reference is tracked integrally.

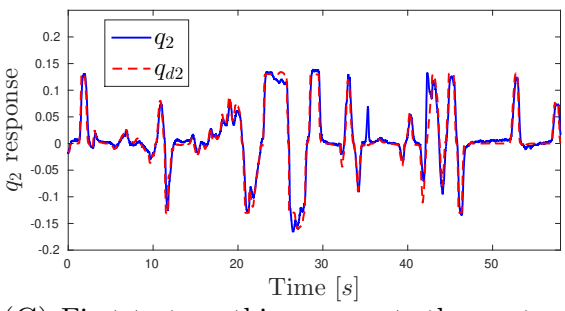
Figure (4.11) represents the quadrotor attitude behavior. The effects of the safety pitch and roll bounding, translated into quaternion references can be appreciated in Figures 4.11B and 4.11C, while Figure 4.11D indicates a complete rotation over the yaw angle by a continuous change of the q_3 value from 1 to -1.



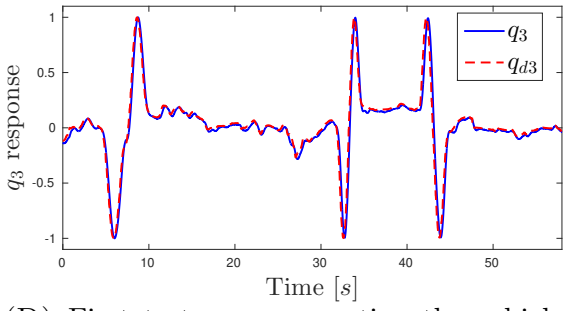
(A) First test q_0 quaternion component, switching its value from -1 to 1 each time a full 360° rotation is performed.



(B) First test q_1, this represents the x component of the quaternion.



(C) First test q_2, this represents the quaternion y component.



(D) First test q_3, representing the vehicle rotation over its z axis.

FIGURE 4.11: Quadrotor attitude quaternion tracking

To give a better illustration of the attitude behavior, the Euler representation for the same test can be observed in Figures 4.12 to 4.14. Notice how the saturation occurs in θ_d and ϕ_d whenever the 15 degrees (0.261 radians) is reached. A close-up view of the pitch and roll signals is included to better illustrate such behavior.

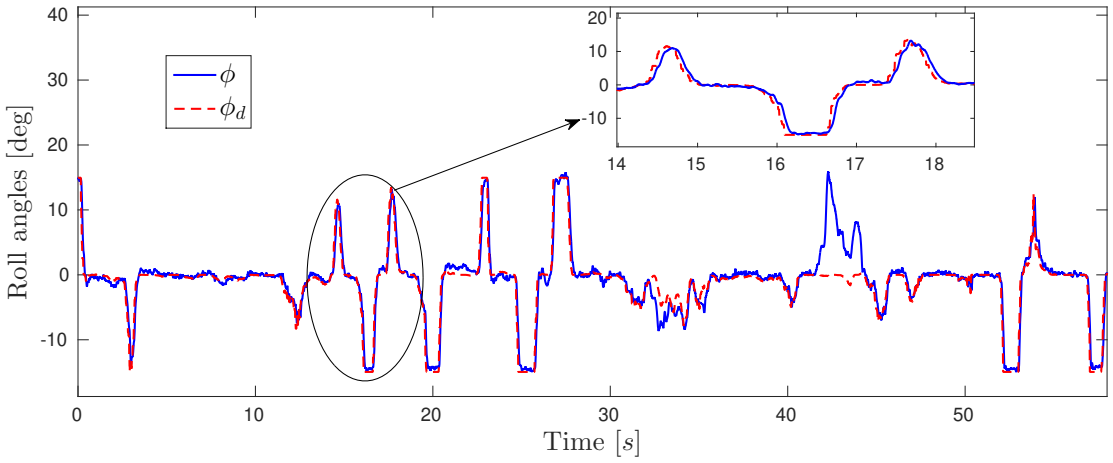


FIGURE 4.12: Roll response using ϕ_d from the bracelet, reference signal is bounded to safe values between $\pm 15°$.

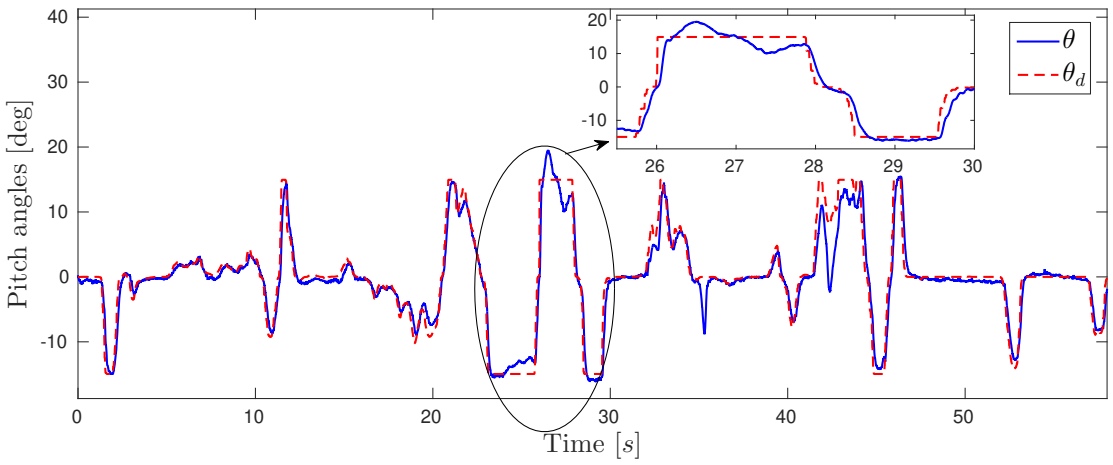


FIGURE 4.13: Pitch response using θ_d from the bracelet, imposing restrictions at $\pm 15°$ values.

Discontinuities in the roll and pitch angles are never present due to the imposed constraints in the algorithms, as well as those from natural human anatomy. However, discontinuities in yaw movements using Euler representation appear whenever a full turn is performed, since the value of this angle suddenly changes from 180 to -180 deg, as can be seen in Figure 4.14. However, when using quaternion representation these discontinuities are not present, as seen in Figure 4.11D.

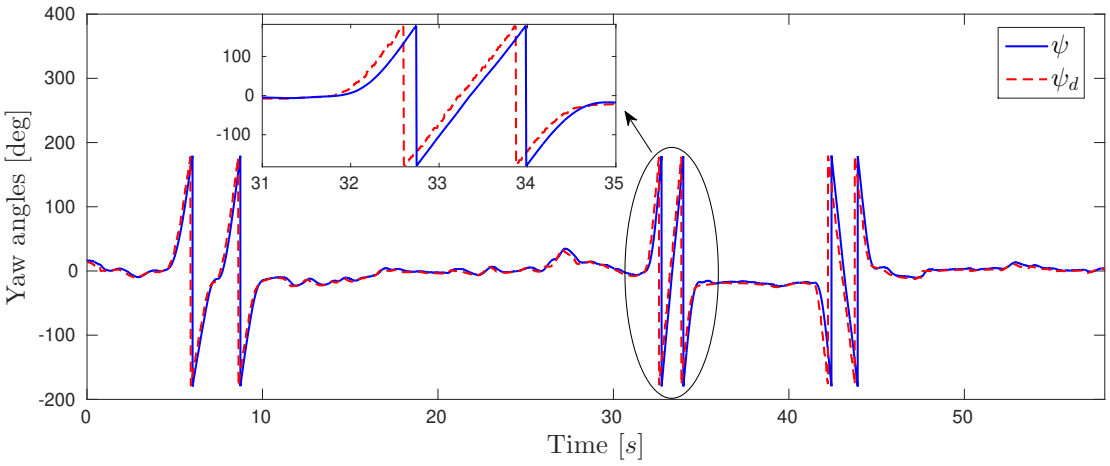
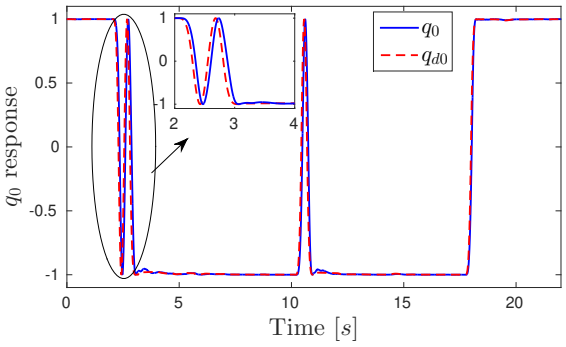
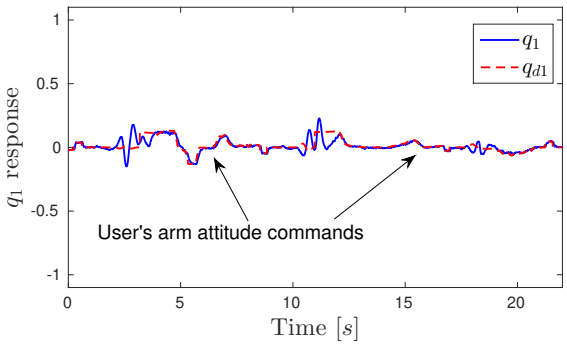


FIGURE 4.14: Yaw response using ψ_d from the bracelet. Notice how discontinuities appear in yaw Euler representation when a full 360° turn is performed.

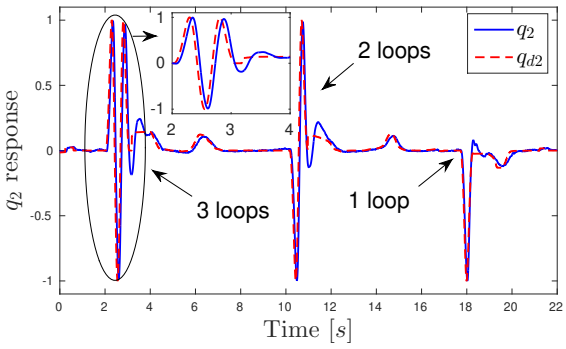
EMG commands: In the second test, the user performed muscular gestures corresponding to a triple-loop, a double-loop, and a single loop consequently, Figure 4.15 illustrates attitude quaternion performance of this experiment.



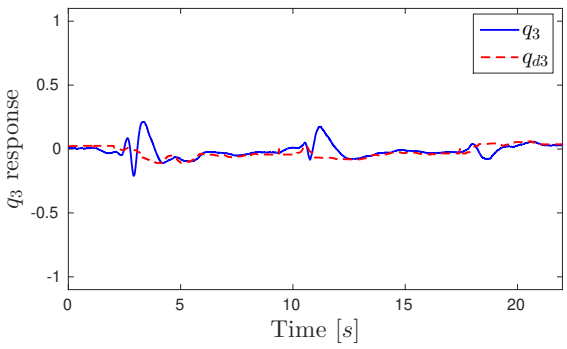
(A) Notice how the sign of q_0 is inverted whenever a loop is performed



(B) Quaternion component q_1, following only the piloting references.



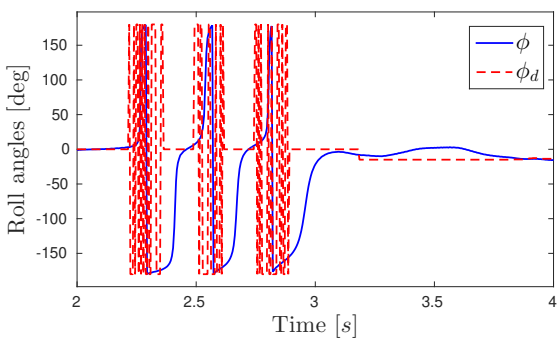
(C) q_2 component, sinusoidal oscillations indicate when a pirouette is performed.



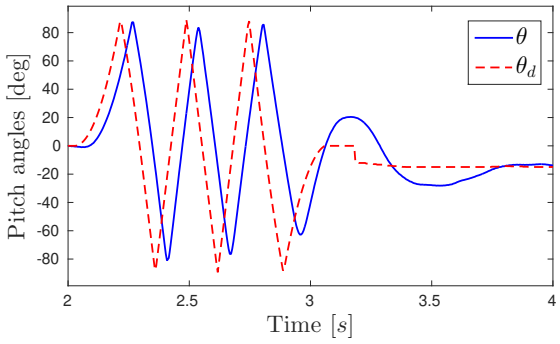
(D) Quaternion component q_3, maintaining the bracelet reference.

FIGURE 4.15: Quadrotor attitude quaternion tracking in looping experiment

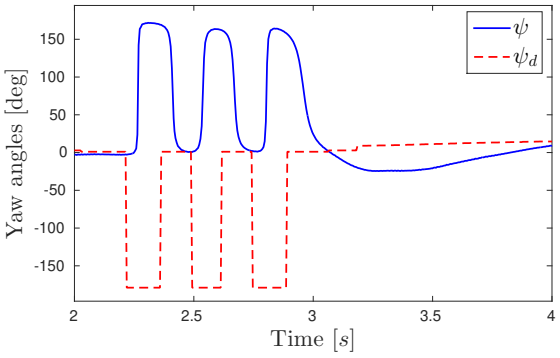
In contrast, close-ups of the Euler angles equivalent responses of this experiment are depicted in Figure 4.16, here, two undesired effects are revealed, since the pirouettes are performed over the y axis, Figures 4.16A and 4.16C reveal a Gimball-lock effect, where multiple roll and yaw angles are possible when the pitch angle reaches a singular value. Figure 4.16B illustrates a triple pitch loop, passing through discontinuities three consecutive times.



(A) Close up of the ϕ euler angle during the triple loop maneuver. Notice the discontinuities in ϕ and the Gimball lock in ϕ_d.



(B) Close up of the θ angle triple loop maneuver, a triangular ramp behavior is the result of the quaternion to Euler angles conversion.



(C) Close up of the ψ angle. Notice how ψ diverges from ψ_d due to the quaternion to Euler angles transformation.

FIGURE 4.16: Euler angles representation of the quadrotor attitude during looping tests

A video showing the complete test can be watched at:
`https://www.youtube.com/watch?v=F4fjOOFGfKQ`

4.2 Path planning for a fleet of quadrotors

In this section, a control approach for coordinated flight of a fleet composed of three drones will be presented. In the context of this section, the term *agent* is interchangeable with *UAV*. Each agent is equipped with individual position and attitude controllers as described by (2.40), (3.35), and (3.43), this algorithm was developed in the context of a collaboration with researchers from CRAN at the Université de Lorraine.

A fleet consists of a set of agents $N = 1, ..., n$ where n is the number of vehicles. Information exchanged among agents are modeled by means of Graph Theory. Let a set of elements $d_{ij}(t)$ represent the distance between two agents i and j, such that a matrix $\partial(t) \in \mathbb{R}^{N \times N}$ is formed. Similarly, the desired distances between agents are defined by d^d_{ij}, and are used to construct a matrix symbolized by ∂_d., while the safety distance is denoted as c (where $c < l$) and l symbolizes the scope of the formation.

Define for each i agent a set of neighbors $\Xi_i(t)$ such that:

$$\Xi_i(t) = \{j \in \mathbb{N} : \|\vec{p}_j(t) - \vec{p}_i(t)\| \leq l\}, \tag{4.22}$$

where $\Xi_i(t)$ is called the metrical neighborhood of the robot i, and $\vec{p}_i(t)$ symbolizes its position. Then, a topological neighborhood is considered where only λ elements of the set $\Xi_i(t)$ from the closest node i are taken into account. The λ nodes represent a new $\Xi_i(t)'$ set with $\Xi_i(t)' \subset \Xi_i(t)$.

The control objectives are then:

1. To bring the fleet from an initial geometrical configuration $\partial(t_0)$ to a desired geometrical configuration ∂_d

$$\lim_{t \to +\infty} \partial(t) = \partial_d. \tag{4.23}$$

2. Ensuring target points tracking, which position is defined as $\vec{p}_T$

$$\forall i \in N, \lim_{t \to +\infty} d_{ip}(t) = d^d_{ip}. \tag{4.24}$$

   where $d_{ip}(t)$ and d^d_{ip} are respectively the real and desired distance between agent i and the target point.

3. To avoid collisions between interacting agents

$$\forall i, j \in N, i \neq j : \|\vec{p}_i(t) - \vec{p}_j(t)\| > c. \tag{4.25}$$

The issue is formulated as an online optimization problem where a positive scalar cost functions $\Lambda(t)$ is constructed and divided amongst all the agents in such a way that its minimum is obtained when the fleet reaches the defined objectives. The main idea is to find at each step time the best displacement $\vec{h}^*_i(t)$ which yields the minimum cost function:

$$\vec{h}^*_i(t) \to \min(\Lambda(t)), \tag{4.26}$$

and

$$\vec{p}_{di}(t + t_\delta) = \vec{p}_i(t) + \vec{h}^*_i(t). \tag{4.27}$$

where $\vec{p}_{di}(t + t_\delta)$ is the desired reference input of agent i, representing a position reference in the global inertial frame.

Figure 4.17 illustrates the general control scheme where a distributed planning path strategy based on Particle Swarm Optimization (PSO) algorithm is proposed to achieve the goal.

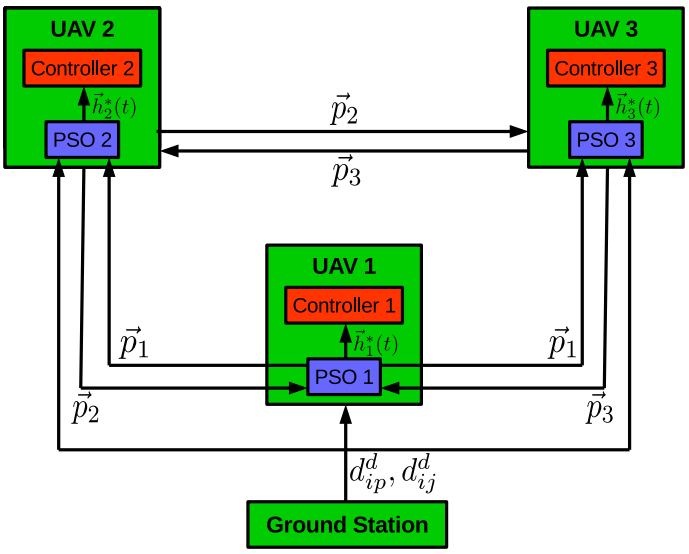


FIGURE 4.17: Distributed path planning structure

4.2.1 Distributed path planning design

The proposed approach is based on a distributed path planning, where each vehicle is equipped with a part of an optimization algorithm. The trajectories computing is therefore not carried out in a central unit but is distributed on a vehicle team, contrary to controllers that are completely decentralized.

Control algorithm: Each quadrotor is equipped with an internal controller, which is designed by introducing (4.27) into (3.31) and (3.30) such that a local control force is computed

$$\vec{F}_{u_i} = -K_{pt}\ \vec{h}_i^*(t) - K_{dt}\ \dot{\vec{p}}_i - m_i\vec{g}, \tag{4.28}$$

which yields an attitude controller given by

$$\begin{aligned}\vec{\tau}_i = &-2K_{pr}\ln\left(\pm\left(\sqrt{\frac{1+\frac{\vec{F}_{u_i}}{||\vec{F}_{u_i}||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}} - \frac{\frac{\vec{F}_{u_i}}{||\vec{F}_{u_i}||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}}{\left|\left|\frac{\vec{F}_{u_i}}{||\vec{F}_{u_i}||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}\right|\right|}\sqrt{\frac{1-\frac{\vec{F}_{u_i}}{||\vec{F}_{u_i}||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}}\right)\otimes\boldsymbol{q}_i\right)\\ &- K_{dr}\vec{\Omega}_i + \vec{\Omega}_i\times J_i\,\vec{\Omega}_i\ ,\end{aligned} \tag{4.29}$$

where $K_{pt}, K_{dt} \in \mathbb{R}^{3\times3}$ denote positive control gains, $\dot{\vec{p}}_i$, m_i, J_i, $\boldsymbol{q}_i$ and $\vec{\Omega}_i$ respectively represent the translational speed, mass, inertia matrix, attitude quaternion and angular velocity of quadrotor i.

4.2.2 Construction of the Cost function

A positive cost function $\Lambda(t)$ is designed for the fleet of drones, and then distributed amongst all of them such that each agent i optimizes a part of the function using its own information and that of its k neighbors. In the context of motion planning, such function is built by taking into account the distance between agents and the location of an element considered as a target for the fleet point $\vec{p}_T$. Its construction also takes into account information to ensure collision avoidance as

$$\Lambda(t) = \sum_{i=1}^{n} \Lambda_i(t), \tag{4.30}$$

$$\Lambda_i(t) = \rho\Big(\|\vec{p}_T - [\vec{p}_i(t) + \vec{h}_i(t)]\| - d_{ip}^d\Big) + \sum_{j=1}^{k} a_{ij}(t)\Big(\|\vec{p}_j(t) - [\vec{p}_i(t) + \vec{h}_i(t)]\| - d_{ij}^d\Big),$$

where

$$a_{ij}(t) = 1 + exp\left(\frac{c - d_{ij}(t)}{\sigma}\right), \tag{4.31}$$

with $i \neq j$, $k = \text{card}(\Xi_i'(t))$, $\rho >> 1$ and $\sigma \neq 0$ denote constant gains, while d_{ip}^d represents the desired distance between agent i and target point.

The choice of ρ depends essentially on the number and type of neighbors of each agent and plays a role on the rate of convergence towards the target,while σ depends on how strongly the agents must react to avoid collisions. A compromise should be found between target monitoring, collision avoidance, and training control. Since an homogeneous fleet is considered in this work, ρ has the same value for all the vehicles.

The main goal is to find the best vector $\vec{h}_i^*(t)$ for each vehicle i minimizing the cost function $\Lambda_i(t)$ such that

$$\forall i \in N : \lim_{t\to\infty} \Lambda_i(t) = 0 \Rightarrow \begin{cases} \lim_{t\to\infty} \partial(t) = \partial_d \\ \forall i \in N, \lim_{t\to+\infty} d_{ip}(t) = d_{ip}^d \\ \forall i, j \in N, i \neq j : \|\vec{p}_i(t) - \vec{p}_j(t)\| > c \ , \end{cases} \tag{4.32}$$

where $\vec{h}_i^*(t)$ represents the desired displacement relative to the local frame of agent i between two instants t and $t + t_\delta$. The reference trajectory at time $t + t_\delta$ for agent i becomes

$$\vec{h}_i^*(t) = \vec{p}_{di}(t + t_\delta) - \vec{p}_i(t). \tag{4.33}$$

During the evolution of the vehicles in the space, non-collision of the agents should be ensured. This constraint is introduced in $\Lambda_i(t)$ through function $a_{ij}(t)$ which is all the greater when $d_{ij}(t) < c$ and $a_{ij}(t) \to 1$ when $a_{ij}(t) >> c$. Therefore, each agent i is more likely to favor the values of $\vec{h}_i(t)$ which avoids the need for the distances $d_{ij}(t) < c$ and minimizes the cost function $\Lambda_i(t)$, thanks to the PSO algorithm.

In order to have a more stable behavior of the fleet in the proximity of the minimum, a minimum value of the cost function $\Lambda_{i\text{MIN}}$ is considered satisfactory. In this case,

when $\Lambda_i(t) < \Lambda_{i\text{MIN}}$, then the value of $\vec{h}_i^*(t)$ is defined as the zero vector, since the function will be optimized in real time, the quadrotors will not move only when the fleet is very close to the desired formation.

The choice of the step time t_δ as well as the search intervals for the displacements $\vec{h}_i(t)$ of each agent are obtained empirically, and are considered as optimization constraints, such that:

$$\vec{h}_{i\text{MIN}}(t) < \vec{h}_i(t) < \vec{h}_{i\text{MAX}}(t) \tag{4.34}$$

The choice of $\vec{h}_i(t)$, thus, depends on the choice of t_δ.

Faulty agent case Assuming every quadrotor is capable of communicating if it becomes defective (actual fault detection algorithms are out of the scope of this work), it is considered that each agent can take into account the presence of a defective neighbor in $\Xi_i'(t)$, then let $K(t)$ be a diagonal matrix $n \times n$ of elements $\delta_i(t)$, where:

1. $\delta_i(t) = 1$ Agent i is free of faults.
2. $\delta_i(t) = 0$ Agent i loses its effectiveness totally and its output is stuck at zero.

The new adjacency matrix $A'(t)$ is defined as

$$A'(t) = A(t)\text{diag}(\delta_1(t), \delta_2(t), ..., \delta_n(t)). \tag{4.35}$$

A defective agent can be then modeled in the cost function as

$$\Lambda_i(t) = \rho\Big(\big|\big|\vec{p}_T - \big[\vec{p}_i(t) + \vec{h}_i(t)\big]\big|\big| - d_{ip}^d\Big) \tag{4.36}$$

$$+ \sum_{j=1}^{q} \delta_j(t) a_{ij}(t)\Big(\big|\big|\vec{p}_j(t) - \big[\vec{p}_i(t) + \vec{h}_i(t)\big]\big|\big| - d_{ij}^d\Big). \tag{4.37}$$

Note the zero value of $\delta_j(t)$ makes it possible to cancel $a_{ij}(t)$ and thus eliminates the influence of the defective agent j on agent i.

4.2.3 Emulated tests

To demonstrate the effectiveness of the proposed method, a comparison with other common techniques was performed. The selected approaches were Artificial Potential Fields (APF), based on a reduced version of the work presented in [96], and Model Predictive Control (MPC) [97], using a reduced adaptation of [98].

20 tests were performed in a real-time emulation environment [95] for each one of the three techniques, considering initial conditions generated at random for every simulation. The objective was to take three quadrotors from their random initial positions to a symmetrical formation around a fixed target.

Figures 4.18 and 4.19 depict the position of the vehicles during the simulations, note their initial positions are scattered throughout the plots. The desired distance the fleet must maintain towards the target is represented by a circle.

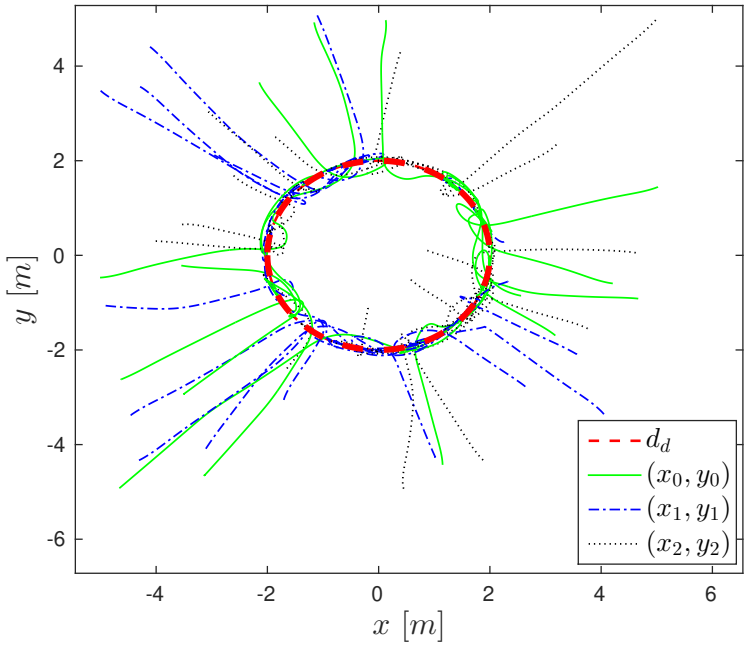
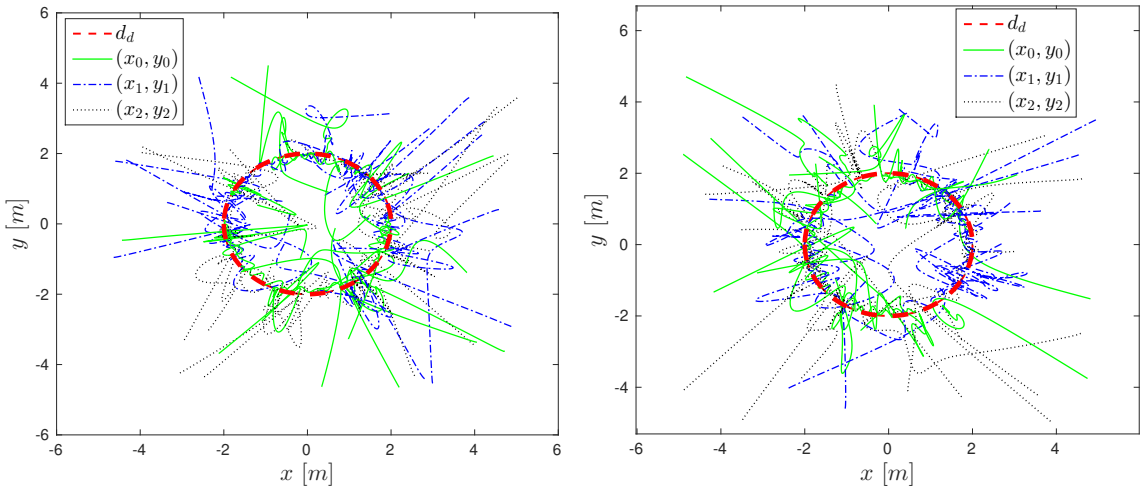


FIGURE 4.18: PSO: Horizontal position of the quadrotors in numeric simulations, presenting a smooth and robust convergence.

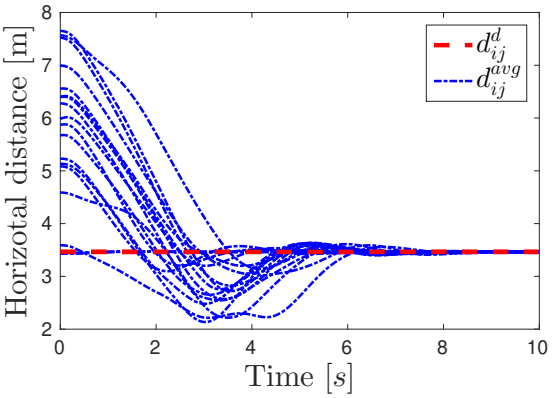


(A) APF: UAV feet formation simulations (B) MPC: Indicating strong oscillations.

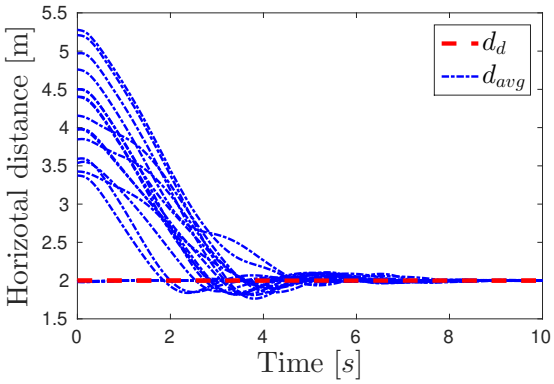
FIGURE 4.19: APF and MPC real-time simulations, the convergence of the fleet is not as smooth as in the PSO algorithm, presenting large oscillations before converging.

Note although the three approaches stabilize the vehicles at the desired distance, the convergence of the proposed PSO algorithm is smoother, presents less oscillations, and the computed trajectories respect a safer distance between the quadrotors and the target.

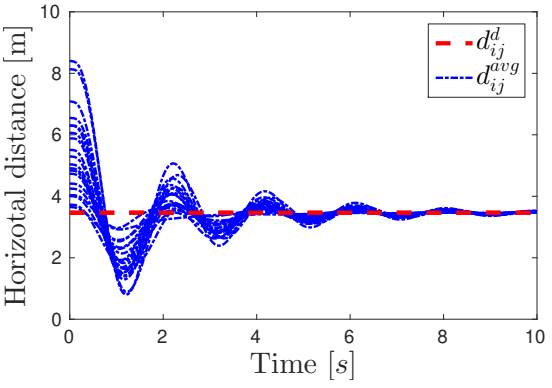
Figure 4.20 illustrates the behavior of the simulated fleet by displaying the average horizontal distance between the quadrotors and the averaged distance towards the target. In general, the PSO algorithm displays a smoother convergence towards the fleet formation with better robustness to unfavorable initial conditions than the other two approaches.



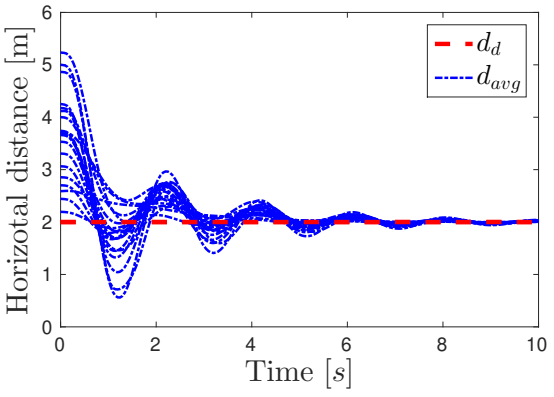
(A) PSO: Average distance between quadrotors for 20 simulations.



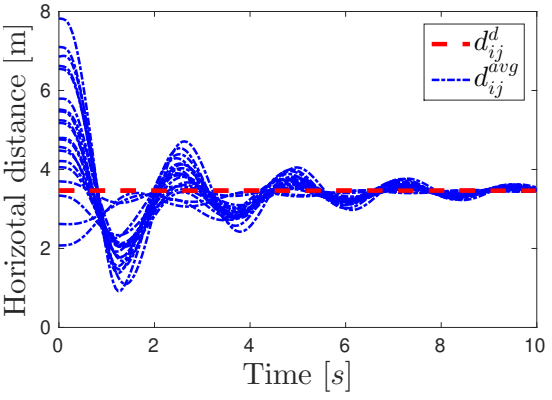
(B) PSO: Distance from UAVs to target, averaged for each simulation, $d_d = 2m$.



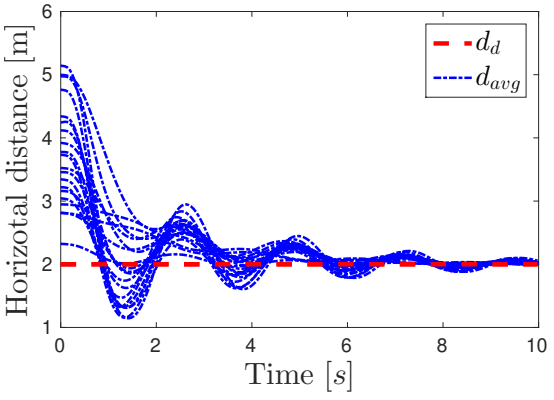
(C) APF: Average distance between UAVs, the vehicles come too close to each other.



(D) APF: UAV to target average distance, the fleet position oscillates while converging.



(E) MPC: Average distance between drones. Also sometimes coming close to a collision.



(F) MPC: Agent to target average distance, farther intial conditions trigger oscillations.

FIGURE 4.20: Comparison between 60 simulations, using three different approaches (PSO, APF, and MPC).

4.2.4 Experimental validation

The proposed algorithm was coded and tested in real-time experiments on a fleet of three quadrotors. All the drones were programmed with the same code. The position of the agents is estimated using an OptiTrack Motion Capture system, and broad-casted to all the UAVs.

Each quadrotor computes its own trajectory, such that an uniform fleet is formed around a given target while avoiding collisions amongst each other. A triangular formation is expected under this configuration.

Next, to illustrate the robustness of the presented approach, unknown disturbances were added, the PSO algorithm re-computes the required trajectory to recover from the perturbations. Lastly, the case in which a defective agent is introduced, and the overall fleet behavior is analyzed.

The parameters considered in for the PSO algorithm are depicted in Table 4.3.

$\rho = 5$	$\|\|\vec{h}_i\|\| < 0.4m$
$\sigma = 0.4$	$\delta t = 0.1s$
$c = 0.5m$	$d^d_{ij} = 1.732$
$\Lambda_{i\text{MIN}} = 0.1$	$d_d = 3.464$
$\|\|\vec{h}_{i\text{MAX}}\|\| = .02$	$\|\|\vec{h}_{i\text{MAX}}\|\| = 0.7$

Table 4.3: PSO considered parameters

The selected UAVs were 3 Parrot ARDrone2, this drone includes a 32bit ARM Cortex A8 processor, working at 1GHz with a 1Go DDR2 RAM at 200 MHz, these resources are quite limited, which is expected since its cost is low compared to other platforms.

Due to these limitations, the number of particles of $\Xi_i(t)$ was set to be 80. If this number is higher, the optimization algorithm becomes more precise, but the internal computer does not arrive to compute in time all the other processes required such as acquiring signals, communication, and motor controllers, the inverse would happen if less particles were considered.

The optimization constraints also consider a bounded region for generating random particles, defined by the values $\vec{h}_{i\text{MIN}}$ $\vec{h}_{i\text{MAX}}$, ensuring the particles are generated close to the current position of the UAV.

Fleet of three quadrotors with fixed target and no disturbances

In this first experiment, the desired configuration for the UAVs from the matrix ∂_d is fixed. The elements $d_{ij}(t)$ are defined such that the desired distance between UAVs

is homogeneous, thus arriving to a triangle configuration around the target point, as illustrated in Figure 4.21.

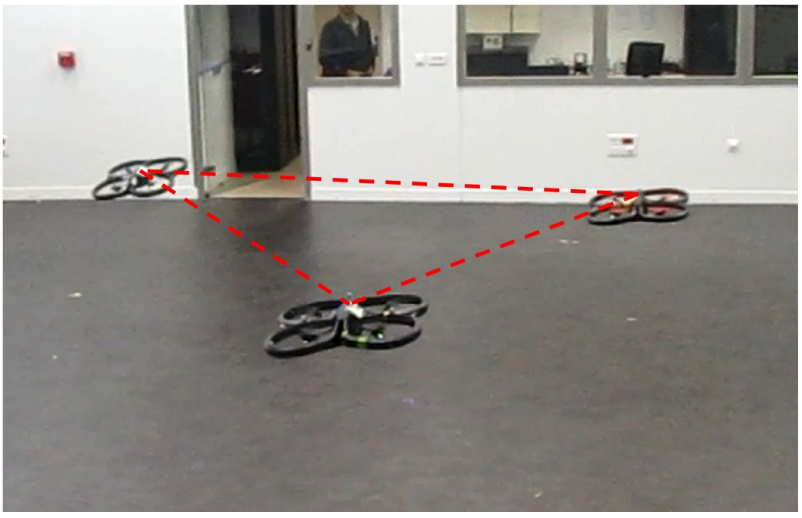


FIGURE 4.21: Three quadrotors in triangular formation.

Four punctual coordinates were considered as targets which location was communicated from a ground station, the targets were switched one to another with a fixed lapse. The optimization algorithm computes the desired position taking into account the constraints of the UAVs, since the position is calculated in each iteration, it forms a trajectory which makes the fleet arrive at the desired objectives.

Figure 4.22 represents the fleet translational behavior, the desired target points, the computed trajectory, and the real position of the UAVs in a 2D space. Note the triangular formation is broken when the target changes from one place to another due to the distance from the current location of the fleet, but it is recovered during the evolution of the PSO algorithm.

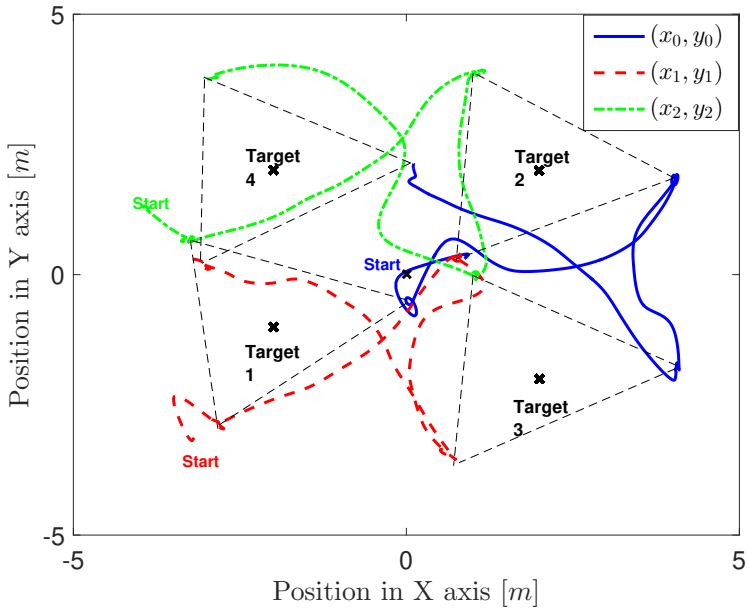


FIGURE 4.22: 2D targets, trajectory and drone positions

An important issue that is addressed in the fleet formation problem is the stabilization of the distance between agents. In this case, a uniform triangular formation is expected with a desired distance of $1.9m$ towards the target, using simple geometry, this implies that the desired distance between agents is $d_{\text{agents}} = 2(1.9m)\cos(\pi/6) = 3.3m$.

The computed references for the agents, ensure that the distance between every pair of agents converge to the desired value, this behavior is illustrated in Figure 4.23.

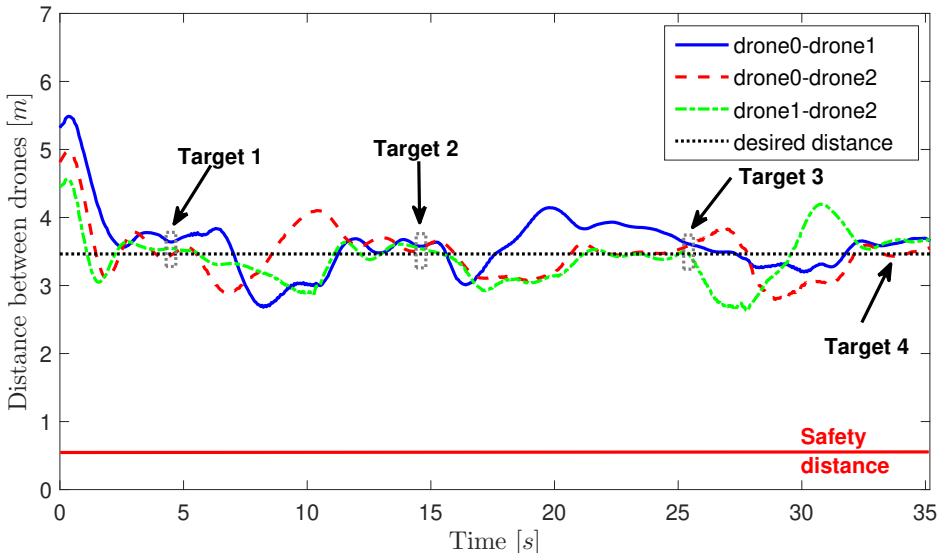


FIGURE 4.23: Desired distance between agents in real-time experiment

Figure 4.24 represents the desired distance between each agent and the target, note its smooth convergence to the desired value even when abrupt changes occur.

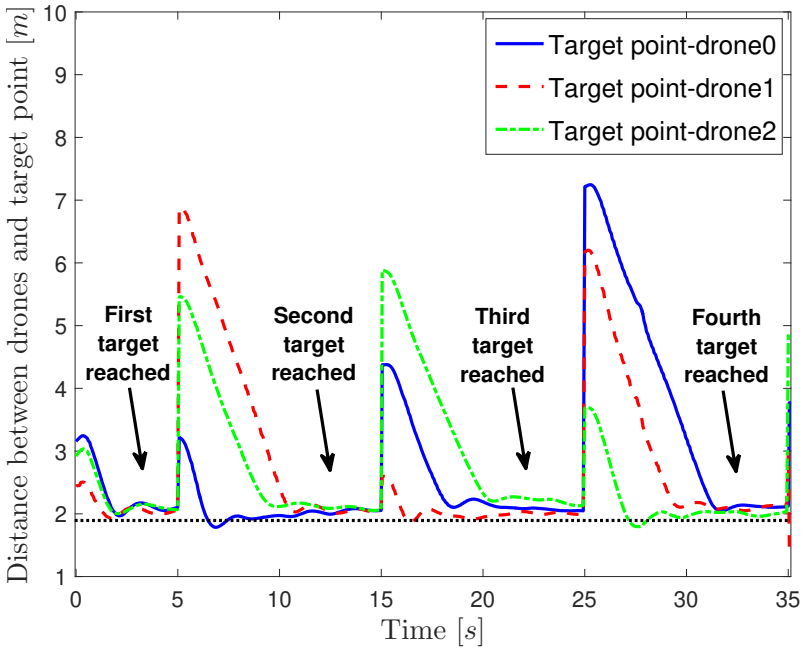
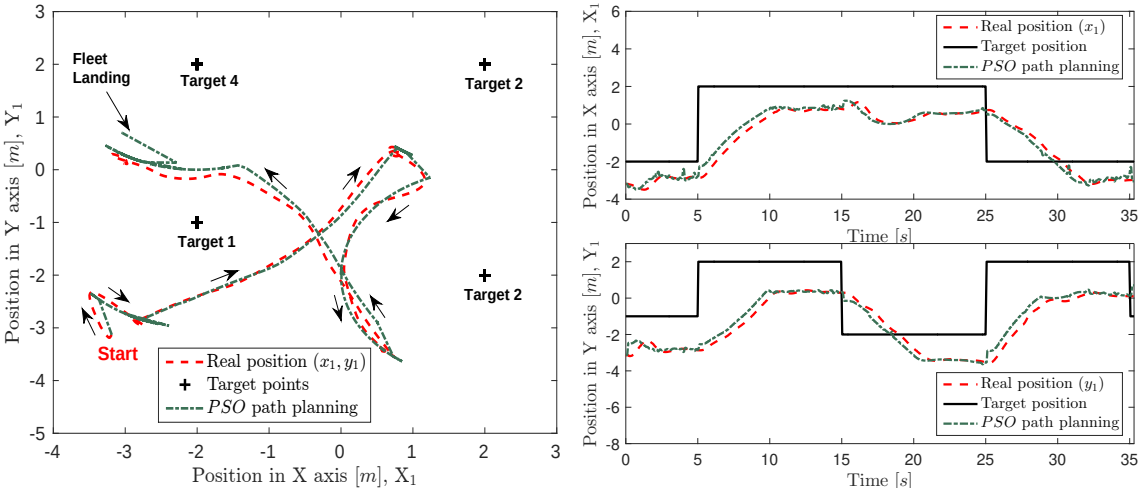


FIGURE 4.24: Desired distance between each and the target point.

To better illustrate the individual behavior of each UAV, Figure 4.25 represents the trajectory generation and tracking of the quadrotor 1, note the smooth convergence of the PSO trajectory even when the target position is changed abruptly.



(A) First quadrotor position, trajectory, and target in 2D.

(B) Target reference, and UAV trajectory and position over time

FIGURE 4.25: First quadrotor position response and target location in the first experiment.

Fleet of three quadrotors with fixed target and disturbances In the second experiment, the same flight configuration was considered, but unknown disturbances were added. Here, one of our team members pulled one of the agents as it flew towards the target (see Figure 4.26), the PSO algorithm compensated the disturbance and corrected the agent path.

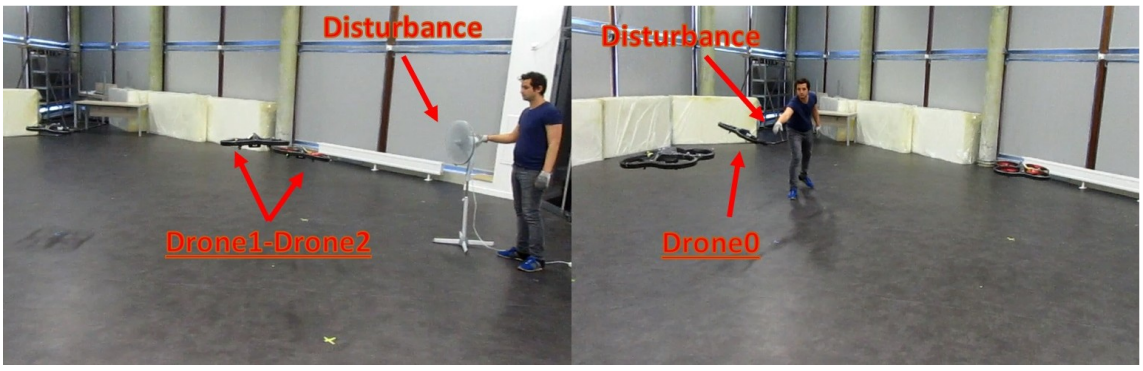
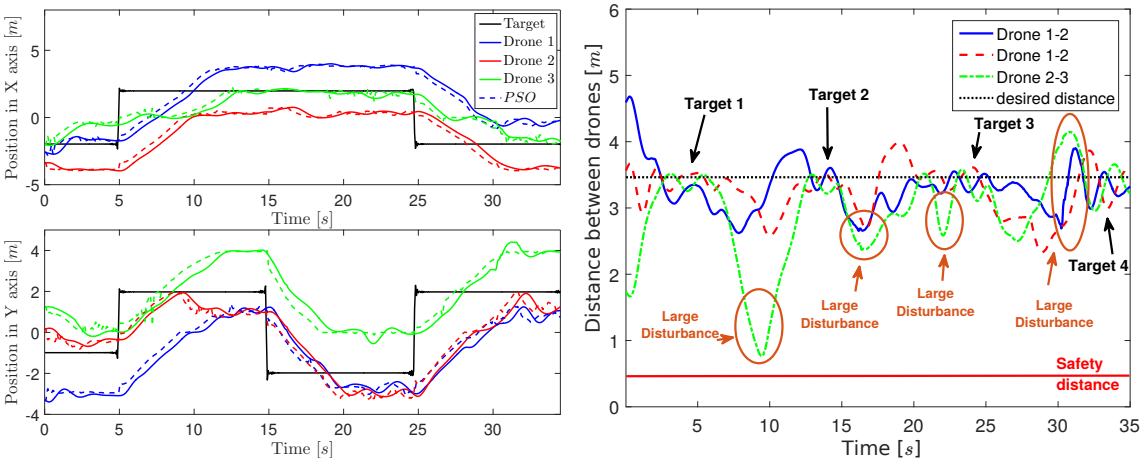


FIGURE 4.26: Disturbances induced to the fleet during experiments.

Figure 4.27A represents the target location, as well as the generated trajectory and position for each drone. Note the fleet formation is maintained while reaching each target, even in the presence of perturbations.

The distances between agents are illustrated in Figure 4.27B, note that strong disturbances are presented at some instants, but the PSO algorithm recovers the position of the affected UAV in a smooth manner, returning to the desired values.



(A) Target reference, trajectory, and position for all UAVs

(B) Desired distance between agents in the presence of strong perturbations.

FIGURE 4.27: Fleet formation in the presence of large disturbances, vehicle positions and distance between agents.

Fleet of three quadrotors with mobile target and disturbances

In this experiment, a different scenario was introduced. A mobile target was considered to test the response of the fleet. The movement of the target was inputted using a joystick connected to a ground station and moved manually. In this case the PSO algorithm computes the trajectory for each UAV such that the overall fleet tracks the moving reference.

Additionally, a member of our team induced strong disturbances to the fleet in order to test its robustness under this configuration, as depicted in Figure 4.28.

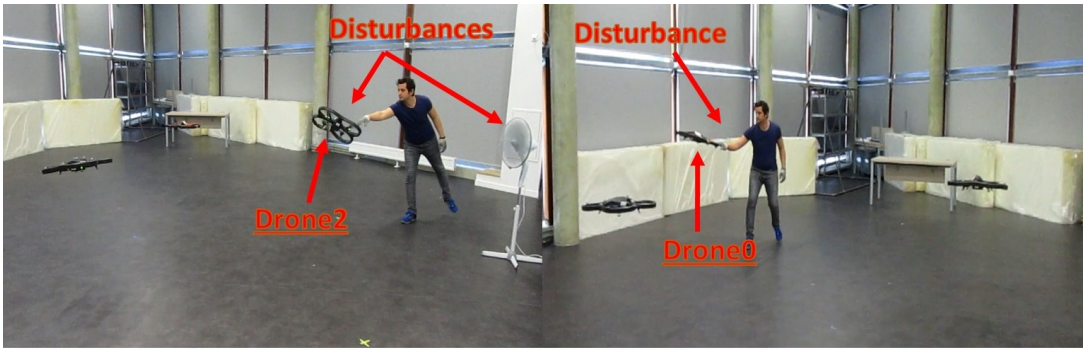


FIGURE 4.28: A member of our team inducing disturbances on the fleet.

Figure 4.29 illustrates the position of the target, and the trajectory generation and tracking for all the drones, note that the behavior of the fleet is consistent with the position changes of the target.

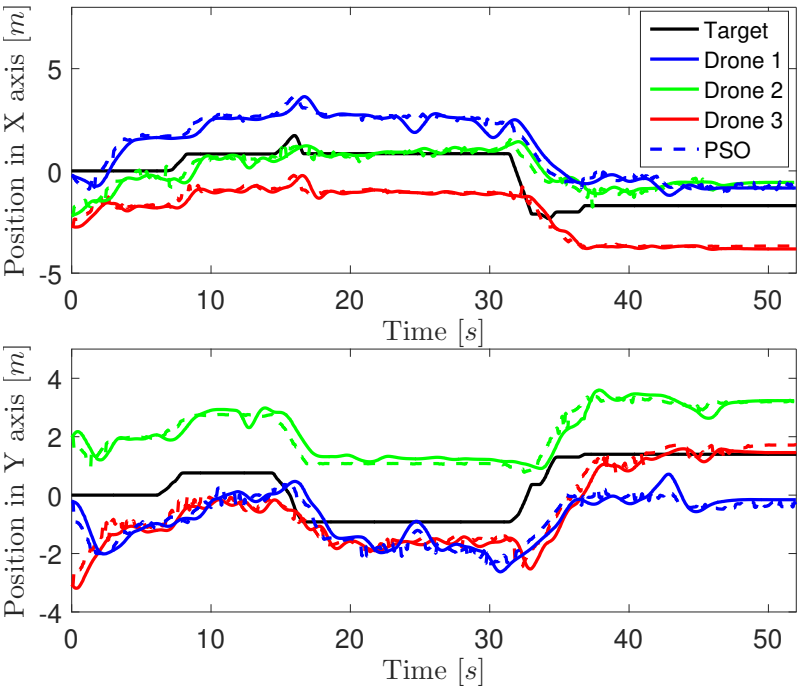
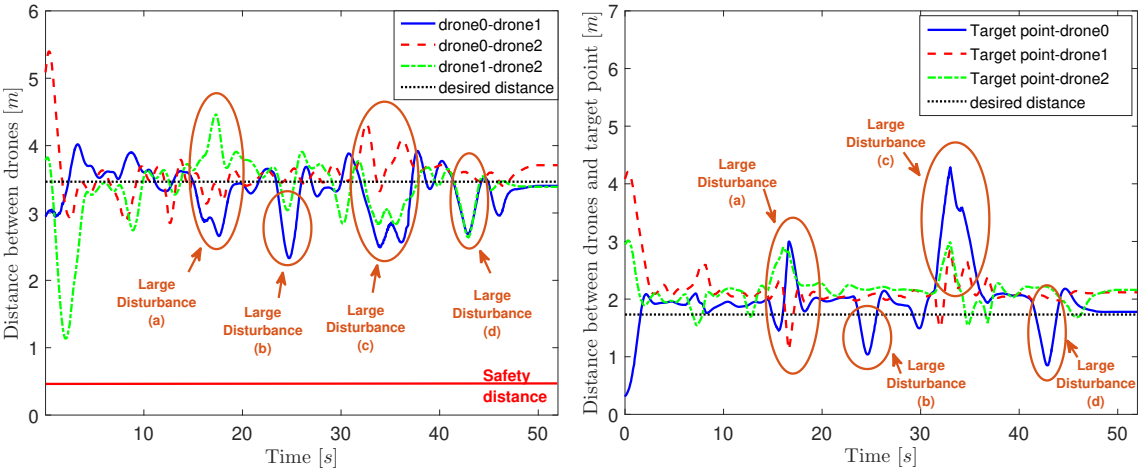


FIGURE 4.29: Target reference, trajectory, and position for all UAVs

Figure 4.30A illustrates the distance between agents, while Figure 4.30B represents the distance between each agent and the target, letters (a,b,c,d) correspond to each disturbance. Note the convergence towards the desired values, even when disturbances are present.



(A) Desired distance between agents, with the presence of a moving target and disturbances

(B) Distance between agents and recovery from perturbations with a moving target

FIGURE 4.30: Fleet formation with a moving target in the presence of large disturbances, vehicle positions and distance between agents.

Fleet of three quadrotors with fixed target and defective agent

In this last experiment, the same scenario as in section 4.2.2 was considered, but in this case, one agent was set to act as defective (Figure 4.31), and aborted its mission and landed while the fleet was moving to the desired reference.

FIGURE 4.31: Defective agent during real-time experiments

The optimization algorithm computes the optimized trajectory for the remaining agents, continuing the mission and arriving to the desired target. Figure 4.32 represents the fleet position tracking towards the desired references, note the remaining agents arrive to the target while the defective one remains in the ground.

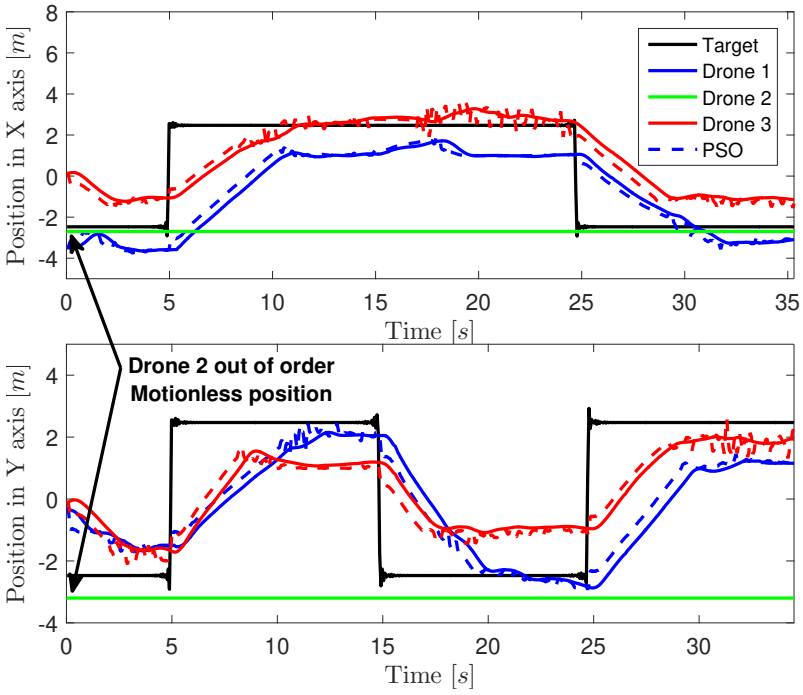


FIGURE 4.32: Target reference, trajectory, and UAV position for all UAVs

The desired distances between each agent and the target point are represented in figure 4.33, note the convergence to the desired value for the quadrotors that are still flying. All the performed experiments can be watched at the following link: `https://youtu.be/VD3vbGZNhqM`

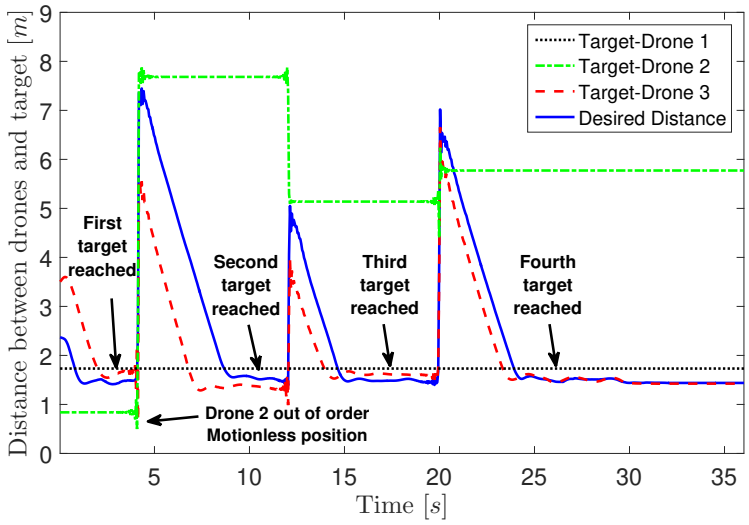


FIGURE 4.33: Desired distance between UAVs and target with a defective agent.

4.3 Quadrotor aggressive deployment

Quickly deploying a rotorcraft can sometimes be difficult and time-consuming, often requiring clear and flat spaces to initialize the vehicle. For instance, in firefighting, security, or rescue scenarios, objects around the deployment area (like tall grass or debris) could hinder the initial steps of any mission, other activities like outdoor sports such as hiking, climbing, or biking could lack of an appropriate surface to place the vehicle.

An intuitive and fast deployment could be to just hand-launch the vehicle (see Figure 4.34). Some expert hobbyists do it manually with advanced piloting skills, nevertheless they usually do so with a close to zero attitude and/or with the motors turned on, however spinning propellers could be hazardous for the launcher, and the vehicle itself, for safety reasons, quadrotor hand-launching should be performed with its motors switched off. Launching a drone in this manner and autonomously recovering while in the air is a real problem that has not been totally solved.

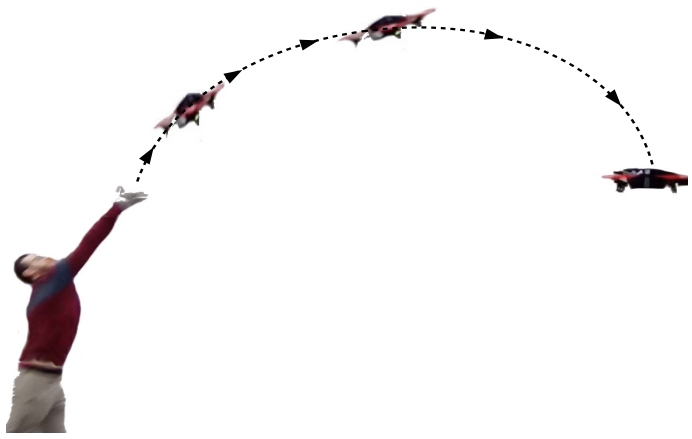


FIGURE 4.34: Quadrotor aggressive deployment

A solution for this problem is here proposed by introducing recovery and control algorithms formulated on quaternion-based and sliding mode methodologies. This technique provides on one hand, an analysis of the launching conditions, proposing an aggressive attitude trajectory to autonomously recover the quadrotor rotational stability, and on the other hand, a nonlinear controller capable of quickly tracking this trajectory. The recovery and launching detection algorithms are computed by only employing inertial sensors allowing the system to be used either in indoors, outdoors, or GPS-denied environments. Continuous functions are defined instead of state machines to activate the motors and switch between recovery and mission-specific attitude references, giving a more precise description of the system and simplifying its implementation. As of the controller, a spherical sliding mode technique previously introduced in Section 3.6 was used to aggressively track the recovery rotation references.

4.3.1 Attitude Trajectory Formulation

An aggressive deployment implies the vehicle could be launched at any possible initial attitude, thus it must be recovered from such conditions to a hovering state before handing over the baton to the user. To address this problem, a combination of two attitude references is proposed.

Recovery Trajectory

Firstly, in order to stop the vehicle from its free-fall condition, the quadrotor thrust vector must compensate the gravitational acceleration as fast as possible.

The representation of the vertical axis of the inertial frame $\mathcal{I}$ with respect to the body's coordinate system $\mathcal{B}$ can be defined by a unit vector $\vec{n}_z^{\mathcal{B}} \in \mathbb{R}^3$ as

$$\vec{n}_z^{\mathcal{B}} = \boldsymbol{q}^* \otimes \vec{n}_z \otimes \boldsymbol{q}, \tag{4.38}$$

where $\vec{n}_z := [0, 0, 1]^T$.

Introducing the definitions of *dot* and *cross* products between two vectors, and widely known trigonometric expressions into (2.15) the shortest rotation between $\vec{n}_z^{\mathcal{B}}$ and the vertical axis of the quadrotor can be computed as

$$\boldsymbol{q}_b := \left(\sqrt{\frac{1 + \vec{n}_z^{\mathcal{B}} \cdot \vec{n}_z}{2}} + \frac{\vec{n}_z^{\mathcal{B}} \times \vec{n}_z}{||\vec{n}_z^{\mathcal{B}} \times \vec{n}_z||} \sqrt{\frac{1 - \vec{n}_z^{\mathcal{B}} \cdot \vec{n}_z}{2}} \right), \tag{4.39}$$

since $\boldsymbol{q}_b$ is computed using vectors expressed in $\mathcal{B}$, the recovery trajectory quaternion is completed by adding the vehicle's rotation with respect to $\mathcal{I}$ as

$$\boldsymbol{q}_r = \boldsymbol{q} \otimes \boldsymbol{q}_b^*, \tag{4.40}$$

see Figure 4.35.

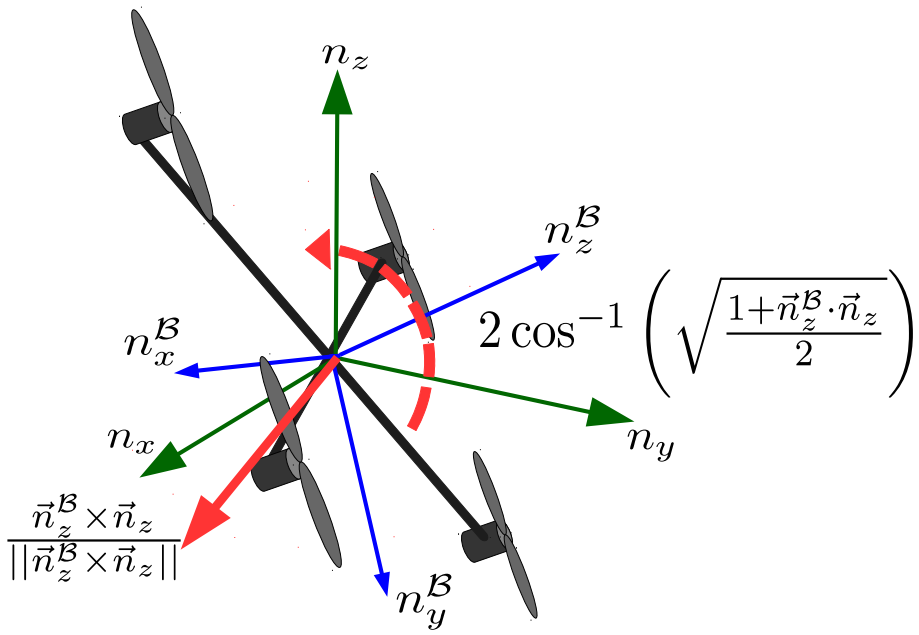


FIGURE 4.35: Shortest quaternion recovery rotation

Mission Specific Reference

Once the vehicle is recovered, the desired orientation might follow any reference according to the mission requirements, some tasks like position holding or translational trajectory tracking, would define it in terms of a high-layer position control, other applications like emergency response or outdoors video capture could require a human operator, whatever the case, the mission attitude reference is independent from the aggressive hand-launching deployment challenge. For repeatability and ease of performing experiments, in this work a human is considered to pilot the quadrotor in semi-autonomous mode by defining a desired orientation $\boldsymbol{q}_{usr}$ provided by a joystick.

Due to quaternion dual covering of rotations, as seen from (2.19), the attitude at the moment the vehicle is recovered might have either positive or negative representation, one of them with an additional 2π rotation. This phenomenon might cause undesired behavior on the vehicle. To tackle this problem, the user input is complemented by changing the sign of the joystick reference such that the additional rotation is avoided by defining

$$\boldsymbol{q}_j \coloneqq \operatorname{sign}(\pi - 2\ln(\boldsymbol{q}_{je}))\boldsymbol{q}_{usr}, \tag{4.41}$$

where $\boldsymbol{q}_{je} = \boldsymbol{q}_{usr}^* \otimes \boldsymbol{q}$ symbolizes the rotation difference between the vehicle attitude and the user input.

Attitude Reference Combination

Once the recovery and user attitude references are defined, they are combined by the following function

$$\boldsymbol{q}_d := \boldsymbol{q}_r \otimes \boldsymbol{q}_{\chi r} \otimes \boldsymbol{q}_j \otimes \boldsymbol{q}_{\chi j}, \tag{4.42}$$

where $\boldsymbol{q}_{\chi r}$ and $\boldsymbol{q}_{\chi j}$ are quaternions that enable and disable $\boldsymbol{q}_r$ and $\boldsymbol{q}_j$ as needed, and are derived from (2.15) and (2.16) as

$$\boldsymbol{q}_{\chi r} := e^{-\gamma_R \ln(\boldsymbol{q}_r)}, \tag{4.43}$$

and

$$\boldsymbol{q}_{\chi j} := e^{-(1-\gamma_R)\ln(\boldsymbol{q}_j)} \tag{4.44}$$

where $0 < \gamma_R < 1$ is a scalar number which changes its value as needed to activate or deactivate each reference.

Remark 4.1. If $\gamma_R \to 0$, then $\boldsymbol{q}_{\chi r} \to 1 + [0\ 0\ 0]^T$, while $\boldsymbol{q}_{\chi j} \to \boldsymbol{q}_j^*$, and in consequence

$$\boldsymbol{q}_d \to \boldsymbol{q}_r, \tag{4.45}$$

inversely, if $\gamma_R \to 1$, then $\boldsymbol{q}_{\chi j} \to 1 + [0\ 0\ 0]^T$, while $\boldsymbol{q}_{\chi r} \to \boldsymbol{q}_r^*$, thus

$$\boldsymbol{q}_d \to \boldsymbol{q}_j. \tag{4.46}$$

γ_R will be defined as a function of the vehicle's acceleration in the following section.

Continuous Switching Strategy

When the quadrotor is launched aggressively with its motors turned off, gravity forces it to fall in a parabolic path, behaving as a body under free-fall conditions.

Under this scenario, the accelerometers display near-zero values, which do not provide sufficient information to estimate the vehicle attitude, nevertheless, their signals can be used to detect the moment in which the quadrotor is deployed as Figure 4.36 illustrates.

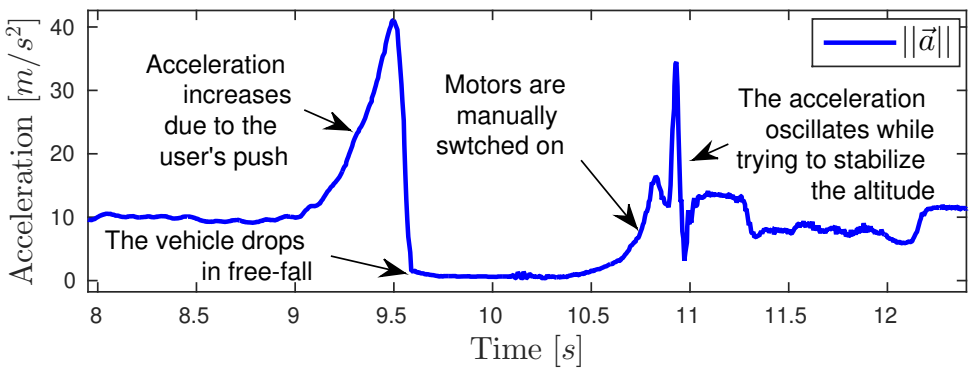


FIGURE 4.36: Acceleration profile example during manual recovery

Control Terms Switching

A continuous switching technique is proposed to activate and deactivate the attitude terms from (3.112) when the accelerometer measurements drop by defining

$$\gamma_a := \frac{1}{2}\left(\tanh\left(\beta_a(||\vec{a}|| - \alpha_a)\right) + 1\right), \tag{4.47}$$

where $\vec{a}$ represents the acceleration vector, $\beta_a > 0$ is a tunning gain, and α_a denotes the acceleration threshold from which free-fall condition is considered. Note that

$$\begin{aligned} \gamma_a &\to 1 \text{ if } ||\vec{a}|| > \alpha_a\,, \text{ and} \\ \gamma_a &\to 0 \text{ if } ||\vec{a}|| < \alpha_a\,, \end{aligned} \tag{4.48}$$

such that the attitude controller is finally rewritten as

$$\vec{\tau} = -K_{d1}\vec{\Omega} - \Gamma\left(\gamma_a K_p 2\ln\left(\boldsymbol{q}^*_{\chi j} \otimes \boldsymbol{q}^*_j \otimes \boldsymbol{q}^*_{\chi r} \otimes \boldsymbol{q}^*_r \otimes \boldsymbol{q}\right) + K_{d2}\vec{\Omega}\right) + \vec{\Omega} \times J\vec{\Omega}, \tag{4.49}$$

notice when the vehicle is in free-fall, the attitude values are ignored and only the angular speed is regulated, otherwise, the controller will track the desired rotation defined by the recovery strategy.

Attitude Reference switching

Following a similar reasoning, two scalar variables dependent on the vehicle attitude and angular velocity are proposed as

$$\gamma_\vartheta := \frac{1}{2}\left(\tanh\left(\beta_\vartheta(||2\ln(\boldsymbol{q}_{re})|| - \alpha_\vartheta)\right) + 1\right), \tag{4.50}$$

and

$$\gamma_\Omega := \frac{1}{2}\left(\tanh\left(\beta_\Omega(||\vec{\Omega}|| - \alpha_\Omega)\right) + 1\right), \tag{4.51}$$

where $\boldsymbol{q}_{re} := \boldsymbol{q}^*_r \otimes \boldsymbol{q}$ represents the rotation difference between the quadrotor attitude and the recovery quaternion from (4.40), $\beta_\vartheta, \beta_\Omega > 0$ are tunning parameters, and $\alpha_\vartheta, \alpha_\Omega$ symbolize the angle error and angular velocity thresholds to consider the vehicle as stabilized in hover mode.

Then (4.50) and (4.51) are combined as

$$\gamma_R(t) := \tanh\left(\beta_R \int_{t_0}^{t} \gamma_\vartheta \gamma_\Omega dt\right), \quad \gamma_R(t_0) = 0, \tag{4.52}$$

where t_0 defines the time when the quadrotor starts waiting to be launched and $\beta_R > 0$ is a tunning parameter.

Since (4.49) will stabilize the vehicle's attitude to the recovery values, then $\exists\, t$ such that $\forall\, t > t_h$ when the quadrotor attitude follows $||2\ln(\boldsymbol{q}_{re})|| < \alpha_\vartheta$ and $||\vec{\Omega}|| < \alpha_\Omega$,

therefore, (4.52) behaves as

$$\begin{array}{lll} \gamma_R \approx 0 & \text{for} & t_0 < t < t_h\,, \quad \text{and} \\ \gamma_R \to 1 & \text{for} & t > t_h\,. \end{array} \tag{4.53}$$

Remark 4.2. (4.42), (4.52), and (4.53) imply that the attitude reference will be defined by the recovery trajectory before t_h, and by the user-defined input after t_h.

Motor Activation

Recalling (2.3), the relation between the force and speed of each propeller $i = 1, \ldots, 4$ can be computed as

$$\omega_i \approxeq \sqrt{f_i/k_i}. \tag{4.54}$$

An expression to automatically activate the motors when the quadrotor is being tossed is proposed as

$$\gamma_\mu(t) := \tanh\left(\zeta_\mu \int_{t_0}^{t} \left(\tanh\left(\beta_\mu(||\vec{a}|| - \alpha_\mu)\right) + 1\right) dt\right), \tag{4.55}$$

where $\zeta_\mu, \beta_\mu > 0$ are tunning parameters, and α_μ denotes the acceleration threshold from which aggressive tossing is considered. Therefore (4.54) can be rewritten as.

$$\omega_i \approxeq \gamma_\mu(t)\sqrt{f_i/k_i}. \tag{4.56}$$

Defining time t_l as when the quadrotor is accelerated for launching and $||\vec{a}|| > a_\mu$. Considering $\gamma_\mu(t_0) = 0$, then (4.55) follows

$$\begin{array}{lll} \gamma_\mu \approx 0 & \text{for} & t_0 < t < t_l\,, \quad \text{and} \\ \gamma_\mu \to 1 & \text{for} & t > t_l\,. \end{array} \tag{4.57}$$

Altitude recovery

Let z_{usr} represent the altitude reference given by the pilot. Since the launching conditions are initially unknown, the vehicle might be far from the desired altitude when the attitude is being recovered. To smoothen the vehicle behavior, an altitude trajectory is introduced as

$$z_d(t) := \begin{cases} z_0 - \int_{t_{z_0}}^{t} v_{zr} d\tau & \forall \quad t_{z_0} \leq t \leq t_{zf}\,, \\ z_{usr} & \text{otherwise}\,, \end{cases} \tag{4.58}$$

where $v_{zr} > 0$ is the desired speed of descent for the quadrotor, t_{z_0} represents the time when free fall conditions are met and is determined according to

$$\gamma_a(t) < 0.5 \;\text{ for }\; t \in [\,t_0\,,\, t_{z_0}\,] \text{ and } \gamma_a(t) \geq 0.5 \text{ for } t > t_{z_0}, \tag{4.59}$$

t_{z_f} defines the time when the altitude reference reaches z_{usr} such that $z_d(t) > z_{usr}$ for all $t_{z_0} < t < t_{z_f}$, and z_0 denotes the initial altitude reference, determined as

$$z_0 = \max\{z(t_{z_0}), z_{min}, z_{usr}\}, \tag{4.60}$$

where z_{min} and $z(t_{z_0})$ symbolize the minimum safety altitude, and the measured height at t_{z_0} respectively.

4.3.2 Experimental validation

Multiple tests have been performed using the proposed strategy, in indoor and outdoor environments. Two experiments illustrate such tests in this thesis, however, more tests can be viewed on-line at: `https://youtu.be/_F_XNmzxPDg`

For these experiments a Parrot ARDrone2 quadrotor was used, the algorithm was coded using the FL-AIR framework [95], developed at Heudiasyc Laboratory.

For this application, the attitude was considered to be controlled with an sliding mode controller, while the position was regulated by a state feedback only in the vertical axis. Let $z \in \mathbb{R}$ symbolize the quadrotor altitude with respect to $\mathcal{I}$, then its vertical dynamics are given by

$$\frac{d}{dt}\begin{bmatrix} z \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ F_{th}\cos(\vartheta_{xy})/m + \vec{g} \end{bmatrix}, \tag{4.61}$$

where $F_{th} = \sum_{i=1}^{4} f_i$, $F_{th} \in \mathbb{R}$ denotes the total thrust force produced by the propellers, the gravitational acceleration is symbolized as $\vec{g} := [0 \;\; 0 \;\; -g]^T$ with $g \approx 9.81[m/s^2]$, m defines the vehicle mass, ϑ_{xy} represents the angle between the vertical axes of $\mathcal{I}$ and $\mathcal{B}$, and its cosine is computed as

$$\cos(\vartheta_{xy}) = \left(\boldsymbol{q} \otimes [0 \;\; 0 \;\; 1]^T \otimes \boldsymbol{q}^*\right) \cdot [0 \;\; 0 \;\; 1]^T. \tag{4.62}$$

Defining the net thrust force as

$$F_{th} := \begin{cases} \dfrac{-k_{pz}(z - z_d) - k_{dz}\dot{z} + mg}{\cos(\vartheta_{xy})} & \text{if } \cos(\vartheta_{xy}) > 0, \\ F_{\min} & \text{otherwise} \end{cases} \tag{4.63}$$

where $k_{pz}, k_{dz} \in \mathbb{R}^+$ are positive gains, and $F_{\min} > 0$ defines the minimum thrust. Note from (4.62) and (4.63) that if the vehicle inclination ϑ_{xy} is equal or greater than 90°, the thrust force will be kept at a minimal value to reduce the risk of accelerating downwards.

The attitude controller parameters manually adjusted and the final values are introduced in the following table:

$\alpha_a = g/2$	$\beta_a = 10$	$\alpha_\vartheta = \pi/4$
$\beta_\vartheta = 10$	$\alpha_\Omega = \pi/4$	$\beta_\Omega = 10$
$\alpha_\mu = 2g$	$\beta_\mu = 10$	$\zeta_\mu = 10$
$\beta_R = 10$	$k_{pz} = 0.6$	$k_{dz} = 0.25$
$K_p = \mathrm{diag}\big([3.5 \quad 3.5 \quad 2]\big)$	$K_{d1} = 0.065\, I_{3\times3}$	$K_{d2} = 0.5\, K_{d1}$

TABLE 4.4: Controller parameters for real world tests.

Multiple tests have been performed using the proposed strategy on a Parrot ARDrone2 quadrotor, the algorithm was coded using the FL-AIR framework [95]. In some experiments, the vehicle was launched with the propellers pointing downwards, others while rotating over its three x (roll), y (pitch), and z (yaw) axes.

In order to maintain a reasonable document length, only the most aggressive scenario is illustrated in this manuscript, however it is recommended to watch more tests on-line at the following video link: https://youtu.be/b52e7K9BHYs

In Figure 4.37 the launching detection timing by the switching parameters from (4.47) and (4.55) is illustrated.

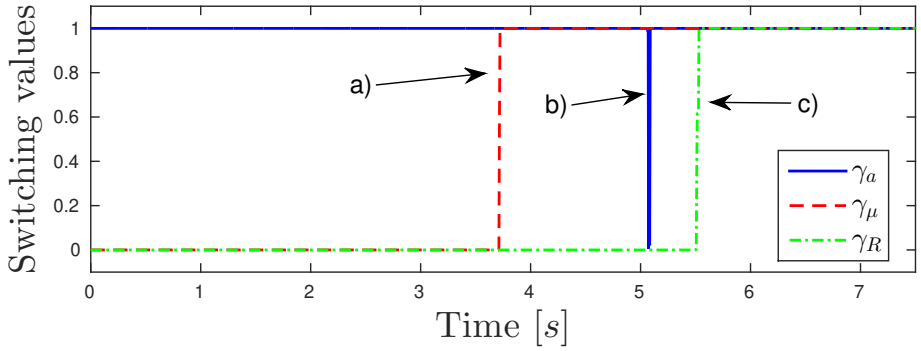


FIGURE 4.37: Detection terms performance, a) Tossing detected, starting motors. b) Free fall detected. c) Recovery achieved.

The quadrotor rotational response is represented in Figures 4.38 to 4.41, the vehicle attitude $\boldsymbol{q}$ was computed by the onboard Inertial Measurement Unit (IMU).

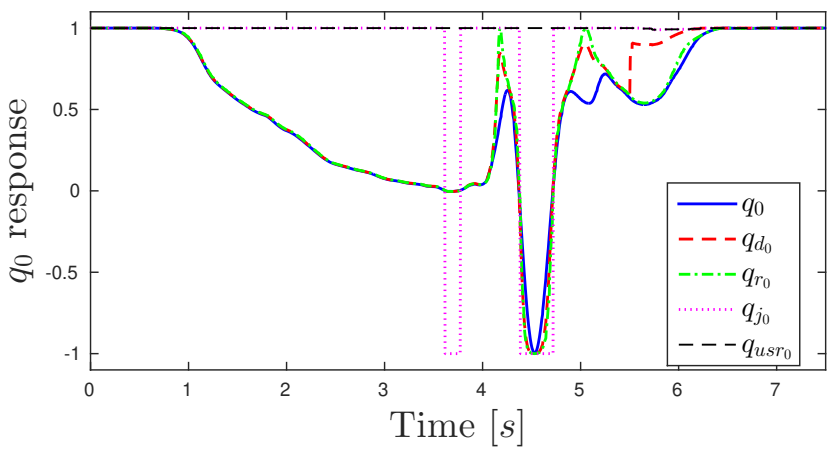


FIGURE 4.38: Quaternion q_0 component response, each change of sign indicates a 360^0 rotation during deployment.

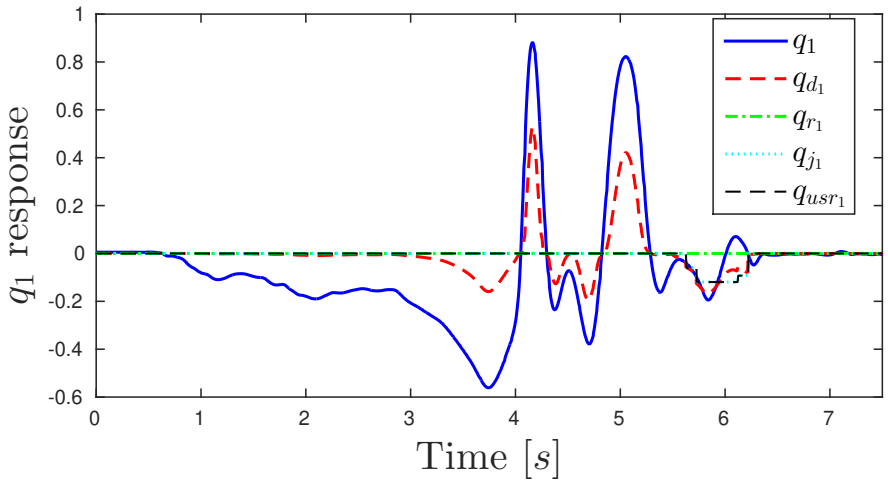


FIGURE 4.39: Quaternion q_1 component (over the x axis), the vehicle is first inclined, then rotates aggressively during the recovery stage.

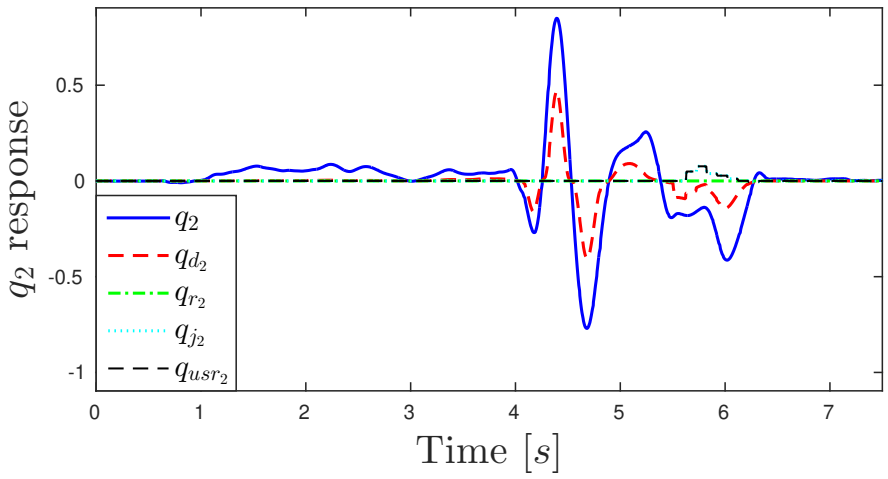


FIGURE 4.40: Quaternion q_2 component (response over the y axis), note how the combined rotation $\boldsymbol{q}_d$ pulls the vehicle attitude towards zero.

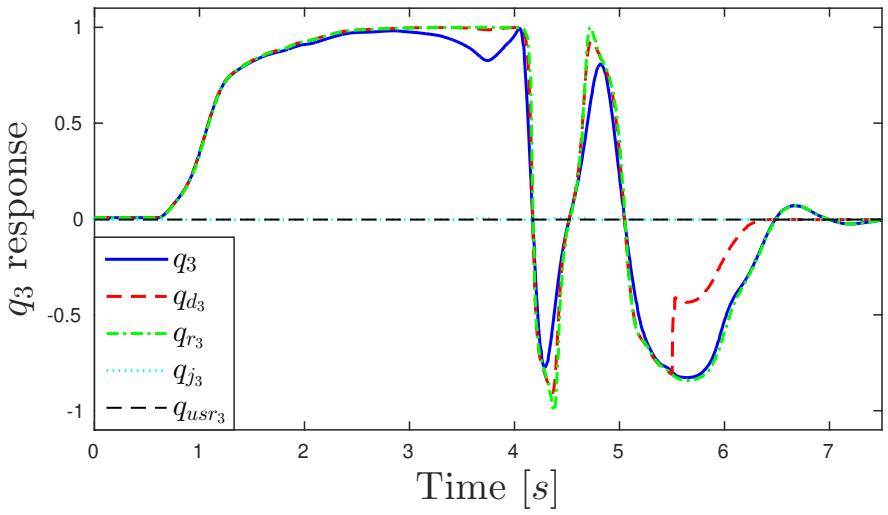


FIGURE 4.41: Quaternion q_3 component (z axis), signaling the vehicle also spins three times before stabilizing successfully.

For illustration purposes, Figures 4.42 to 4.44 depict the equivalent Euler angles computed from the previous quaternions during this experiment, due to the nature of this type of attitude representation, some undesired effects are revealed such as discontinuities and sudden changes in the signals.

Notice how the magnitude of the simultaneous pitch and roll rotations, along with their oscillations indicate the vehicle is aggressively rotated such that it passes through vertical, horizontal, and upside-down poses, but at the end, the proposed algorithm successfully recovers the quadrotor.

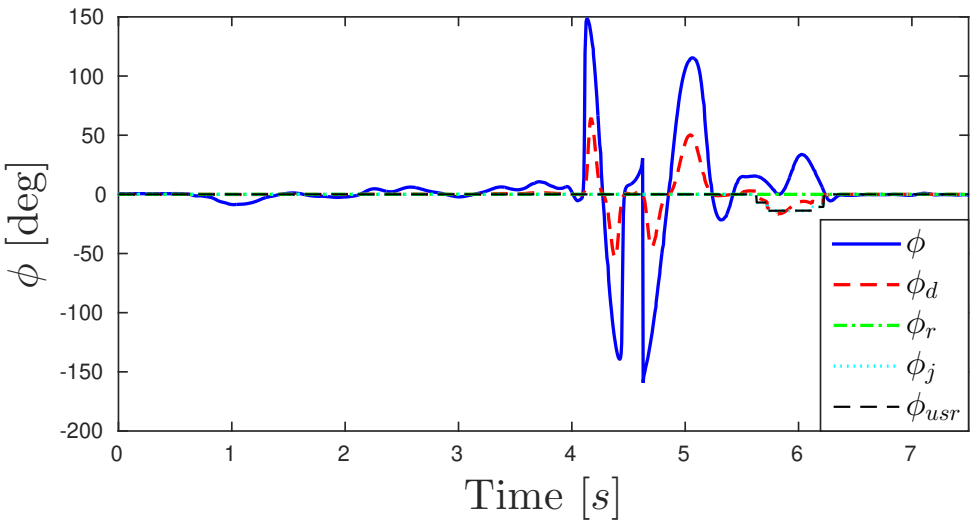


FIGURE 4.42: Equivalent roll angle response (rotation over the x axis), reaching almost 150° in multiple occasions.

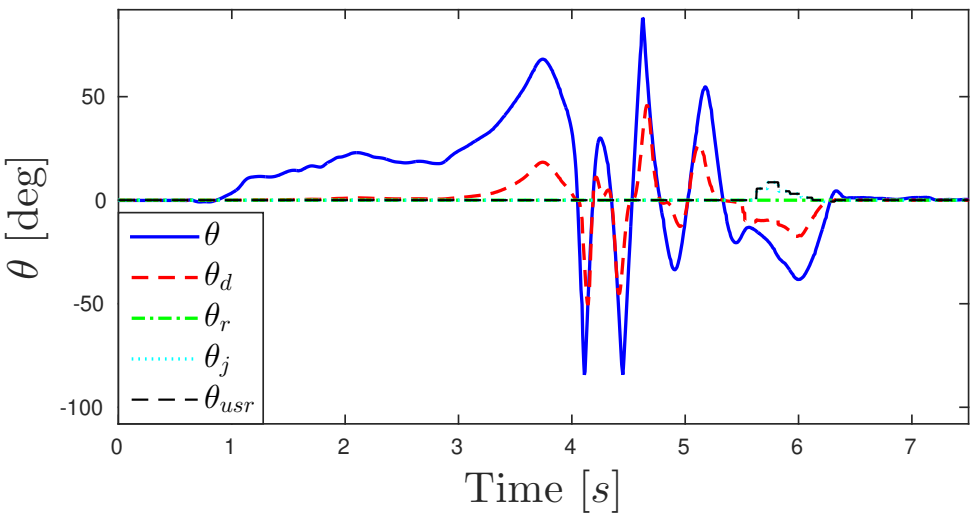


FIGURE 4.43: Equivalent pitch angle response (rotation over the y axis), approaching vertical poses of 90° several times.

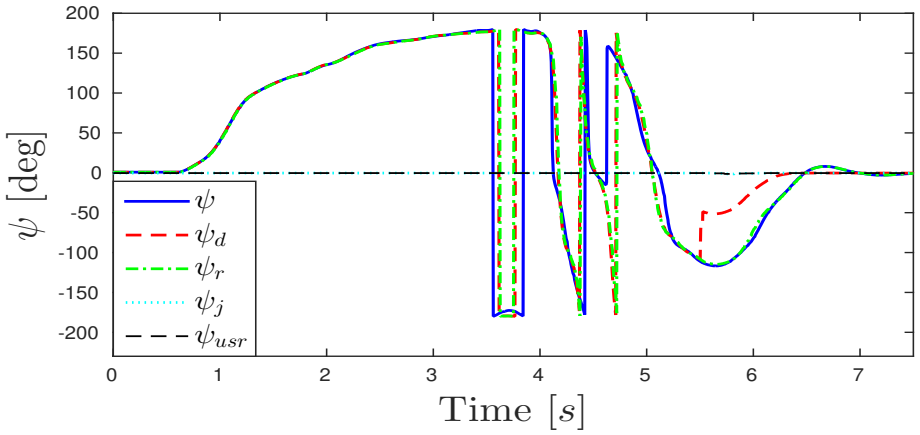


FIGURE 4.44: Equivalent yaw angle (rotation over the z axis), revealing multiple 360° turns before stabilizing.

4.4 Quadrotor navigation conclusions

Quaternion-based controllers provide advantageous properties for navigating aerial vehicles. The aim of this chapter is to introduce navigation techniques that employ quaternion-based algorithms for performing robust autonomous and semiautonomous navigation.

Firstly, a semi-autonomous scheme for safely piloting a quadrotor using arm gestures was proposed. The scheme is composed by a gesture-based orientation reference and a EMG triggered gesture sequences for performing desired movements in the drone. The proposed algorithms were validated in flight tests where the safety measures and customizable sensitivity effectively made the interface easier to handle for inexperienced users by reducing the learning curve and overall achieving a more intuitive command scheme.

Then, a distributed path planning approach for controlling a fleet of autonomous vehicles was proposed. The objective has been formulated as an optimization problem based on a Particle Swarm Optimization algorithm. The proposed control decouples the translational movement from the rotational dynamics for computing optimal trajectories, where the stability of each vehicle is ensured locally. Experimental results on a fleet of real quadrotors have shown that the proposed method is effective for target tracking and collisions avoidance, even considering scenarios where an agent of the fleet is lost.

An aggressive deployment strategy for a quadrotor was finally presented in which the vehicle is launched with its motors turned off. In order to enable the vehicle in-flight, a continuous recovery trajectory was proposed using unit quaternions, which is the tracked by an attitude controller, based on a chattering-free spherical sliding mode algorithm.

Multiple real-world experiments were executed for all of the proposed schemes, validating their performances, and setting a foundation for autonomous navigation strategies between aerial and ground vehicles.

Chapter 5

Autonomous tracking of dynamic targets

Aerial and ground autonomous vehicles provide both advantages and disadvantages that could help or difficult certain tasks. For instance, UAVs are capable of quickly reaching or surveying large areas, as well as providing higher points of view, however, they offer limited payload carrying capacities as well as relatively short times of autonomy.

In contrast, Unmanned Ground Vehicles (UGVs) can carry larger batteries and payloads, which might extend the length of some missions, and can provide advantages thanks to additional equipment that can be loaded, nevertheless, they have limited motion capabilities and points of view compared to UAVs.

Heterogeneous systems can use the advantages of each kind of vehicles to counteract their weaknesses. In this chapter, two navigation techniques for UAV-UGV heterogeneous systems are presented, firstly, a trajectory generation algorithm which includes takeoff, circular tracking, and landing stages for a quadrotor following a ground vehicle is proposed based on a set of dynamic equations. Then, a trajectory generation algorithm is introduced where two quadrotors cooperate to autonomously track a moving UGV by describing circles around it while respecting a desired configuration and avoiding collisions.

5.1 Circular trajectory for autonomous tracking

As a first cooperative navigation scheme, consider a scenario where a large area needs to be inspected by a UGV. In order to enhance the visible area for the UGV, a quadrotor is deployed such that it describes circles of a certain radius around the UGV's center, such that its privileged point of view could be used other members of the team (human operators and/or the UGV) in their operations.

To achieve this objective, a trajectory generation algorithm has been introduced, which is computed by solving a set of differential equations. This trajectory includes autonomous navigation stages such as take-off, circle describing, and landing.

5.1.1 Trajectory description

Consider the UGV as a static agent in a given reference point. A trajectory is proposed such that the quad-rotor follows autonomously a maximum of three stages of movement, where two possible scenarios can be achieved by considering this trajectory.

For the first scenario the drone would follow two stages of movement (refer to Figure 5.1), this mean only lifting and landing stages are accomplished.

- The quad-rotor will start lifting to a desired altitude h and converging to a circle with a defined turning radius r. This will be considered to happen in a time interval given by $0 \leq t < T_{F_1}$.
- The landing of the quad-rotor back to the UGV will start in a time defined as T_{F_1}, starting at this point an asymptotic descent and landing.

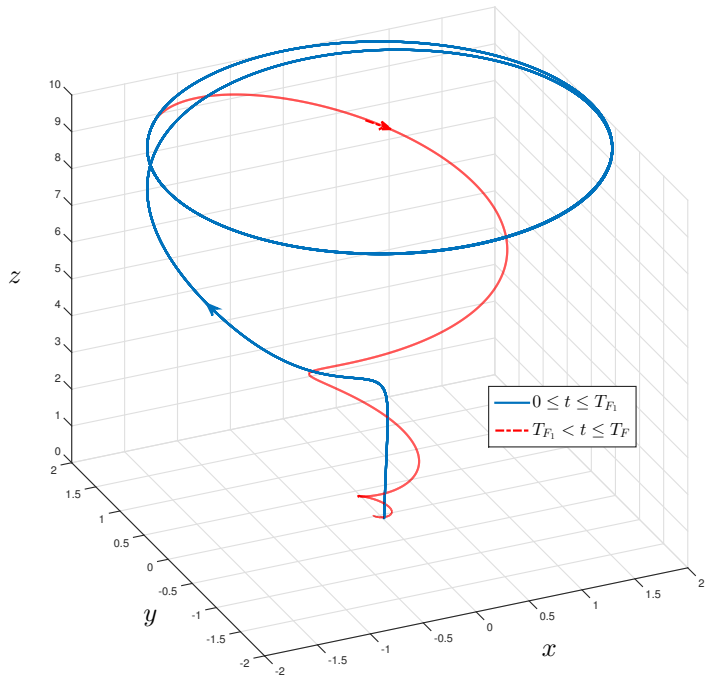


FIGURE 5.1: Two stage case: circular trajectory and landing.

The second scenario is when lifting, positioning and vertical landing stages are considered, see Figure 5.2.

- The quad-rotor will start lifting and converging to a desired altitude and turning radius in $0 \leq t < T_{F_1}$.
- The UAV will be positioned in the center of the circle while maintaining its altitude in a time $T_{F_1} \leq t < T_{F_2}$.
- The asymptotic landing to the center of the circle and altitude $h = 0$ in an interval represented as $T_{F_2} \leq t \leq T_F$.

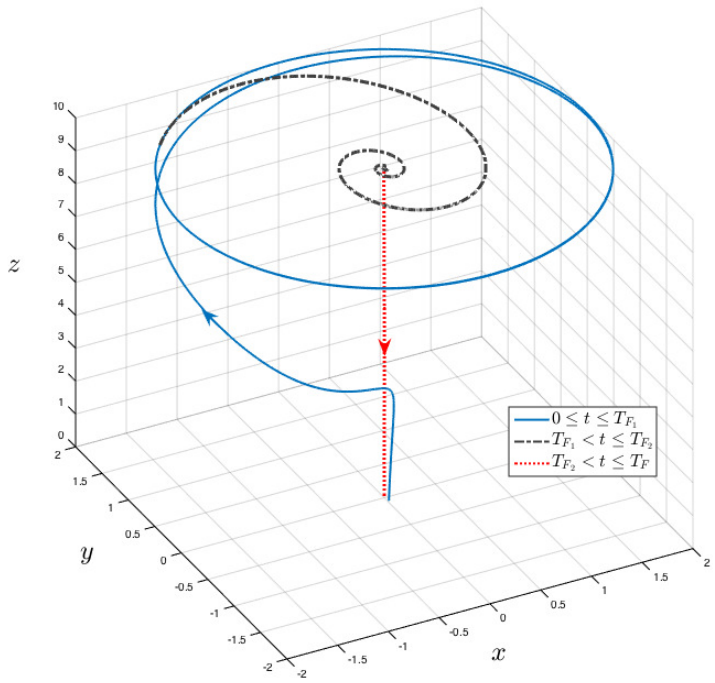


FIGURE 5.2: Three stages are accomplished: circular trajectory, central positioning and landing.

Instead of using heuristic approaches to define the trajectory and its stages, a mathematical formulation is proposed to design the conceived flight trajectory.

5.1.1.1 Mathematical justification

Consider a nonlinear system with a single equilibrium point at the origin as

$$\begin{array}{rcl} \dot{x} & = & \mu x + y - x(x^2 + y^2) \\ \dot{y} & = & -x + \mu y - y(x^2 + y^2) \end{array}, \tag{5.1}$$

where μ represents a bifurcation parameter.

The evolution of this system is such that for the same initial conditions at the equilibrium point, it will behave differently for distinct μ values. More specifically, for $\mu \leq 0$ the origin will behave as an attractor, and any solution will consist on a stable spiral, however if $\mu > 0$ then the origin will be a repeller, and any solution results in an unstable spiral that grows out of the origin, see Figure 5.3

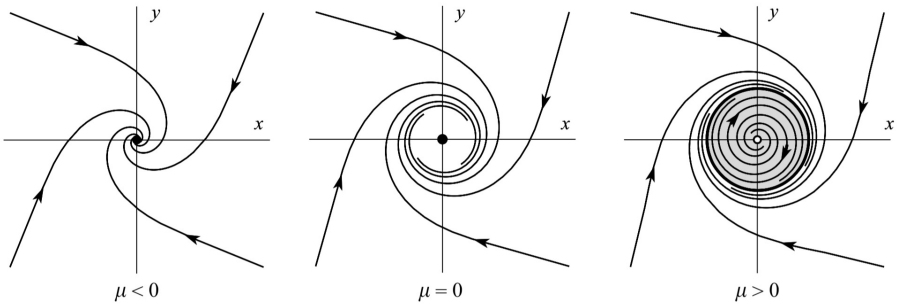


FIGURE 5.3: Development of a limit cycle in a Hopf bifurcation

Proposing the circular relation $r^2 = x^2 + y^2$, where r denotes the desired radius of a circular trajectory, if the bifurcation parameter is selected as $\mu = r^2 > 0$ then $r = \sqrt{\mu}$. For more mathematical background, refer to [99].

To complete a flight trajectory, an altitude variable z is included to (5.1). Define the error $e_z = z - h$, where h is the desired altitude, considered to be a constant. Then an asymptotic convergence for e_z, can easily be achieved by proposing $\dot{e}_z = \dot{z} = -ke_z$ with $k > 0$, then $z \to h$ when $t \to \infty$.

This extends (5.1) with

$$\dot{z} = -k\left(z - h\right) \tag{5.2}$$

The nonlinear equations can then represented as a dynamic system in terms of the trajectory's velocity and acceleration

$$\frac{d}{dt}\begin{bmatrix}\vec{p}_d \\ \dot{\vec{p}}_d\end{bmatrix} = \frac{d}{dt}\begin{bmatrix}x_d \\ y_d \\ z_d \\ \dot{x}_d \\ \dot{y}_d \\ \dot{z}_d\end{bmatrix} = \begin{bmatrix}\dot{x}_d \\ \dot{y}_d \\ \dot{z}_d \\ \left(\mu - 3x_d^2 - y_d^2\right)\dot{x}_d + \left(1 - 2x_d y_d\right)\dot{y}_d \\ -\left(1 + 2y_d x_d\right)\dot{x}_d + \left(\mu - x_d^2 - 3y_d^2\right)\dot{y}_d \\ -k\dot{z}_d\end{bmatrix} \tag{5.3}$$

with any initial conditions defined as $x_d\left(0\right), y_d\left(0\right), z_d\left(0\right)$ for the position, and for the velocity as

$$\dot{\vec{p}}(0) = \begin{bmatrix}\dot{x}_d(0) \\ \dot{y}_d(0) \\ \dot{z}_d(0)\end{bmatrix} = \begin{bmatrix}\mu x_d\left(0\right) + y_d\left(0\right) - x_d\left(0\right)\left(x_d\left(0\right)^2 + y_d\left(0\right)^2\right) \\ -x_d\left(0\right) + \mu y_d\left(0\right) - y_d\left(0\right)\left(x_d\left(0\right)^2 + y_d\left(0\right)^2\right) \\ kh\end{bmatrix}$$

Note that (5.3) only expresses the first stage of the trajectory, when the UAV is lifting and converging to the desired circle. In order to include more flight steps, a switching technique between multiple differential equations is introduced.

When the trajectory starts, since $\mu > 0$, then $(0,0)$ behaves as a repeller in the $x-y$ plane, but since the desired height h is an attractor, the trajectory converges to a stable circle at the desired height.

Define a time T_{F_1} when the equilibrium point $(0,0,h)$ can be switched into an attractor, thus any solution will consist on a stable spiral that converges to the circle center at the same height as the circle.

Likewise, at another time T_{F_2}, a switching can be the applied to get a third differential equation, where the equilibrium point $(0,0,h)$ will be converted from an attractor to a repeller, but now the equilibrium point $(0,0,0)$ becomes an attractor, thus taking the trajectory to a landing stage, see Figures 5.1 and 5.2.

Therefore, applying the switching technique, the nonlinear differential equations that govern the lifting, hovering, and landing stages of the trajectory are given by

$$\frac{d}{dt}\begin{bmatrix} x_d \\ y_d \\ z_d \\ \dot{x}_d \\ \dot{y}_d \\ \dot{z}_d \end{bmatrix} = \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{z}_d \\ (1-\tilde{f}(t))\left\{(\mu - 3x_d^2 - y_d^2)\,\dot{x}_d + (1-2x_d y_d)\,\dot{y}_d\right\} - \tilde{f}(t)\left(k_{p1}x_d + k_{d1}\dot{x}_d\right) \\ (1-\tilde{f}(t))\left\{(\mu - x_d^2 - 3y_d^2)\,\dot{y}_d - (1+2y_d x_d)\,\dot{x}_d\right\} - \tilde{f}(t)\left(k_{p2}y_d + k_{d2}\dot{y}_d\right) \\ \left(\bar{f}(t)-1\right)k\dot{z}_d - \bar{f}(t)\left(k_{p3}z_d + k_{d3}\dot{z}_d\right) \end{bmatrix}, \tag{5.4}$$

where

$$\tilde{f}(t) = \begin{cases} \exp\left(n\left(t - T_{F1}\right)\right) & \forall\ 0 \le t < T_{F1} \\ 1 & \forall\ \ t \ge T_{F1} \end{cases},$$
$$\bar{f}(t) = \begin{cases} \exp\left(n\left(t - T_{F2}\right)\right) & \forall\ 0 \le t < T_{F2} \\ 1 & \forall\ \ t \ge T_{F2} \end{cases}, \tag{5.5}$$

and $T_{F1} \le T_{F2} < T_F$ represent the time of each one of the flight stages, $n > 0$ and k_{p_i}, k_{d_i} are given by

$$\begin{aligned} k_{p_i} &= 2\alpha_i \\ k_{d_i} &= (\alpha_i^2 + \beta_i^2) \end{aligned} \quad \Bigg| \quad \begin{aligned} &\alpha_i > 0, \beta_i = \text{are positive gains} \\ &\qquad \text{with } i : 1,2,3. \end{aligned}$$

5.1.2 Autonomous circular UGV tracking

For this next part, consider the equilibrium of system (5.1) is set at the coordinates $\vec{p}_G = [x_G\,,\ y_G\,,\ z_G]^T$ of a UGV moving in the ground at a constant speed such that

$$\begin{aligned} \dot{x} - \dot{x}_G &= \mu(x - x_G) + (y - y_G) - (x - x_G)((x - x_G)^2 + (y - y_G)^2), \\ \dot{y} - \dot{y}_G &= -(x - x_G) + \mu(y - y_G) - (y - y_G)((x - x_G)^2 + (y - y_G)^2), \end{aligned} \tag{5.6}$$

note system (5.6) evolves similar to (5.1), with the difference that its origin is located at the coordinates of the UGV, which will also behave as an attractor or repeller according to μ. Differentiating the previous equations yields to the following system in terms of the trajectory's velocity and acceleration, considering $\ddot{\vec{p}}_G \approx 0$

$$\begin{aligned} \ddot{x}_d &= (\mu - 3(x_d - x_G)^2 - (y_d - y_G)^2)\ (\dot{x}_d - \dot{x}_G) \\ &\quad + (1 - 2(x_d - x_G)(y_d - y_G))\ (\dot{y}_d - \dot{y}_G), \\ \ddot{y}_d &= (\mu - (x_d - x_G)^2 - 3(y_d - y_G)^2)\ (\dot{y}_d - \dot{y}_G) \\ &\quad - (1 + 2(y_d - y_G)(x_d - x_G))\ (\dot{x}_d - \dot{x}_G), \\ \ddot{z}_d &= -k(\dot{z}_d - \dot{z}_G). \end{aligned} \tag{5.7}$$

Finally, applying the switching technique from (5.5) the nonlinear differential equations that govern lifting, hovering, and landing stages in coordination with the UGV kinematic behavior are given by

$$\begin{aligned} \ddot{x}_d &= (1 - \tilde{f}(t))\,\{(\mu - 3(x_d - x_G)^2 - (y_d - y_G)^2)\ (\dot{x}_d - \dot{x}_G) \\ &\quad + (1 - 2(x_d - x_G)(y_d - y_G))\ (\dot{y}_d - \dot{y}_G)\} \\ &\quad - \tilde{f}\,(t)\,(k_{p1}(x_d - x_G) + k_{d1}(\dot{x}_d - \dot{x}_G))\,, \\ \ddot{y}_d &= (1 - \tilde{f}\,(t))\,\{(\mu - (x_d - x_G)^2 - 3(y_d - y_G)^2)\ (\dot{y}_d - \dot{y}_G) \\ &\quad - (1 + 2(y_d - y_G)(x_d - x_G))\ (\dot{x}_d - \dot{x}_G)\} \\ &\quad - \tilde{f}\,(t)\,(k_{p2}(y_d - y_G) + k_{d2}(\dot{y}_d - \dot{y}_G))\,, \\ \ddot{z}_d &= \left(\bar{f}\,(t) - 1\right) k(\dot{z}_d - \dot{z}_G) - \bar{f}\,(t)\,(k_{p3}(z_d - z_G) + k_{d3}(\dot{z}_d - \dot{z}_G))\,. \end{aligned} \tag{5.8}$$

To compute the final trajectory for its implementation in simulations and real-world experiments, two cascaded numeric integrators are applied to $\ddot{\vec{p}}_d$ to compute $\vec{p}_d$ and $\dot{\vec{p}}_d$ which are finally introduced to a quaternion-based control algorithm

5.1.3 Quaternion-based Control

Following the state feedback controller proposed in Section 3.2, the trajectory described by the solution of (5.8) can be tracked by defining a control force as

$$\vec{F}_u = -K_{pt}\left(\vec{p} - \int_0^t \int_0^t \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} dt\,dt\right) - K_{vt}\left(\dot{\vec{p}} - \int_0^t \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} dt\right) - m\vec{g}, \tag{5.9}$$

where K_{pt}, and K_{dt} denote proportional and derivative control gains.

Then, an attitude control law is applied such that the vehicle's orientation $\boldsymbol{q}$ converges to the desired reference $\boldsymbol{q}_d$, computed according to (2.41)

$$\vec{\tau} = -2K_{pr} \ln\left(\boldsymbol{q} \otimes \boldsymbol{q}_d^*\right) - K_{vr}\vec{\Omega} + \vec{\Omega} \times J\,\vec{\Omega}, \tag{5.10}$$

where K_{pr}, and K_{dr} denote control gains. Controllers (5.9) and (5.10) guarantee system stability, thus tracking the trajectory from Section 5.1.1.1.

5.1.4 Simulations

For the numerical validation, the trajectory parameters were selected such that the simulated circle is $1m$ radius:

$$k_{p1} = k_{p2} = k_{p3} = k_{d1} = k_{d2} = k_{d3} = 1, n = 12, \quad r = 1 \rightarrow \mu = 1, \tag{5.11}$$

concerning the quaternion control law, the gains were empirically chosen as

$$\begin{array}{ll} K_{pt} = \mathrm{diag}([5,5,5]), & K_{pr} = \mathrm{diag}([50,50,50]) \\ K_{dt} = \mathrm{diag}([4,4,4]), & K_{dr} = \mathrm{diag}([10,10,10]) \end{array}. \tag{5.12}$$

Random noise (with values of $\pm 0.5m$) has been added to the position feed-backed signals to validate the robustness of this configuration.

T_{F1}, T_{F2}, and T_F were considered to be increasing until a command for changing the flight step is given. This was accomplished in the following way:

1. When the quad-rotor is lifting and converging to the desired altitude and radius, T_{F1} is considered to be 10 seconds greater than the current simulated time, and T_{F2} is considered to be 10 seconds greater than T_{F1}.
2. When the quad-rotor is commanded to position in the center of the circle, T_{F1} is frozen in its last value, while T_{F2} is considered to be 10 seconds greater than the current time.
3. When the asymptotic landing is actioned, T_{F2} is frozen in its last value, thus making the UAV's position converge to the origin.

Figures 5.4 and 5.5 illustrate the quad-rotor's simulated performance while tracking the desired trajectory. It can be seen that the vehicle's position (solid line) actively pursues the trajectory reference (dotted line).

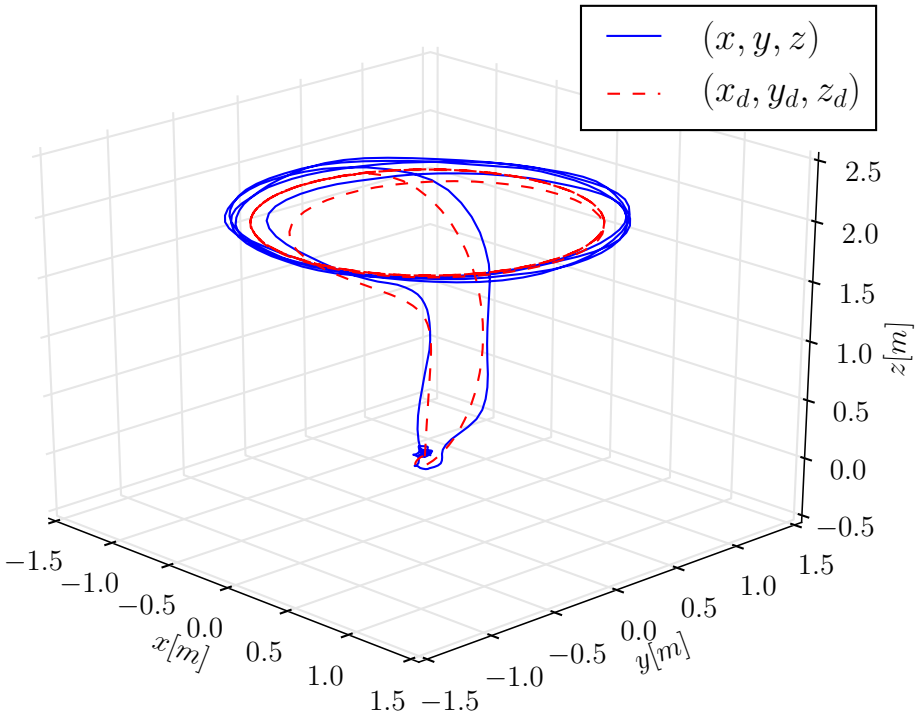


FIGURE 5.4: Simulated trajectory tracking of the quad-rotor, only two stages are accomplished

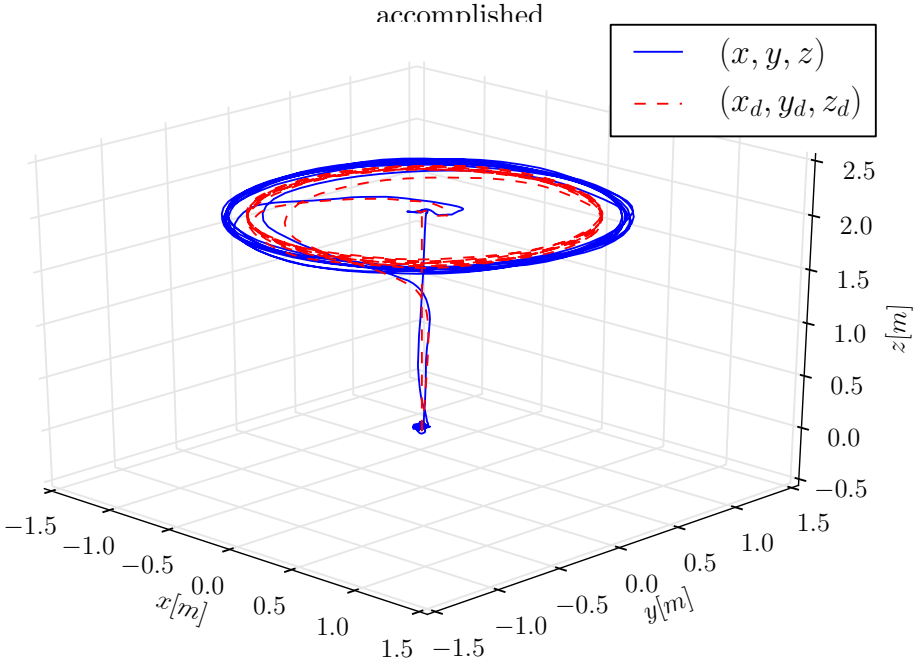


FIGURE 5.5: Quad-rotor's simulated trajectory tracking, The UAV is positioned over the UGV before landing.

5.1.5 Experimental Validation

Our team has made efforts to validate our proposed schemes by performing tests in our prototypes. In this case, the selected drone was a Parrot™AR drone 2 running the proposed algorithms programmed using the Fl-Air framework over a Linux-based operating system, [95].

The selected UGV was a Wifibot, by Nexter Robotics™, which is a miniature vehicle provided with 4 wheels and several sensors, useful at different navigation scenarios.

Both vehicles were connected to an OptiTrack Motion Capture System, which precisely detects their position in an indoor environment.

The trajectory was generated by numerically integrating (5.8), thus updating the translational reference with each iteration of the program.

The parameters of the trajectory were selected identically as in the simulations, while the control gains were considered as

$$K_{pt}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, K_{dt}=\begin{bmatrix}0.5 & 0 & 0\\ 0 & 0.5 & 0\\ 0 & 0 & 0.5\end{bmatrix}, K_{pr}=\begin{bmatrix}5 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 6\end{bmatrix}, K_{pr}=\begin{bmatrix}0.8 & 0 & 0\\ 0 & 0.8 & 0\\ 0 & 0 & 1\end{bmatrix}. \quad (5.13)$$

5.1.5.1 Scenario 1

In the first experiment, the UGV was set to stay still in an arbitrary position, the UAV then takes off from a near-by location, and moves to hover position at $50cm$ over the ground robot. Then, the trajectory sequence starts, the UAV rises 1 meter more above the UGV while describing $1m$ radius circles.

As mentioned in section 5.1.1, two scenarios arrive from this trajectory, depending on the timing configuration and on how many stages are considered.

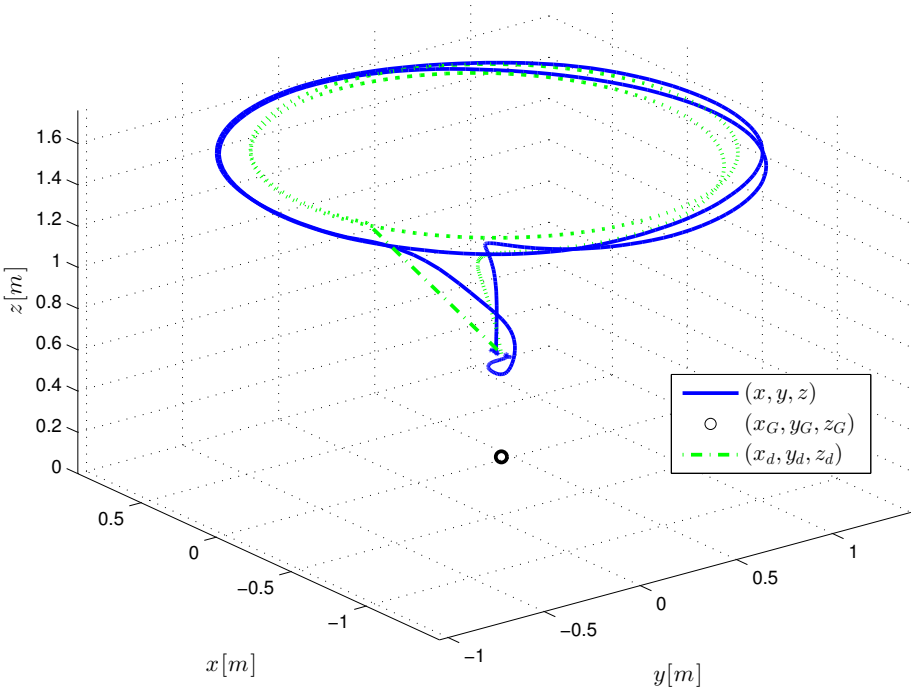


FIGURE 5.6: Quad-rotor's circular trajectory, first scenario: 3D view

Figure 5.6 illustrates the trajectory tracking for the first scenario (only two flight stages accomplished), while the second scenario (three steps of movement) is depicted in Figure 5.7.

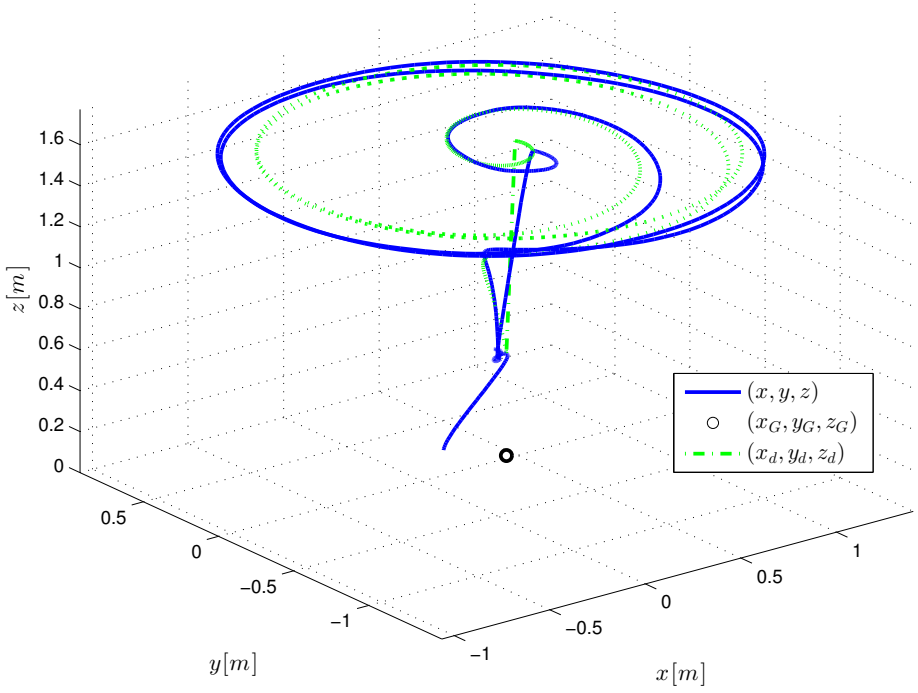


FIGURE 5.7: Quad-rotor's circular trajectory, second scenario: 3D view

In both scenarios the altitude of the UAV starts 0.5 meters above the marker that represents the location of the UGV, and then describes circles at 1 meter more above its initial location above the ground while the circular trajectory is correctly tracked.

Note in the first scenario, that the UAV descends directly from its circular trajectory to the UGV's reference start position, while in the second scenario it aligns horizontally before descending to the ground vehicle's location.

The attitude tracking of the quaternion reference described in section 5.1.3 is illustrated in Figure 5.8 for the first scenario and in Figure 5.9 for the second one. Note that the quad-rotor's quaternion is stabilized in the desired attitude references.

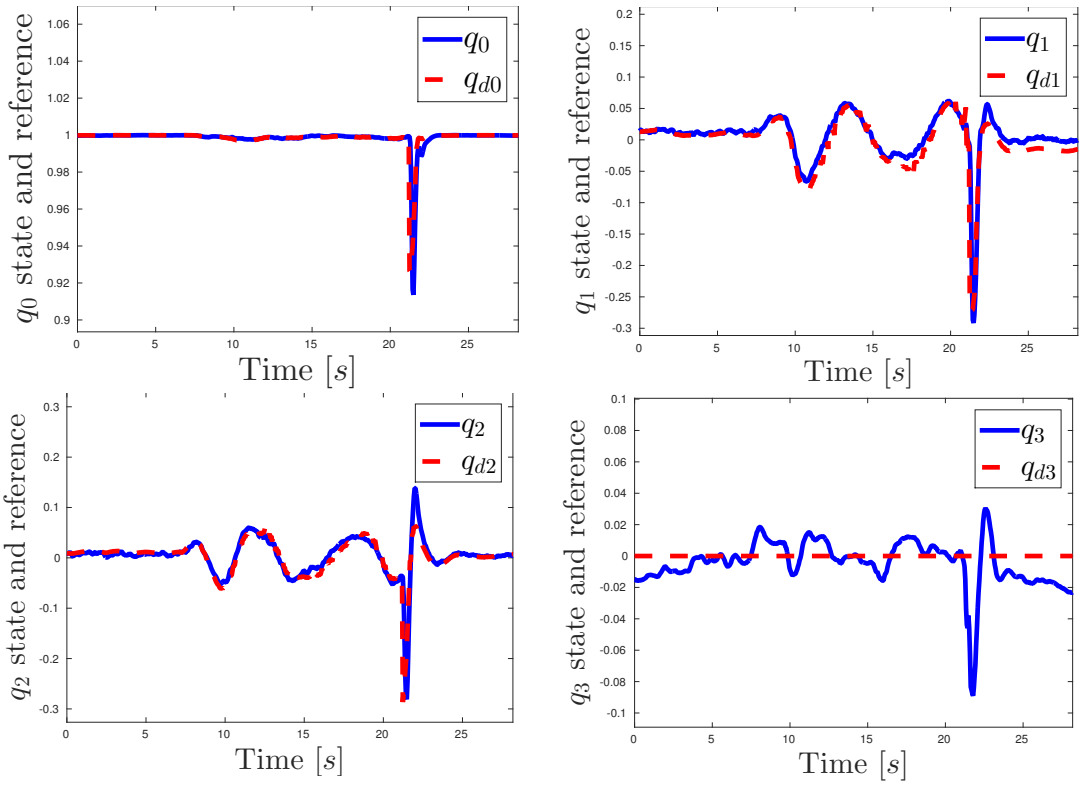


FIGURE 5.8: First scenario's quaternion trajectory tracking

Note from Figure 5.8, that a peak is present in the quaternion attitude, this means that the quadrotor performs a strong inclination to directly converge from its circular path to its landing position, Figure 5.9 lacks such peak because it is already located at the circle center, so no inclination is needed in its descending path.

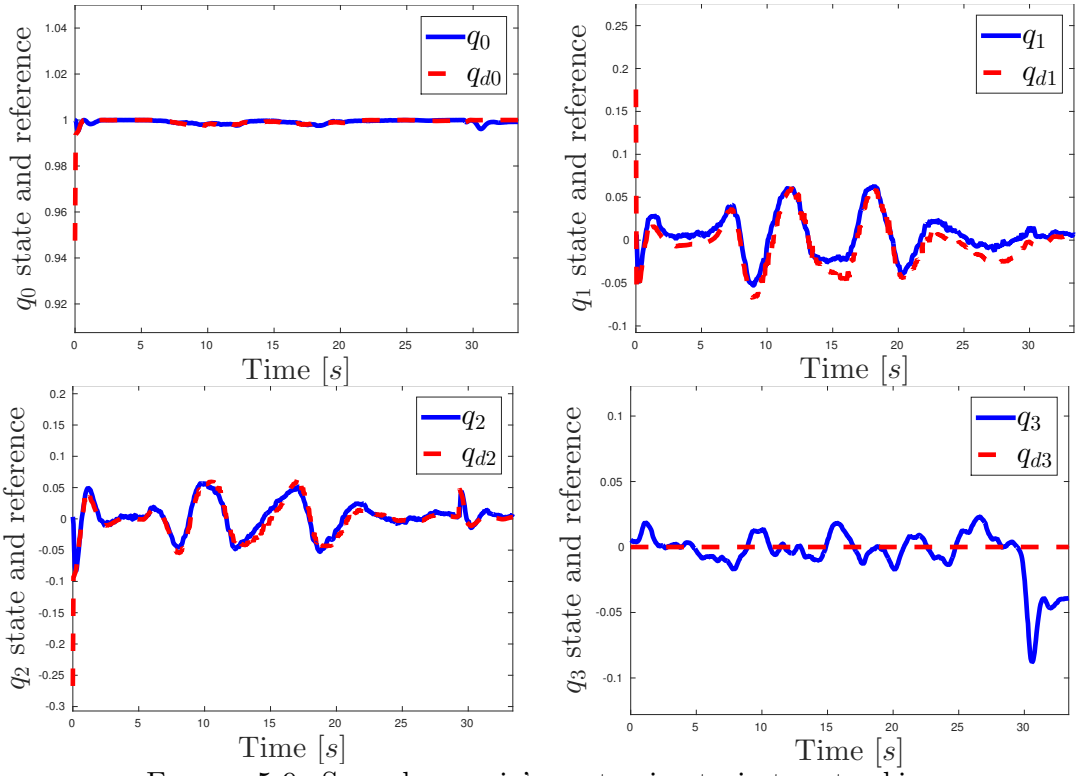


FIGURE 5.9: Second scenario's quaternion trajectory tracking

5.1.5.2 Scenario 2

For this experiment, a dynamic ground vehicle was considered. The UGV was manually moved to randomly selected references in the horizontal plane. Before the UGV starts to navigate, the UAV was stabilized 50cm above its target, then the circular trajectory was started slightly before the UGV starts to move.

The first scenario considers that the UAV must return to the UGV's location while it is still flying in circles (Figure 5.10), while the second scenario considers an horizontal alignment with the ground robot before descending to its reference.

A 3-Dimensional view of both flights is given in Figures 5.10 and 5.11, the line that describes the UGV's movement stays in the ground while the UAV tries to describe circles while hovering in the air.

Note that complete circular and semi-circular movements are accomplished when the UGV stops at certain points in its trajectory.

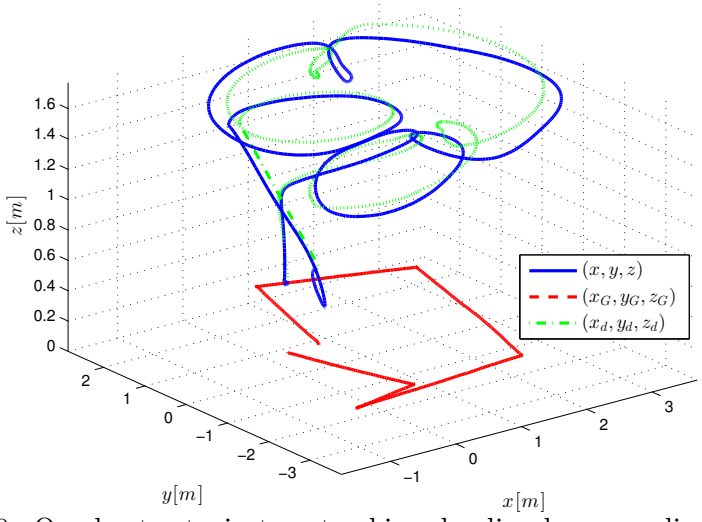


FIGURE 5.10: Quad-rotor trajectory tracking, landing happens directly from the circular path

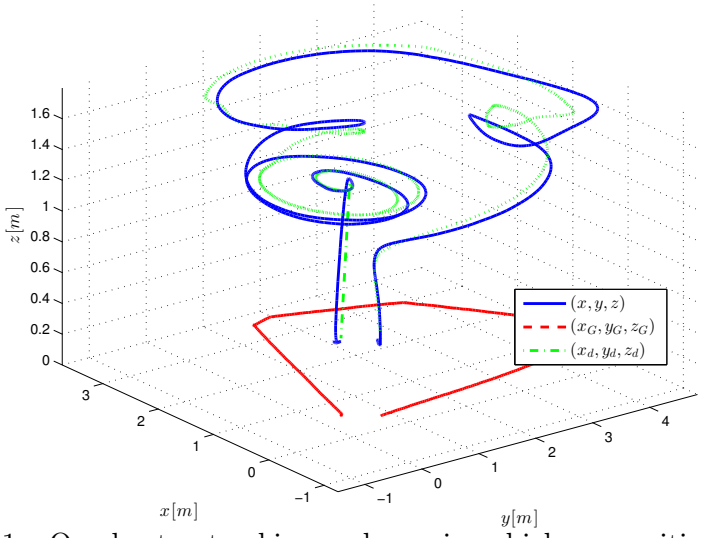


FIGURE 5.11: Quad-rotor tracking a dynamic vehicle, a positioning stage is performed before landing

5.2 Coordinated circular UAV target tracking

The objective of this section is to propose a technique to track a moving Unmanned Ground Vehicle (UGV) which is considered as a target for a pair of quadrotors equipped with front-facing cameras. The UAVs would start from arbitrary positions, and compute a trajectory which takes them towards a formation defined by a desired distance d^d_{iT} with respect the UGV, and a separation $d^d_{ij} = 2d^d_{iT}$ between the quadrotors (see Figure 5.12).

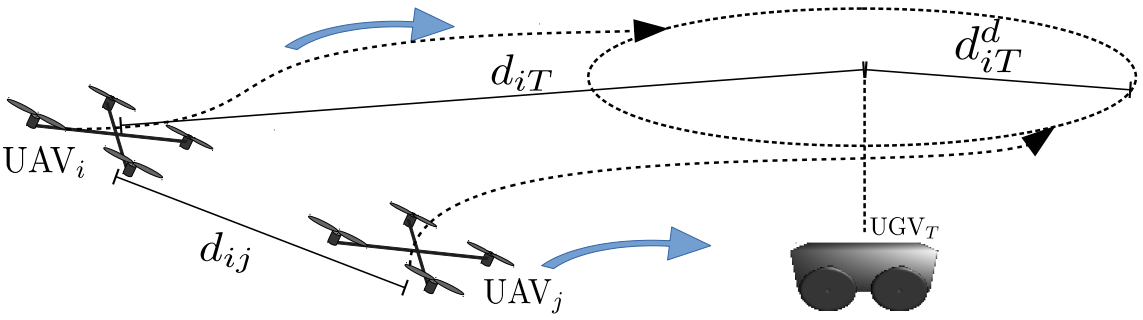


FIGURE 5.12: Two quadrotors take off and move to a symmetrical formation.

As the vehicles approach to the desired configuration, the trajectory would autonomously adapt such that the quadrotors start describing circles around their target while remaining perfectly coordinated by respecting the desired radius and distances, see Figure 5.13. Considering that the quadrotors are equipped with front-facing cameras, their trajectories should ensure that their front axis is always pointing towards the target UGV.

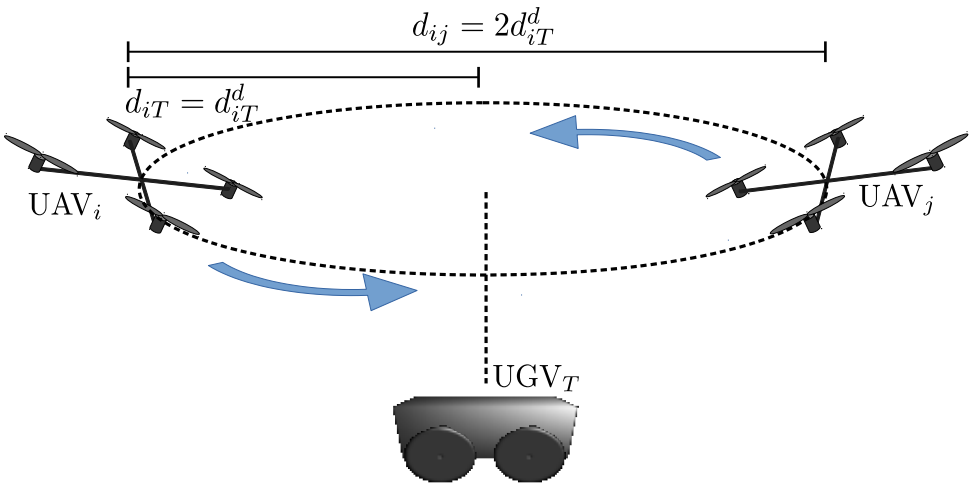


FIGURE 5.13: Autonomously tracking a UGV by describing coordinated circles around it.

5.2.1 Distributed Path Planning Algorithm

A distributed path planning strategy is introduced to generate a trajectory that will take the set of quadrotors from their initial positions to their formation around the target position $p_T(t) \in \mathbb{R}^2$ and, once such configuration is reached, autonomously describe perfectly coordinated circles. The trajectory generation algorithm is coded among the quadrotors as a distributed optimization problem which is solved in real time [100], [101].

Let $\Lambda(t)$ define a positive cost function which value decreases when the horizontal position $p_i(t) \in \mathbb{R}^2$ of both quadrotors is such that the distance $d_{iT}(t)$ to the target, and the separation between them $d_{ij}(t)$ approaches to the desired values. $\Lambda(t)$ is divided between the agents such that each agent i optimizes a part of the function using its own information and that of its neighbor j as:

$$\Lambda(t) := \sum_{i=1}^{n} \Lambda_i(t), \tag{5.14}$$

$$\Lambda_i(t) := \rho\Big|\Big|\|p_T - [p_i(t) + h_i(t)]\| - d_{iT}^d\Big|\Big| + a_{ij}(t)\Big|\Big|\|p_j(t) - [p_i(t) + h_i(t)]\| - d_{ij}^d\Big|\Big|,$$

with

$$a_{ij}(t) = 1 + \exp\left(\frac{c - d_{ij}(t)}{\gamma}\right), \tag{5.15}$$

being $i \neq j$ the index of each drone, $h_i(t) \in R^2$ represents the solution of the cost function for each quadrotor, $\rho, \gamma \in R^+$ denote tunning parameters, and c symbolizes their safety separation.

This path planning strategy (presented in [100] and [102]), consists on finding the optimal displacement $h_i^*(t)$ for each drone $i = 1, 2$, at each time instant while minimizing $\Lambda(t)$. Note that $\Lambda(t) \to 0$ when $d_{ij}(t) \to d_{ij}^d$, $d_{iT}(t) \to d_{iT}^d$ and $h_i^*(t) \to 0$ meaning that the quadrotors will hold their static position once the formation is reached, unless the target changes its position, or an agent is disturbed.

In this work, it is desired that the agents follow a circular path while attaining their configuration, this can be achieved by redefining the cost function as

$$\begin{aligned}\Lambda_i(t) :=& \rho\Big|\Big|\|p_T(t) - [p_i(t) - \eta_i(t) + h_i(t)]\| - d_{iT}^d\Big|\Big| \\ &+ a_{ij}(t)\Big|\Big|\|p_j(t) - [p_i(t) - \eta_i(t) + h_i(t)]\| - d_{ij}^d\Big|\Big|,\end{aligned} \tag{5.16}$$

where $\eta_i(t) \perp (p_T - p_i(t))$ denotes a vector in $\mathbb{R}^2$ which is perpendicular to the radius between quadrotor i and the target, and is computed as

$$\eta_i(t) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left[[0\ 0\ 1]^T \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \left[\frac{(p_T(t) - p_i(t))}{||(p_T(t) - p_i(t))||} \right] \omega_i(t) \right], \tag{5.17}$$

with turning rate defined as

$$\omega_i(t) := \omega_d \exp\left(-\left|\left|\|p_j(t) - p_i(t)\| - d_{ij}^d\right|\right| - \left|\left|\|p_T - p_i(t)\| - d_{iT}^d\right|\right|\right), \tag{5.18}$$

being $\omega_d \in \mathbb{R}$ a parameter defined by the user which dictates how fast and on which direction the circular trajectory will be described.

The construction of $\omega_i(t)$ is such that when the quadrotors are located far from the desired configuration, the turning rate diminishes exponentially, such that the vehicles can move to the required distances in contrast, the closer the UAVs get to a symmetrical formation, then $\omega_i(t) \to \omega_d$.

The minimum $\Lambda_i(h_i(t))$ is reached when both drones converge to positions such that

$$\begin{aligned} \|p_T(t) - [p_i(t) - \eta_i(t) + h_i(t)]\| &\to d_{iT}^d \\ \|p_j(t) - [p_i(t) - \eta_i(t) + h_i(t)]\| &\to d_{ij}^d \\ h_i^*(t) &\to \eta_i(t) \end{aligned} \tag{5.19}$$

meaning that the optimal displacement $h_i^*(t)$ will converge to a perpendicular displacement with respect to the target direction, see Figure 5.14.

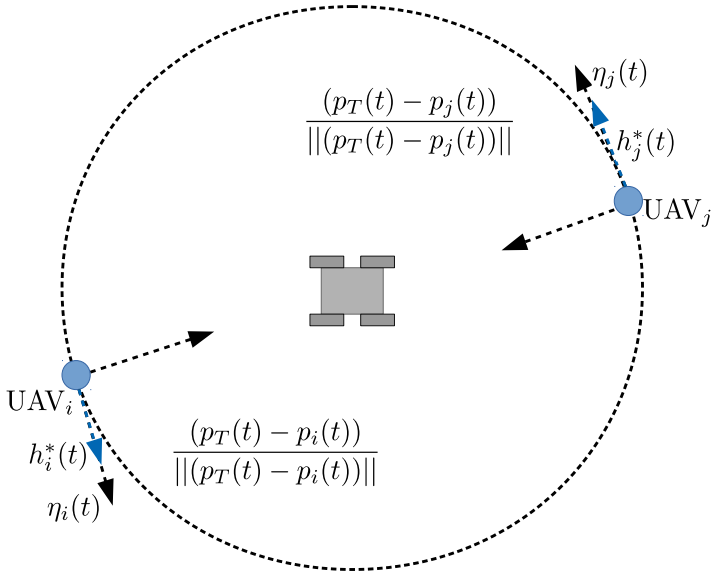


FIGURE 5.14: Circular path generation using perpendicular vectors on the optimization cost function.

The optimal solution of (5.16) is computed by generating N random values $h_{i,n}(t), n = 1, \cdots, N$ contained in a vicinity $0 < h_{i,n}((t) < h_{i\max}$, where $h_{i\max}$ is manually tuned. Each value is then evaluated on the cost function, generating N solutions $\Lambda_{i,n}(h_i(t))$, the best evaluation is used to compute a the velocity and position of all the particles on a recursive loop. The optimal path followed by the drone is defined as $h_i^*(t)$, and is selected as the corresponding minimal solution of $\Lambda_i^*(t)$ using a Particle Swarm Optimization (PSO) algorithm.

$$h_i^*(t) = \arg\min\left(\Lambda_i\big(h_{i,n}(t)\big)\right). \tag{5.20}$$

Finally, the 3-dimensional trajectory is composed by adding $h_i^*(t)$ to the current quadrotor position, resulting in

$$\vec{p}_{di}(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} [p_i(t) + h_i^*(t)] + \begin{bmatrix} 0 \\ 0 \\ z_d \end{bmatrix}, \tag{5.21}$$

where z_d denotes the desired quadrotor altitude over the target UGV.

5.2.2 Target Locking

In the proposed scenario, the two quadrotors are equipped with front-facing cameras, in order to maintain the target in-sight, the front axis of the quadrotors should be pointed towards the position of the UGV.

Define a normalized vector $\vec{\chi}_i \in \mathbb{R}^3$ that points horizontally from the position of quadrotor i to the target location as

$$\vec{\chi}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \frac{p_T - p_i}{||p_T - p_i||}. \tag{5.22}$$

Considering that the front of the quadrotor points towards the x axis of the body frame, and following the idea of (2.40), a quaternion $\boldsymbol{q}_z$ which aligns the vehicle z axis with $\hat{m}$ is proposed as

$$\boldsymbol{q}_{zi} := \pm \left(\sqrt{\frac{1 + \vec{\chi}_i \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{2}} + \frac{\vec{\chi}_i \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{\left|\left| \vec{\chi}_i \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right|\right|} \sqrt{\frac{1 - \vec{\chi}_i \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{2}} \right), \tag{5.23}$$

see Figure 2.5.

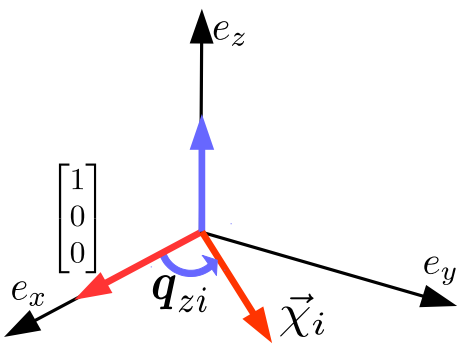


FIGURE 5.15: Aligning the vehicle front-facing axis towards the target direction using a quaternion.

5.2.3 Control Algorithm

In order to have a smooth and robust response on the quadrotor, a position controller is developed on each quadrotor by introducing the PSO-generated trajectory from (5.21) into the cylindrical bounded controller from (3.83) which yields a control force computed as

$$\vec{F}_{ui} = -mK_t\sigma_c\left(K_c\dot{\vec{p}}_i + \sigma_d\left(K_dK_c\begin{bmatrix}-h^*_{ix}(t)\\-h^*_{iy}(t)\\p_{iz}(t)-z_d\end{bmatrix} + K_d\dot{\vec{p}}_i\right)\right) - m\vec{g}, \tag{5.24}$$

which then yields an attitude reference for every vehicle, derived from (2.40) as

$$\boldsymbol{q}_{ti} = \pm\left(\sqrt{\frac{1+\frac{\vec{F}_{ui}}{||\vec{F}_{ui}||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}} + \frac{\frac{\vec{F}_{ui}}{||\vec{F}_{ui}||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}}{\left|\left|\frac{\vec{F}_{ui}}{||\vec{F}_{ui}||}\times\begin{bmatrix}0\\0\\1\end{bmatrix}\right|\right|}\sqrt{\frac{1-\frac{\vec{F}_{ui}}{||\vec{F}_{ui}||}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}}{2}}\right), \tag{5.25}$$

finally, a bounded control torque, which include the target-locking rotation from (5.23) as

$$\vec{\tau}_i = \vec{\Omega}_i\times J\vec{\Omega}_i - JK_r\sigma_a\Bigg(K_a\vec{\Omega}_i \tag{5.26}$$

$$+\sigma_b\left(2K_bK_a\ln\left(\pm\left(\sqrt{\frac{1+\vec{\chi}_i\cdot\begin{bmatrix}1\\0\\0\end{bmatrix}}{2}} - \frac{\vec{\chi}_i\times\begin{bmatrix}1\\0\\0\end{bmatrix}}{\left|\left|\vec{\chi}_i\times\begin{bmatrix}1\\0\\0\end{bmatrix}\right|\right|}\sqrt{\frac{1-\vec{\chi}_i\cdot\begin{bmatrix}1\\0\\0\end{bmatrix}}{2}}\right)\otimes\boldsymbol{q}^*_{ti}\otimes\boldsymbol{q}_i\right) + K_b\vec{\Omega}_i\right)\right).$$

In this work, both quadrotors are considered to have identical mechanical configurations, for this reason all control gain matrices, bounding limits, mass and inertia matrices are considered to be the same from one vehicle to the other.

5.2.4 Emulated Results

The proposed algorithm was implemented on a simulation environment included in the "FL-air" framework for UAVs, developed at the Heudiasyc laboratory [95].

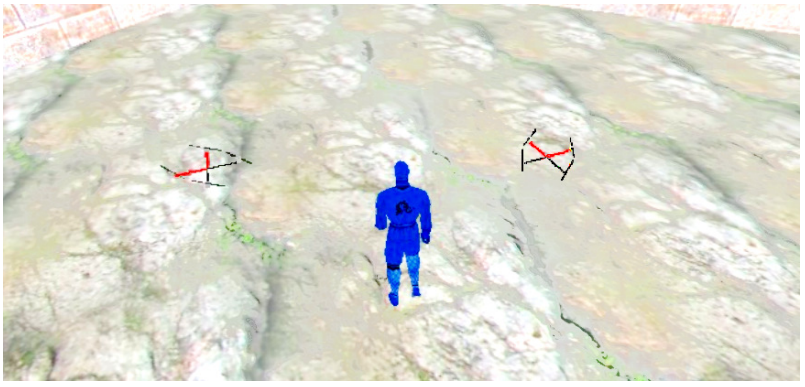


FIGURE 5.16: FL-air framework simulation environment with a human target and two tracking quadrotors.

In this case, the simulator was set to include two quadrotors and a human considered as a ground target. Figure 5.17 illustrates the movement of the target, and the two drones following the desired formation around it. The colored arrows indicate the direction of each vehicle front axis which are computed by rotating a unit vector with the attitude quaternion as $e^b_{xi} = \boldsymbol{q}_i \otimes [1\ 0\ 0]^T \otimes \boldsymbol{q}_i^*$

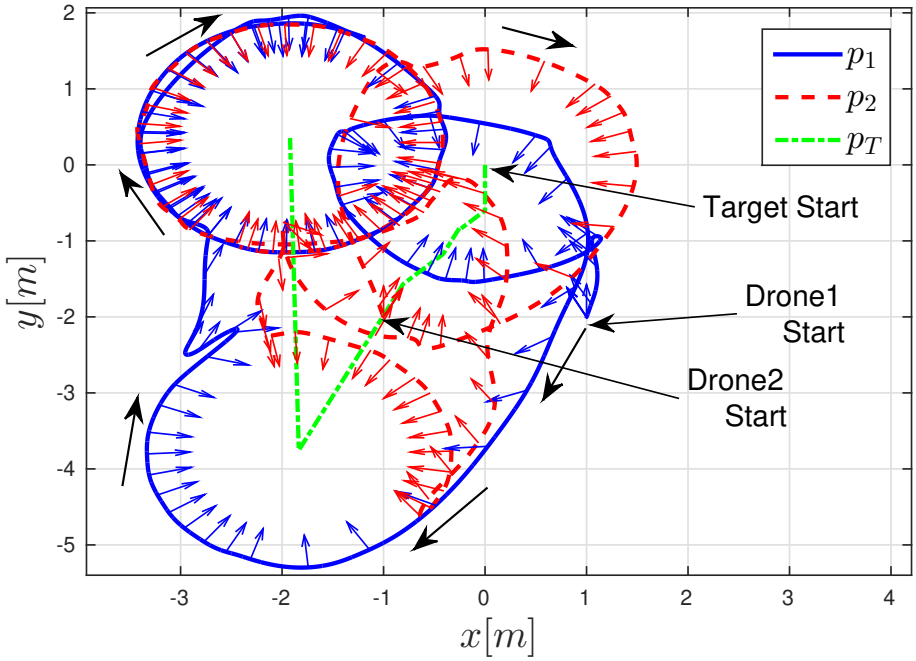


FIGURE 5.17: Two quadrotors following a target on an emulated environment, the UAVs describe circles while pointing their fronts towards the objective.

Figures 5.18 and 5.19 illustrate the translational behavior of both drones on the x and y axes respectively, fine-dotted lines illustrate the reference trajectories computed by the PSO algorithm, while continuous and large-dotted lines represent the response of each vehicle.

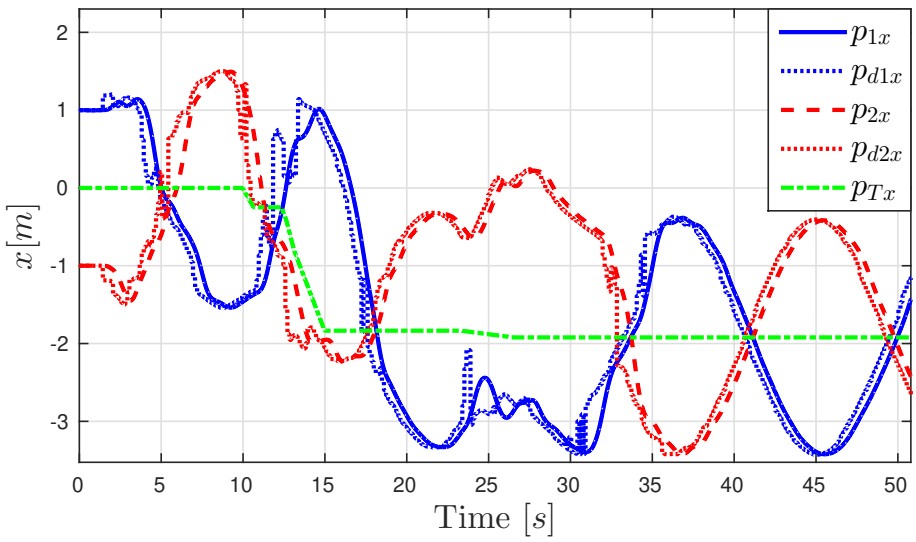


FIGURE 5.18: Translation over the x axis, the circular tracking of the target is indicated by sinusoidal oscillations on both quadrotors.

Both quadrotors use the PSO algorithm to solve the optimization problem from (5.16), resulting in trajectories defined by following (5.21). Using controllers (5.24) and (5.26), the vehicles are stabilized to their corresponding trajectories

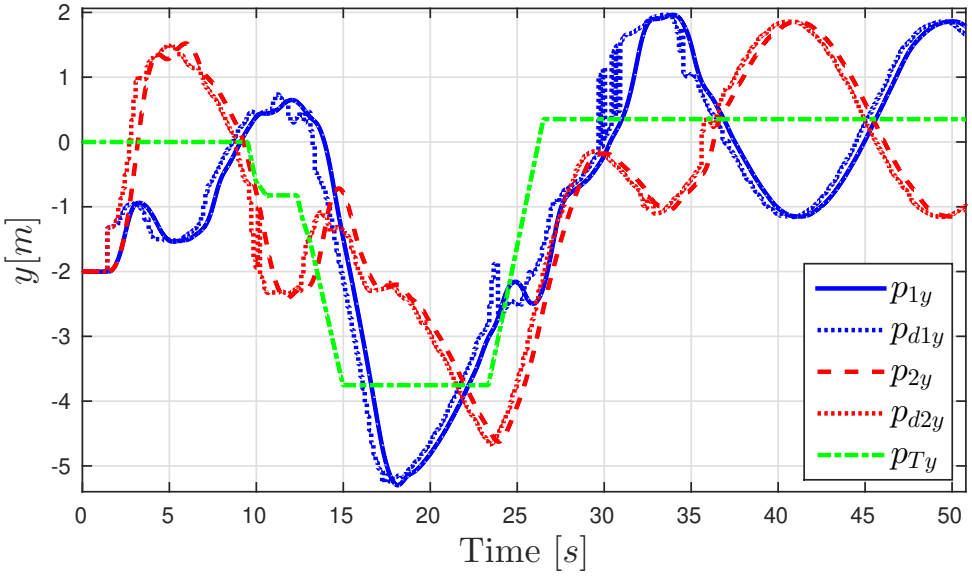


FIGURE 5.19: y axis position response, note how both quadrotors oscilate at oposite locations from the target, indicating their coordinated behavior.

Figure 5.20 depicts the distances between both vehicles and with respect to their target, note the PSO algorithm along with the control laws ensure the desired distances will be respected even when the quadrotors are following circular paths.

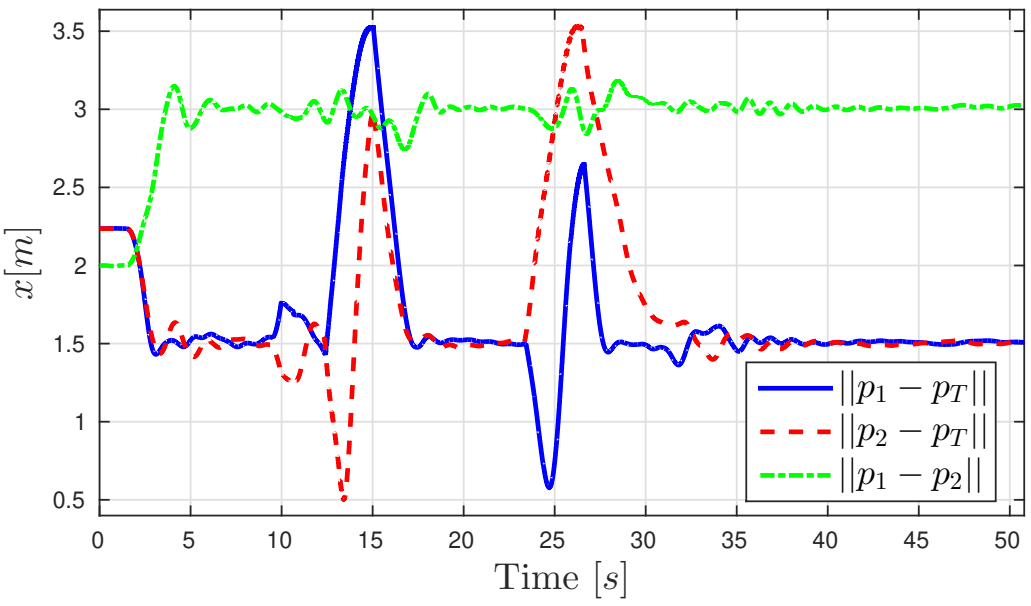


FIGURE 5.20: Distance between quadrotors and from each agent to the target, the desired radius was set as $d^d_{iT} = 1.5m$, therefore $d^d_{ij} = 2d^d_{iT} = 3m$.

5.2.5 Experimental validation

Two real world experiments were performed using AR-Drone2 quadrotors to track a Parrot Jumping Race target UGV as depicted on Figure 5.21.

The trajectory generation and control strategies were programmed on the same framework as on the real-time emulation, each drone computes its corresponding part of the algorithms.

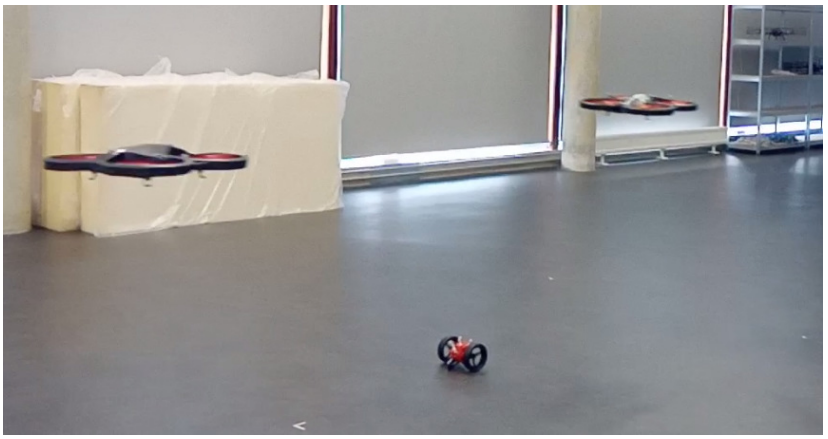


FIGURE 5.21: Experimental set for circular UAV-UGV tracking, a UGV is set as a target for two quadrotors

An OptiTrack motion capture system was used to estimate the vehicles positions and translational velocities and broadcasted using a WiFi network.

5.2.5.1 Stationary Target Scenario

On the first scenario, the target was left in a fixed position while the two quadrotors kept track of it, their coordinated path resulted in circular trajectories, as illustrated on Figure 5.22.

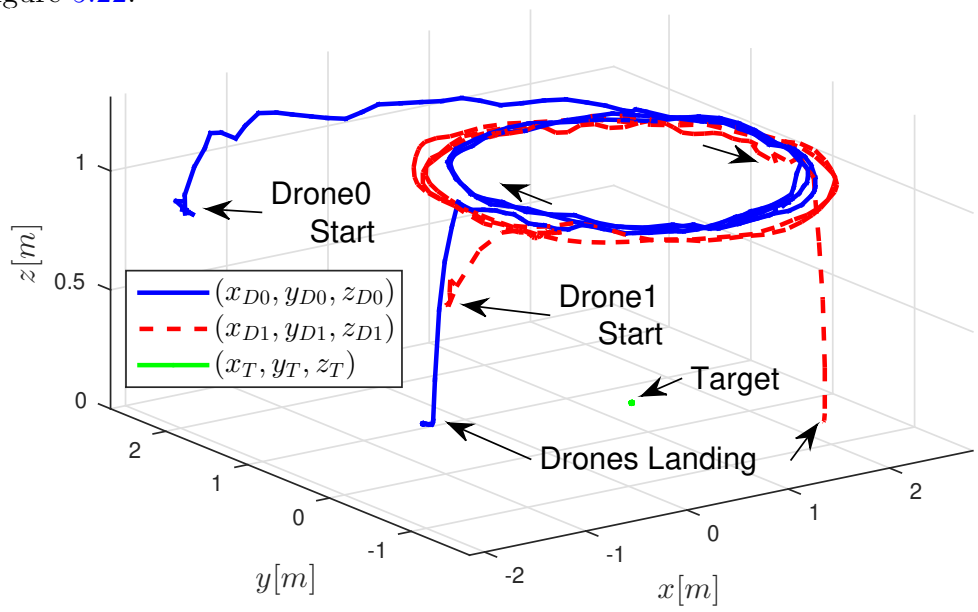


FIGURE 5.22: Two quadrotors tracking a static UGV, the vehicles describe coordinated circles around the target.

The desired trajectory is computed following the proposed equations on each UAV using the PSO algorithm, the control algorithm then tracks the computed references, as illustrated in Figures 5.23 and 5.24. Notice the position of the quadrotors is opposite one from another, indicating a coordinated circular path.

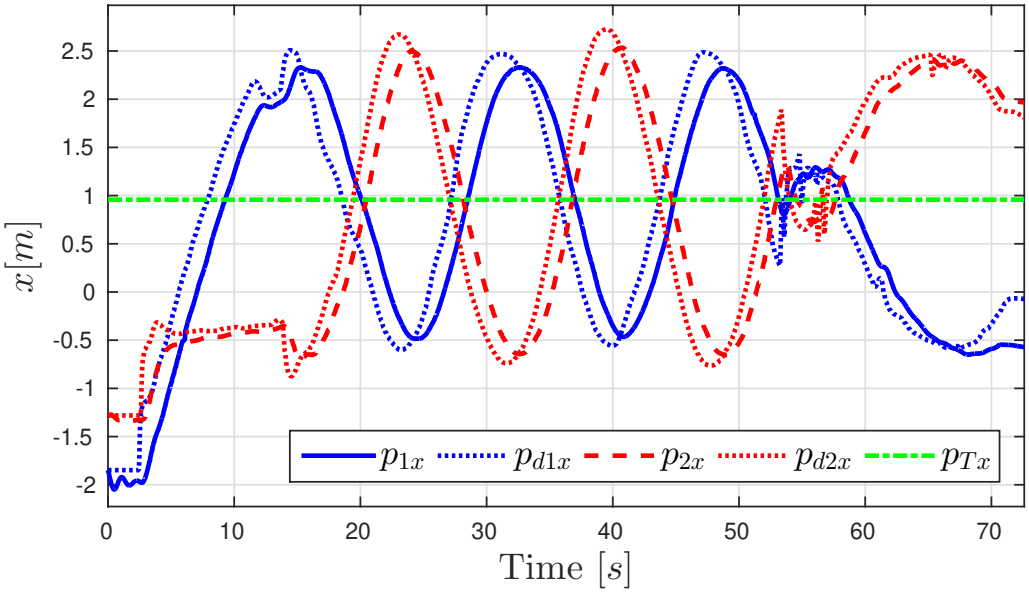


FIGURE 5.23: x axis position of both vehicles during a static target test, fine-dotted lines represent the trajectory generated by the PSO algorithm.

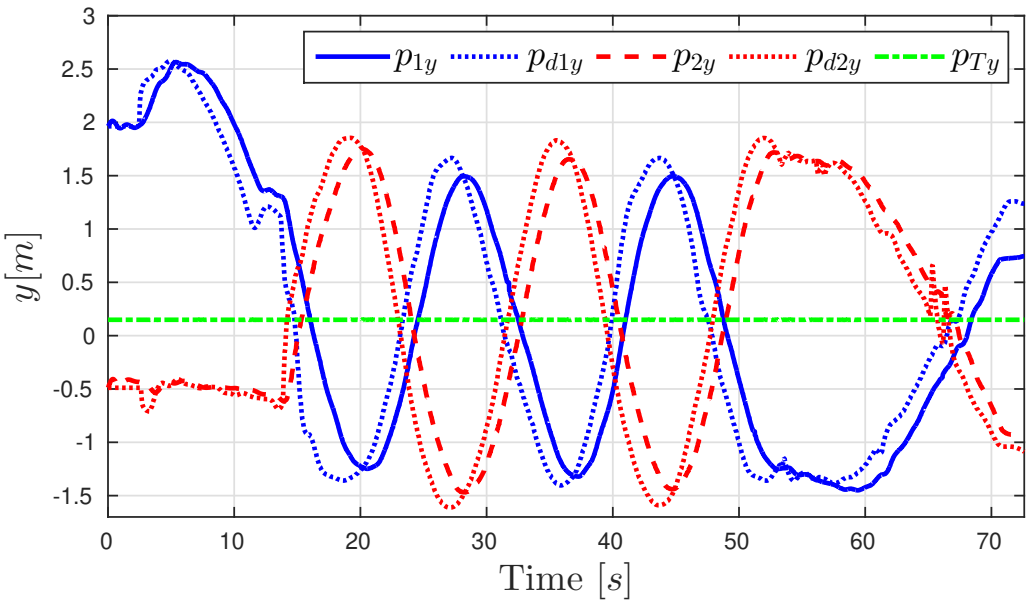


FIGURE 5.24: Translational y axis response for a static target experiment, opposing oscilations indicate how coordinated circles are followed.

The PSO algorithm optimizes the position of each drone according to (5.16) such that the required distances are respected as illustrated in Figure 5.25, in this experiment, the desired radius was set as $d^d_{iT} = 1.5m$, therefore $d^d_{ij} = 2d^d_{iT} = 3m$.

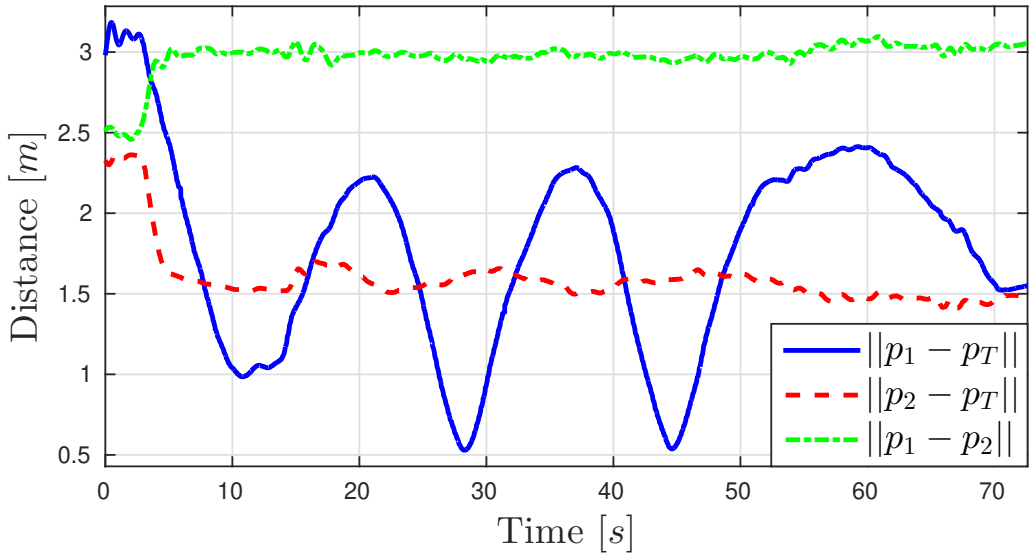


FIGURE 5.25: Static target experiment, the trajectory generation algorithm optimizes the distance between quadrotors and from each agent to the target.

In order to lock the quadrotor's view of the target, a rotation around the vehicle z axis is computed by (5.23) to point the x axis of the body reference frame towards the UGV location. Figure 5.26 illustrates this behavior using colored arrows, since the target is static, they always point towards the circular path's center.

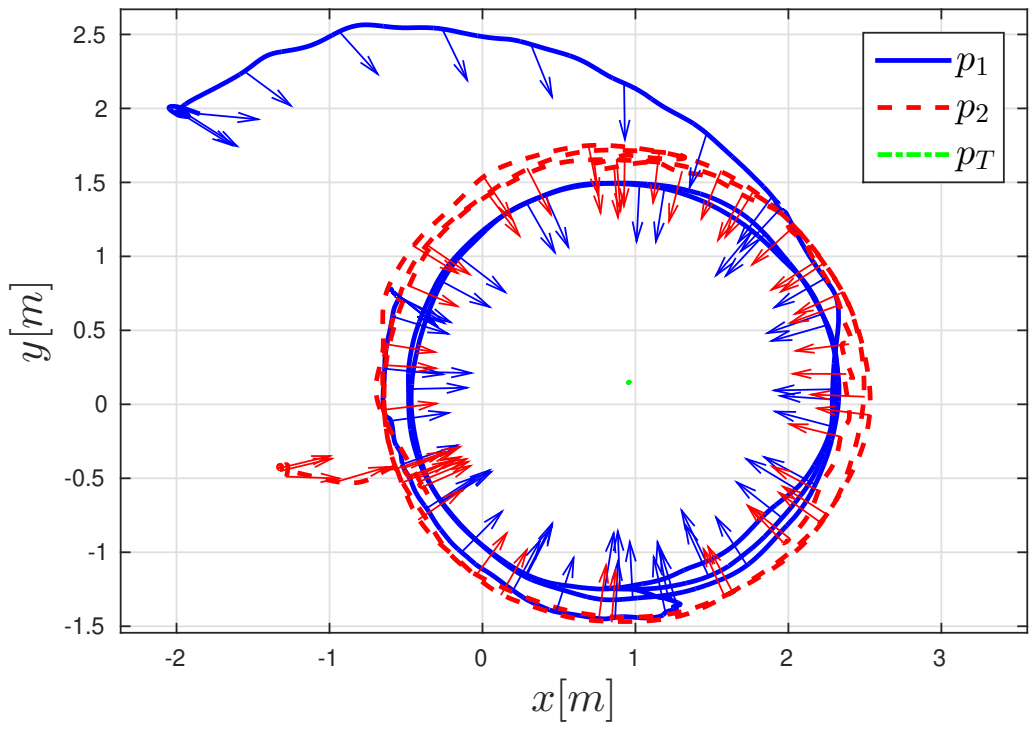


FIGURE 5.26: Circular tracking of a fixed UGV, arrows illustrate the front axis of each quadrotor which is pointed towards the center.

Equations (2.40) and (5.23) yield a quaternion reference which is then tracked by the attitude controller. Figures (5.27) and (5.28) depict the quaternion q_0 and q_3 elements which respectively relate to the total attitude angle, the z axis rotation.

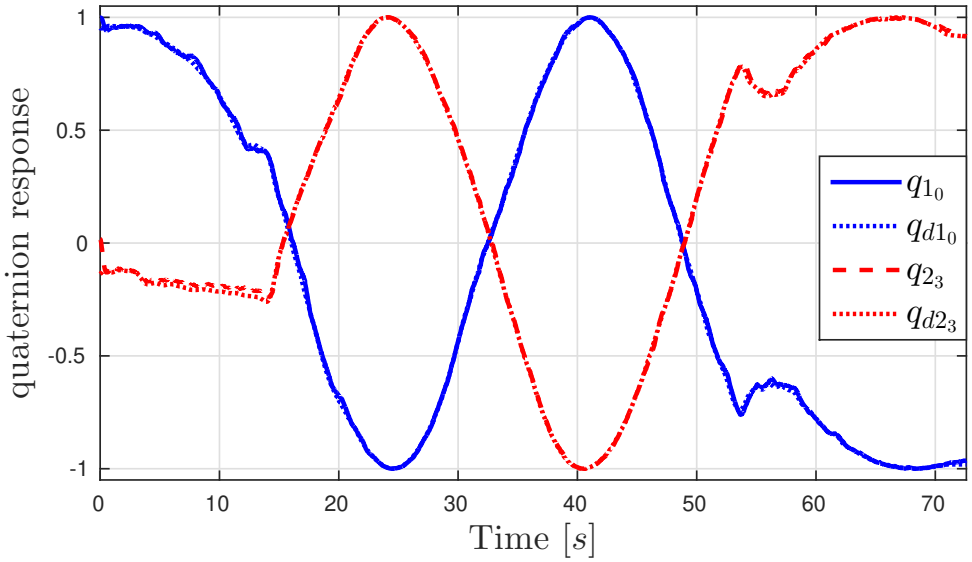


FIGURE 5.27: Quaternion reference and response of both quadrotors over the z axis, sinusoidal signals are described to face towards the target.

Since the vehicle front axis tracks the circle center, the z axis rotation of both quadrotors describe supplementary angles, since the definition of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ depends on sinus and cosinus functions from the Euler-Rodrigues formula (2.15), this results in a behavior such that $q_{1_0} \approx -q_{2_3}$ and $q_{1_3} \approx q_{2_0}$.

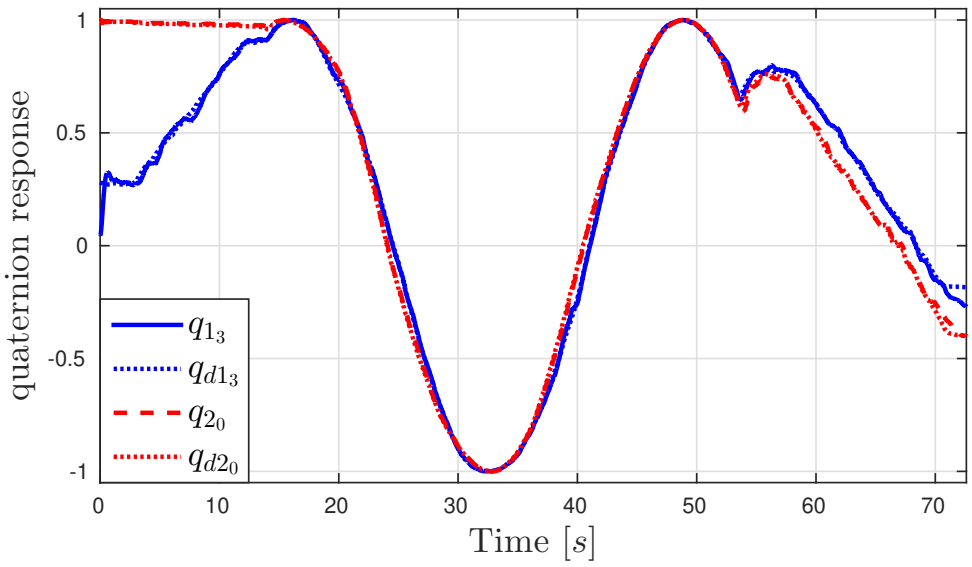


FIGURE 5.28: Attitude over the z axis of both quadrotors, opposing positions over the circular path yield similar values in the rotational signals.

5.2.5.2 Moving Target Scenario

On the second experiment, the UGV was moved using a remote controller, the PSO algorithm generates trajectories for each quadrotor to reach the target, once the desired distances are achieved, circular movements are performed around the ground vehicle. If the target moves, the effects of (5.18) interrupt the quadrotors' circular paths until the formation is recovered, see Figure 5.29.

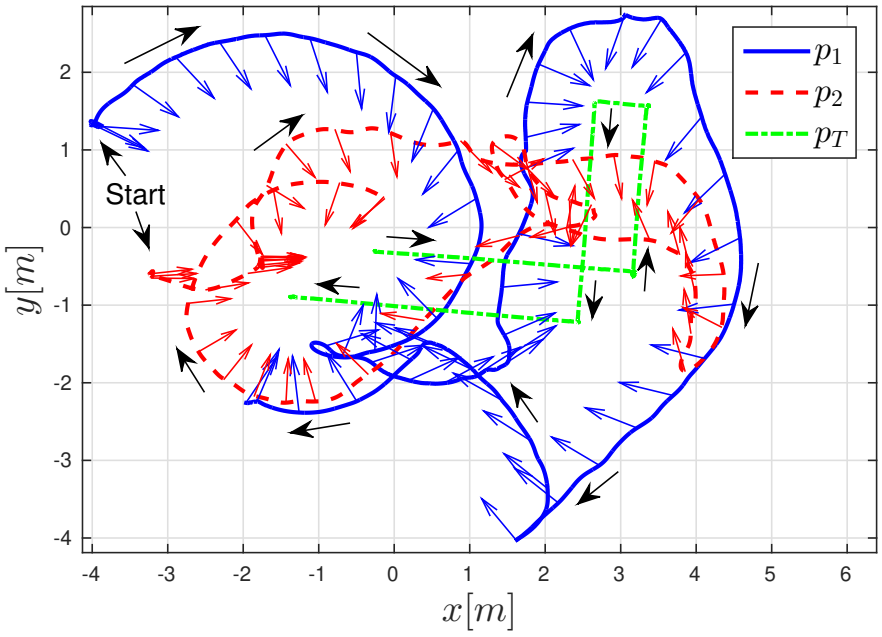


FIGURE 5.29: Two quadrotors following a moving UGV. The vehicles' paths are indicated by black arrows while colored ones represent the vehicles front axes.

Figures 5.30 and 5.31 illustrate the translation of the UGV and both quadrotors on the x and y axes, Notice when the target moves, the drones adjust their trajectories to reach opposing positions around it.

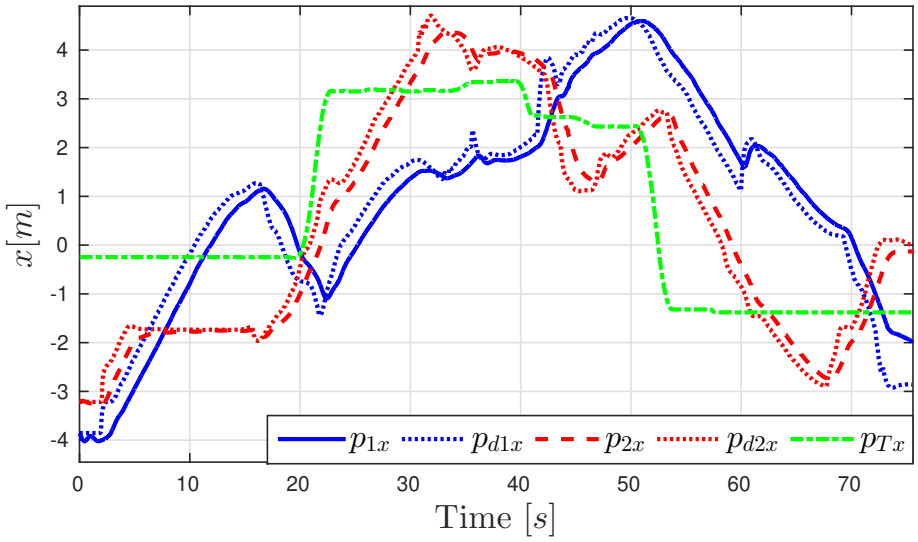


FIGURE 5.30: x axis response when following a moving, the PSO algorithm computes the optimal paths (fine-dotted lines) to reach the target.

The trajectory is computed in the internal microprocessor of each quadrotor using the distributed path planning algorithm from (5.14), which is then used as a reference for the position control law from (5.24) and (5.26).

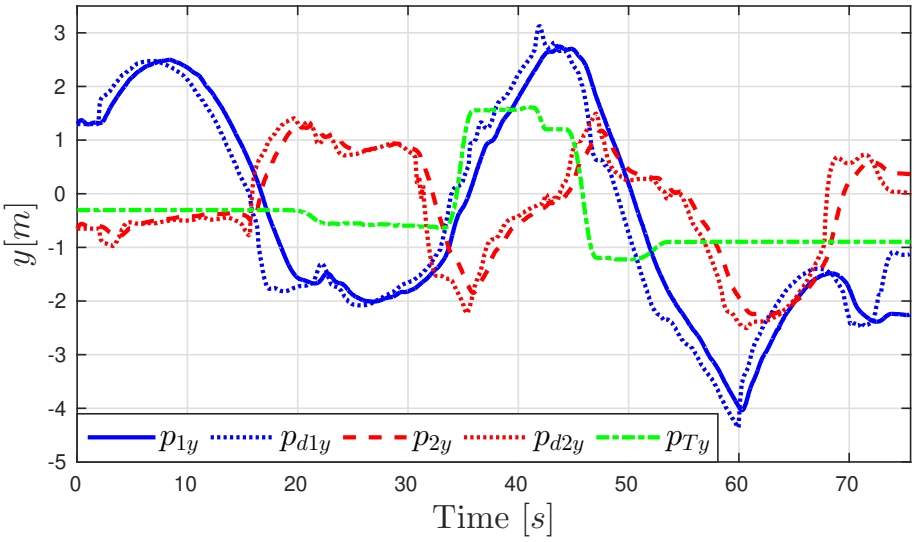


FIGURE 5.31: Translational behavior over the y axis, the proposed controllers ensure that each quadrotor follows the required trajectory to reach the desired distance around the target.

5.2.5.3 Disturbed Agent Scenario

In the last test, the quadrotors were once again set to track a static target. But in this case, one of the drones was highly disturbed by pulling it by hand and placing it on different locations as depicted in Figure 5.32.

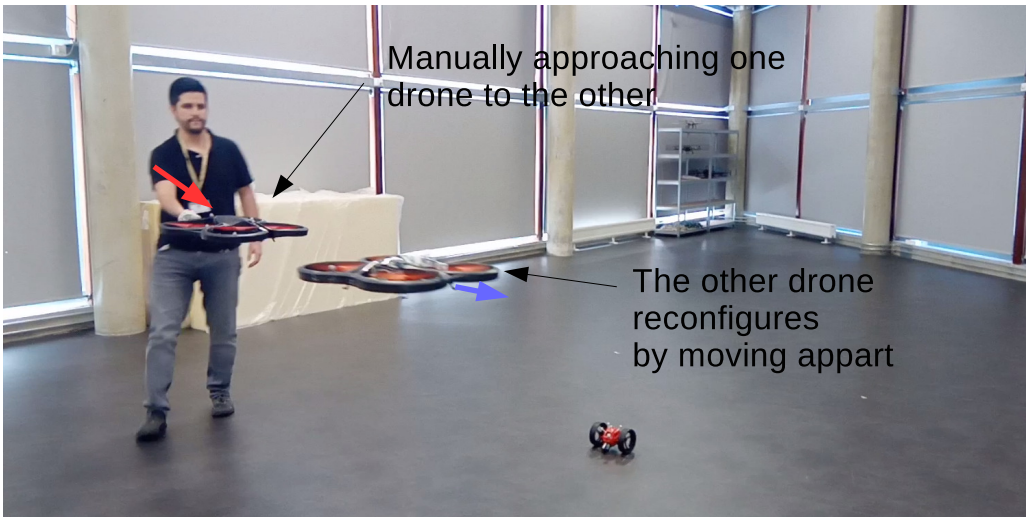


FIGURE 5.32: A drone is disturbed by manually displacing it, the quadrotor drone reconfigures to recover the desired distances while pointing towards the target.

The trajectory generation algorithm using PSO provided robustness to the system by adapting the drones behavior to such changes, and recovering the formation once the

perturbation ends, while the proposed controllers ensured that the force and torque reactions to counteract the disturbances remain bounded. Figure 5.33 illustrates the upper view of the experiment, note when one of the drones is disturbed, the other one reconfigures its position according to the cost function.

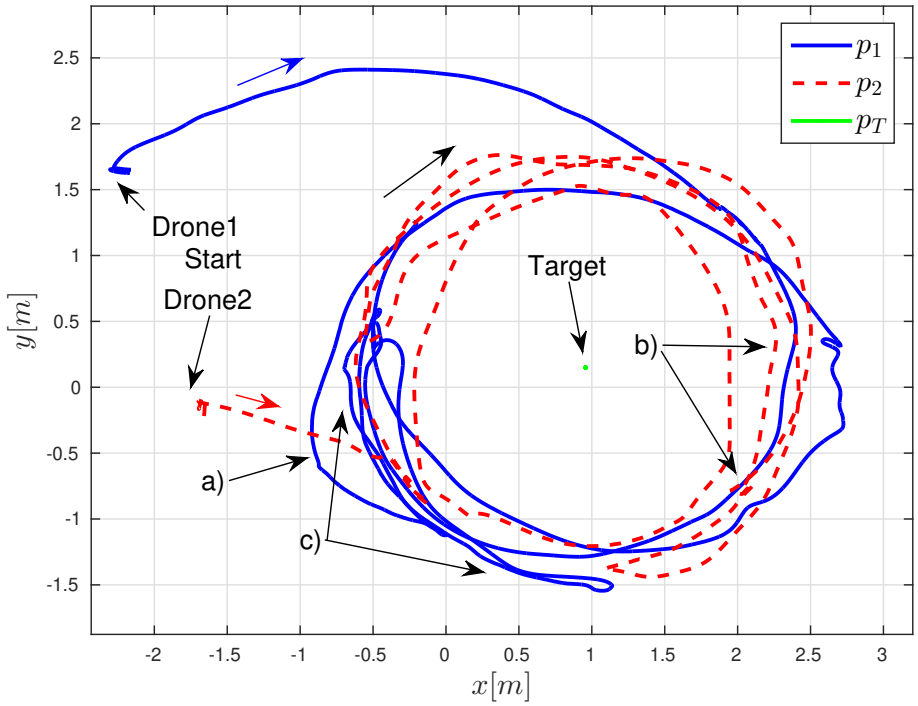


FIGURE 5.33: Disturbed experimet: a) The first drone is moved contrary from its path, b) The second drone is approached to the first one in opposing direction from the circular path, c) The first drone pushed towards the second drone.

The responses of both quadrotors towards strong disturbances are illustrated in Figures 5.34 and 5.35, each disturbance is represented by a letter, which correspond to the ones displayed in Figure 5.33 as follows:

a) Quadrotor 1 was pulled 1 meter contrary to its circular path, the PSO trajectory is still computed in the optimal direction but it fails to be reached because of the disturbance. The reaction of quadrotor 2 is to change its direction in order to maintain the desired distances and avoid coliding.

b) Quadrotor 2 was also pushed 2 meters in the opposite direction of the trajectory, approaching quadrotor 1 which adapts its direction thanks to the PSO agorithm to avoid crashing and to recover the formation.

c) Quadrotor 1 was grabbed and pulled in the same course the circular path for 3 meters, and then stopped, the PSO algorithm then computes a farther reference for quadrotor 2 such that it moves faster to maintain the desired distances.

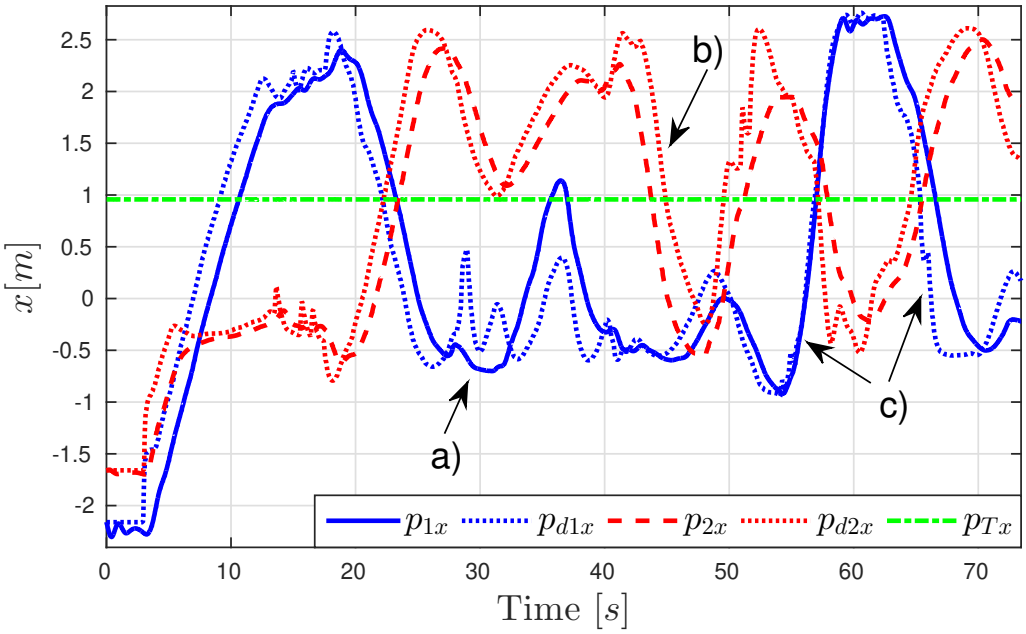


FIGURE 5.34: Response over the x axis for two quadrotors tracking a static UGV in the presence of disturbances, the PSO trajectory is represented with fine-dotted lines.

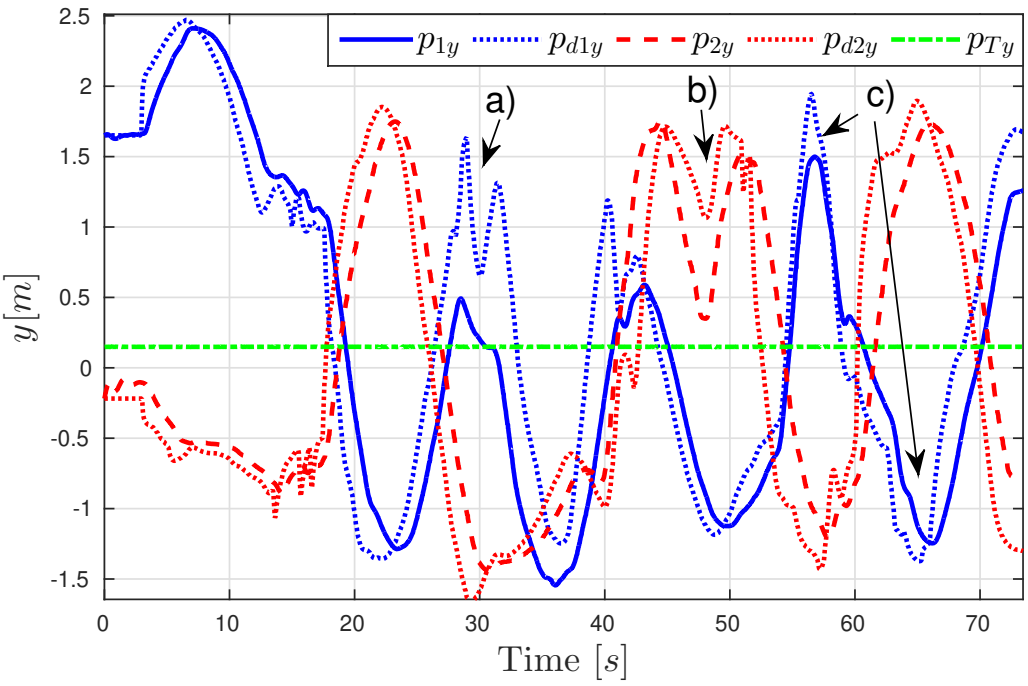


FIGURE 5.35: y axis response in the presence of strong perturbations, the optimization algorithm adapts the trajectory behavior while the proposed controllers ensure a bounded and robust tracking of the computed references.

Finally, Figure 5.36 illustrates the distances between quadrotors and towards the target, the separation from one drone to the other was set at 3 meters, notice that even if disturbances a) and b) reduce this distance the trajectory computed by the PSO algorithm adapts the trajectories to avoid colissions, and recovers to the desired value once the perturbation stops.

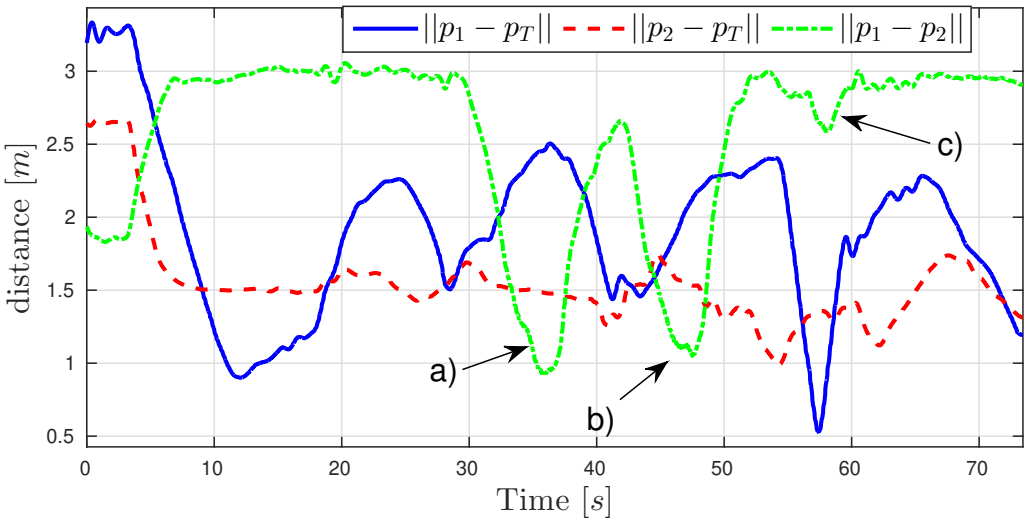


FIGURE 5.36: Distances between quadrotors and with respect to the UGV target, the PSO algorithm along with the bounded controller ensure the formation is recovered after a disturbance occurs.

All the previous experiments can be watched at the following link:
`https://youtu.be/ThisIsADummyLink`

5.3 Autonomous target tracking conclusions

Cooperative navigation between UAVs and UGVs can benefit from the advantages of each kind of robot to counteract their disadvantages in benefit of a common mission. The privileged sight of aerial vehicles, along with their freedom of movement can be used to expand the covered area of the overall system, while the ground robot can be tasked with carrying heavier sensors, and providing a larger time of autonomy for other assignments.

Benefiting from properties of the quaternion-based controllers proposed in previous chapters, which are capable of robustly tracking position and orientation reference in 3-dimensional spaces, two autonomous navigation algorithms were proposed for heterogeneous systems composed by UAVs and UGVs.

Firstly, a set of differential equations are used to design a dynamic trajectory which includes stages for taking-off from a ground vehicle, followed by a tracking phase in which circles are described over the UGV at a desired distance, finalizing with two options for autonomous landing. A quaternion-based feedback controller was employed to track this trajectory with a quadrotor UAV.

Then, a distributed path planning algorithm was introduced in which a set of quadrotors cooperatively track a ground robot target while maintain a symmetrical formation, the construction of the navigation equations are such that once the desired

distances between the UAVs and with respect to the UGV are reached, the drones start describing circles, all while continuously pointing their front-facing axes towards their target. The trajectory generation algorithm was based on an optimization problem which is solved with a PSO approach. In this case, a cylindrical bounded controller was used to ensure that the UAVs follow their trajectories while being robust against unknown disturbances.

Several simulations and real-world tests were performed for both approaches, validating their capabilities for autonomous tracking of ground vehicles using quadtorots

Chapter 6

Conclusions and Future Perspectives

Autonomous robotic systems have been developing in an accelerating pace in recent years, what started as an interesting concept, exclusive to only a few institutions, has become a global trend which is being constantly enhanced by researchers, companies and individuals allover the world.

A common task for aerial vehicles in cooperative or surveillance missions is to follow ground-located targets, in many cases, such targets can be dynamic and perform unpredictable movements, which have to be taken into account by the aerial vehicles' navigation algorithms in order to accurately track them.

In this thesis, autonomous navigation and control strategies for quadrotors tracking ground targets have been developed. The chosen methodologies, which were based on unit quaternions, resulted in algorithms which are capable of following and tracking individual and collective targets, perform robustly towards disturbances, aggressive maneuevers, and even interact with operators in safe and intuitive ways.

On one hand, several quaternion-based control strategies for quadrotors were introduced, ranging from simple feedback linear regulators all the way to geometrical-based bounded algorithms with sliding-mode properties. Since these controllers were developed from quaternion-based representations of the vehicle dynamics, smooth, precise and robust performances were achieved for tracking static and dynamic references.

On the other hand, autonomous navigation algorithms were proposed for single and multiple quadrotors, first, to provide users with a safe and intuitive interface with the system, then, to introduce new ways to deploy and launch aerial vehicles, finally to be able of cooperating and forming fleets that autonomously track moving targets while flying in a coordinated manner.

The proposed controllers and navigation algorithms were validated, first in simulations, then in real-world experiments, displaying precise and robust performances which were partly a consequence from their quaternion-inherited properties.

6.1 Future perspectives

During the development of this thesis, several points that can be improved have been detected, hence the perspectives and future works are numerous

Firstly, this work was mainly focused on developing navigation and control algorithms, but the problem of detecting a target was not addressed, an interesting improvement would be to study vision-based algorithms for detecting objects using down-facing cameras on the quadrotors, such that the proposed algorithms can be tested in more realistic scenarios.

Another point of improvement is the use of different sensors for estimating position signals. For the experiments performed during this thesis, an indoor motion-capture system was used that accurately detects the quadrotors and target positions using infrared cameras, the main disadvantage systems is that their use in outdoors environments can be difficult, expensive, or even impossible in many cases. A solution could be to use GPS signals, which can be improved using filters and observers, or by combining them with visual-inertial techniques for position estimation. The algorithms proposed in this thesis could be tested in outdoors experiments if the signals of these sensors are correctly treated.

Outdoors experiments also represent a challenge for aerial vehicles due to the presence of external disturbances originated by wind gusts. Although the proposed controllers proved to be robust towards unmodeled disturbances, their performance on outdoors environments can be improved by adding observers that estimate wind perturbations, which can then be considered by the controllers by including additional compensating terms.

Regarding the user interaction algorithms, some muscular gestures that can be detected by the employed armband proved to be precise for only a few gestures, while many others were confusing for the system specially when the device was changed from one user to another. An improvement of this work could be to develop algorithms for robustly detecting new muscular gestures, that then can be applied to perform more tasks with the quadrotors. Some primary ideas have been proposed using machine learning algorithms, but further development has to be performed.

Finally, some of the techniques developed during this PhD can be extended to follow flying targets, new cost function can be designed to perform pursuit trajectories, such that their optimal solution yield a distributed cooperative navigation scheme, similar to the one employed in this thesis.

Appendix A

Publications

This PhD work has resulted in the following publications:

A.1 Book Chapters

Ch1 **H. Abaunza**, P. Castillo, and R. Lozano, *Chapter: "Quaternion Modeling and Control Approaches", Book: Handbook of Unmanned Aerial Vehicles, Second Edition*, Kimon P. Valavanis and George J. Vachtsevanos, Springer, 2019.

Ch2 **H. Abaunza**, J. Cariño, and P. Castillo, *Chapter 2 "Modeling approaches", Book: Indoor navigation strategies for aerial autonomous systems*, P. Castillo, L. E. Munoz, P. Garcia, ISBN: 780128051894, Fev. 2017, Elsevier.

A.2 International Journals

J1 **H. Abaunza**, P. Castillo, *Quatrotor aggressive deployment, using a quaternion-based spherical chattering-free sliding-mode controller*, IEEE Transactions on Aerospace and Electronic Systems (IEEE TAES). *submitted in 2019

J2 E. Ibarra, P. Castillo, and **H. Abaunza**, *Nonlinear control with integral sliding properties for circular aerial robot trajectory tracking: real-time validation*, International Journal of Robust and Nonlinear Control (IJRNC), Wiley Online Library, *submitted in 2019.

J3 L.F. Sanchez, **H. Abaunza**, and P. Castillo, *User-Robot Interaction For Safe Navigation of a Quadrotor*, Robotica, Cambridge University Press, *submitted in 2019.

J4 A. Belkadi, **H. Abaunza**, L. Ciarletta, P. Castillo, and D. Theilliol, *Design And Implementation of Distributed Path Planning Algorithm for a Fleet Of UAVs*, IEEE Transactions on Aerospace and Electronic Systems (IEEE TAES), 2019.

J5 **H. Abaunza**, P. Castillo, A. Victorino, and R. Lozano, *Dual Quaternion Modeling and Control of a Quad-rotor Aerial Manipulator*, Journal of Intelligent & Robotic Systems, 2017.

J6 M.E. Guerrero, **H. Abaunza**, P. Castillo, R. Lozano, and C.D. García, *Quadrotor Energy-Based Control Laws: A Unit-Quaternion Approach*, Journal of Intelligent & Robotic Systems, 2017.

J7 M.E. Guerrero, **H. Abaunza**, P. Castillo, and R. Lozano, *Passivity-Based Control for a Micro Air Vehicle using Unit Quaternions*, Applied Sciences - Open Access Journal, December 2016.

A.3 National Journals

NJ1 **H. Abaunza**, and P. Castillo, *Les Applications de Drones Aériens - L'utilisation civile des UAVs*, Techniques de l'Ingenieur, Editions T.I., Août 2017, Ref s7816.

A.4 International Conferences

C1 **H. Abaunza**, P. Castillo, D. Theilliol, A. Belkadi, and L. Ciarletta *Cylindrical Bounded Quaternion Control for Coordinated Circular Target Tracking on UAVs*, IEEE Conference on Decision and Control (IEEE CDC). *submitted in 2019

C2 B. Rojas, **H. Abaunza**, P. Castillo, and I. Thouvenin, *Bilateral teleoperation of a drone using a virtual environment*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), *submitted in 2019.

C3 M. Hartmann, **H. Abaunza**, P. Castillo, M. Stolz, and D. Watzening *Pedestrian in the Loop: An approach using flying drones*, IEEE International Instrumentation & Measurement Technology Conference, Houston, USA, 2018.

C4 **H. Abaunza**, E. Ibarra, P. Castillo, and A. Victorino, *Quaternion based control for circular UAV trajectory tracking, following a ground vehicle: Real-time validation*, 20th IFAC World Congress, Toulouse, France, 2017.

C5 A. Belkadi, **H. Abaunza**, L. Ciarletta, P. Castillo, and D. Theilliol, *Distributed Path Planning for Controlling a Fleet of UAVs : Application to a Team of Quadrotors*, 20th IFAC World Congress, Toulouse, France, 2017.

C6 L.F. Sanchez, **H. Abaunza**, and P. Castillo, *Safe Navigation Control for a Quadcopter using User's arm commands*, International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 2017.

C7 M.E. Guerrero, **H. Abaunza**, P. Castillo, R. Lozano, and C.D. García, *Energy Based Control for a Quadrotor using Unit Quaternions*, International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 2016.

C8 **H. Abaunza**, P. Castillo, R. Lozano, and A. Victorino, *Quadrotor Aerial Manipulator based on Dual Quaternions*, International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 2016.

C9 **H. Abaunza**, J. Cariño, P. Castillo, R. Lozano, *Quadrotor Dual Quaternion Control*, In the Workshop on Research, Education and Development of Unmanned Aerial Systems (REDUAS), Cancún, México 2015.

C10 J. Cariño, **H. Abaunza**, and P. Castillo, *Quadrotor Quaternion Control*, International Conference on Unmanned Aircraft Systems (ICUAS), Denver, USA, 2015.

Bibliography

[1] KC Wong. Aerospace industry opportunities in australia-unmanned aerial vehicles (uavs)-are they ready this time? are we? *Department of Aeronautical Engineering, University of Sydney, November*, 1997.

[2] Kimon P Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*, volume 33. Springer Science & Business Media, 2008.

[3] Jan Roskam. *Airplane flight dynamics and automatic flight controls*. DARcorporation, 1998.

[4] Andra Saicharan Sagar, PD Shendge, and SM Vaitheeswaran. Attitude control of fixed wing uav using multiple sliding surface. In *Power Electronics, Intelligent Control and Energy Systems (ICPEICES), IEEE International Conference on*, pages 1–6. IEEE, 2016.

[5] Amar Benghezal, Rabah Louali, Abedelouahab Bazoula, and Taha Chettibi. Trajectory generation for a fixed-wing uav by the potential field method. In *Control, Engineering & Information Technology (CEIT), 2015 3rd International Conference on*, pages 1–6. IEEE, 2015.

[6] R Hugh Stone. Control architecture for a tail-sitter unmanned air vehicle. In *Control Conference, 2004. 5th Asian*, volume 2, pages 736–744. IEEE, 2004.

[7] R Hugh Stone, Peter Anderson, Colin Hutchison, Allen Tsai, Peter Gibbens, and KC Wong. Flight testing of the t-wing tail-sitter unmanned air vehicle. *Journal of Aircraft*, 45(2):673–685, 2008.

[8] John Hauser, Shankar Sastry, and George Meyer. Nonlinear control design for slightly non-minimum phase systems: Application to v/stol aircraft. *Automatica*, 28(4):665–679, 1992.

[9] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 3. Addison-Wesley Reading, MA, 1994.

[10] Byoung S Kim and Anthony J Calise. Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics*, 20(1):26–33, 1997.

[11] Tarek Hamel, Robert Mahony, Rogelio Lozano, and James Ostrowski. Dynamic modelling and configuration stabilization for an x4-flyer. In *IFAC 15th World Congress on Automatic Control, Barcelona, Spain*, 2002.

[12] Erdinc Altug, James P Ostrowski, and Robert Mahony. Control of a quadrotor helicopter using visual feedback. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 72–77. IEEE, 2002.

[13] Erdinc Altug, James P Ostrowski, and Camillo J Taylor. Quadrotor control using dual camera visual feedback. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 4294–4299. IEEE, 2003.

[14] Samir Bouabdallah, Andre Noth, and Roland Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2451–2456. IEEE, 2004.

[15] Chien-Shiun Huang and King Yuan. Output tracking of a non-linear non-minimum phase pvtol aircraft based on non-linear state feedback control. *International Journal of Control*, 75(6):466–473, 2002.

[16] Rogelio Lozano, Pedro Castillo, and Alejandro Dzul. Global stabilization of the pvtol: real-time application to a mini-aircraft. *IFAC Proceedings Volumes*, 37(21):235–240, 2004.

[17] Arturo Zavala-Rio, Isabelle Fantoni, and Rogelio Lozano. Global stabilization of a pvtol aircraft model with bounded inputs. *International Journal of Control*, 76(18):1833–1844, 2003.

[18] Anand Sánchez, Isabelle Fantoni, Rogelio Lozano, and Jesús De León Morales. Observer-based control of a pvtol aircraft. *IFAC Proceedings Volumes*, 38(1):121–126, 2005.

[19] Ahmed Chemori and Nicolas Marchand. Global discrete-time stabilization of the pvtol aircraft based on fast predictive control. *IFAC Proceedings Volumes*, 41(2):1747–1752, 2008.

[20] Ahmed Chemori and Nicolas Marchand. A prediction-based nonlinear controller for stabilization of a non-minimum phase pvtol aircraft. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(8):876–889, 2008.

[21] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. Stabilization of a mini-rotorcraft having four rotors. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2693–2698. IEEE, 2004.

[22] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. Stabilization of a mini rotorcraft with four rotors. *IEEE control systems*, 25(6):45–55, 2005.

[23] Samir Bouabdallah and Roland Y Siegwart. Full control of a quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007: IROS 2007; Oct. 29, 2007-Nov. 2, 2007, San Diego, CA*, pages 153–158. Ieee, 2007.

[24] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6461, 2007.

[25] Tarek Madani and Abdelaziz Benallegue. Backstepping control for a quadrotor helicopter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260. IEEE, 2006.

[26] Bora Erginer and Erdinc Altug. Modeling and pd control of a quadrotor vtol vehicle. In *2007 IEEE Intelligent Vehicles Symposium*, pages 894–899. IEEE, 2007.

[27] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a quad-rotor robot. In *Proceedings Australasian Conference on Robotics and Automation 2006*. Australian Robotics and Automation Association Inc., 2006.

[28] Tarek Madani and Abdelaziz Benallegue. Control of a quadrotor mini-helicopter via full state backstepping technique. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1515–1520. IEEE, 2006.

[29] Saif A Al-Hiddabi. Quadrotor control using feedback linearization with dynamic extension. In *2009 6th International Symposium on Mechatronics and its Applications*, pages 1–3. IEEE, 2009.

[30] David W Kun and Inseok Hwang. Linear matrix inequality-based nonlinear adaptive robust control of quadrotor. *Journal of Guidance, Control, and Dynamics*, pages 996–1008, 2015.

[31] Rong Xu and Umit Ozguner. Sliding mode control of a quadrotor helicopter. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 4957–4962. IEEE, 2006.

[32] Damien Galzi and Yuri Shtessel. Unmanned rotorcraft tight formation flight control using sliding mode control driven by sliding mode disturbance observers. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7171, 2008.

[33] Steven Lake Waslander, Gabriel M Hoffmann, Jung Soon Jang, and Claire J Tomlin. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3712–3717. IEEE, 2005.

[34] Alexander Joos, Andy Haefner, Florian Weimer, and Walter Fichter. Quadrocopter ground effect compensation with sliding mode control. In *AIAA Guidance, Navigation, and Control Conference*, page 7871, 2010.

[35] P Castillo, Pedro Albertos, Pedro Garcia, and Rogelio Lozano. Simple real-time attitude stabilization of a quad-rotor aircraft with bounded signals. In *Decision and Control, 2006 45th IEEE Conference on*, pages 1533–1538. IEEE, 2006.

[36] A Alatorre, Pedro Castillo, and S Mondie. Improving the performance of aerial vehicles with delayed measures via state predictor-control: Experimental data validation. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 910–919. IEEE, 2014.

[37] Benoit Landry, Robin Deits, Peter R Florence, and Russ Tedrake. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1469–1475. IEEE, 2016.

[38] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2017.

[39] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5774–5781. IEEE, 2017.

[40] Syed Ali Raza and Jason Etele. Autonomous position control analysis of quadrotor flight in urban wind gust conditions. In *AIAA Guidance, Navigation, and Control Conference*, page 1385, 2016.

[41] Syed Ali Raza, Jason Etele, and Giovanni Fusina. Hybrid controller for improved position control of quadrotors in urban wind conditions. *Journal of Aircraft*, 55(3):1014–1023, 2017.

[42] S.L. Altmann. Hamilton, rodrigues, and the quaternion scandal. *Mathematics Magazine*, pages 291–308, 1989.

[43] K.W. Spring. Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review. *Mechanism and machine theory*, 21(5):365–373, 1986.

[44] Jack CK Chou. Quaternion kinematic and dynamic differential equations. *IEEE Transactions on robotics and automation*, 8(1):53–64, 1992.

[45] J.B. Kuipers. *Quaternions and rotation sequences*, volume 66. Princeton university press Princeton, 1999.

[46] BP Ickes. A new method for performing digital control system attitude computations using quaternions. *AIAA journal*, 8(1):13–17, 1970.

[47] Bong Wie and Peter M Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[48] Bong Wie, H Weiss, and A Arapostathis. Quaternion feedback regulator for spacecraft eigenaxis rotations. *Journal of Guidance, Control, and Dynamics*, 12(3):375–380, 1989.

[49] Roberto Cristi, Jeffrey Burl, and Nick Russo. Adaptive quaternion feedback regulation for eigenaxis rotations. *Journal of Guidance, Control, and Dynamics*, 17(6):1287–1291, 1994.

[50] Shih-Che Lo and Yon-Ping Chen. Smooth sliding-mode control for spacecraft attitude tracking maneuvers. *Journal of Guidance, Control, and Dynamics*, 18(6):1345–1349, 1995.

[51] John L Crassidis and F Landis Markley. Sliding mode control using modified rodrigues parameters. *Journal of Guidance, Control, and Dynamics*, 19(6):1381–1383, 1996.

[52] A Tayebi and S McGilvray. Attitude stabilization of a four-rotor aerial robot. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1216–1221. IEEE, 2004.

[53] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on control systems technology*, 14(3):562–571, 2006.

[54] Cédric Berbra, Suzanne Lesecq, Sylviane Gentil, and J-M Thiriet. Co-design of a safe network control quadrotor. *IFAC Proceedings Volumes*, 41(2):5506–5511, 2008.

[55] Emanuel Stingu and Frank Lewis. Design and implementation of a structured flight controller for a 6dof quadrotor using quaternions. In *Control and Automation, 2009. MED'09. 17th Mediterranean Conference on*, pages 1233–1238. IEEE, 2009.

[56] José-Fermi Guerrero-Castellanos, Ahmad Hably, Nicolas Marchand, and Suzanne Lesecq. Bounded attitude stabilization: Application on four-rotor helicopter. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 730–735. IEEE, 2007.

[57] JF Guerrero-Castellanos, Nicolas Marchand, Suzanne Lesecq, and Jérôme Delamare. Bounded attitude stabilization: Real-time application on four-rotor mini-helicopter. *IFAC Proceedings Volumes*, 41(2):3167–3173, 2008.

[58] JF Guerrero-Castellanos, Nicolas Marchand, Ahmad Hably, Suzanne Lesecq, and Jérôme Delamare. Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. *Control Engineering Practice*, 19(8):790–797, 2011.

[59] Chutiphon Pukdeboon, Alan SI Zinober, and May-Win L Thein. Quasi-continuous higher order sliding-mode controllers for spacecraft-attitude-tracking maneuvers. *IEEE Transactions on Industrial Electronics*, 57(4):1436–1444, 2010.

[60] Zheng Zhu, Yuanqing Xia, Mengyin Fu, et al. Adaptive sliding mode control for attitude stabilization with actuator saturation. *IEEE Transactions on Industrial Electronics*, 58(10):4898–4907, 2011.

[61] Anezka Chovancová, Tomas Fico, Peter Hubinskỳ, and F Duchoň. Comparison of various quaternion-based control methods applied to quadrotor with disturbance observer and position estimator. *Robotics and Autonomous Systems*, 79:87–98, 2016.

[62] Emil Fresk and George Nikolakopoulos. Full quaternion based attitude control for a quadrotor. In *Control Conference (ECC), 2013 European*, pages 3864–3869. IEEE, 2013.

[63] Moses Bangura, Robert Mahony, Hyon Lim, Hee Jin Kim, et al. An open-source implementation of a unit quaternion based attitude and trajectory tracking for quadrotors. 2014.

[64] Bin Xian, Chen Diao, Bo Zhao, and Yao Zhang. Nonlinear robust output feedback tracking control of a quadrotor uav using quaternion representation. *Nonlinear Dynamics*, 79(4):2735–2752, 2015.

[65] C Izaguirre-Espinosa, AJ Muñoz-Vázquez, A Sánchez-Orta, V Parra-Vega, and G Sanahuja. Fractional attitude-reactive control for robust quadrotor position stabilization without resolving underactuation. *Control Engineering Practice*, 53:47–56, 2016.

[66] Carlos Izaguirre-Espinosa, Aldo Jonathan Muñoz-Vázquez, Anand Sánchez-Orta, Vicente Parra-Vega, and Pedro Castillo. Attitude control of quadrotors based on fractional sliding modes: theory and experiments. *IET Control Theory & Applications*, 10(7):825–832, 2016.

[67] Yonggen Ling, Tianbo Liu, and Shaojie Shen. Aggressive quadrotor flight using dense visual-inertial fusion. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1499–1506. IEEE, 2016.

[68] Ick Wang, Vladimir Dobrokhodov, Isaac Kaminer, and Kevin Jones. On vision-based target tracking and range estimation for small uavs. In *AIAA guidance, navigation, and control conference and exhibit*, page 6401, 2005.

[69] Richard Wise and Rolf Rysdyk. Uav coordination for autonomous target tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6453, 2006.

[70] Herbert G Tanner. Switched uav-ugv cooperation scheme for target detection. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3457–3462. IEEE, 2007.

[71] Jongrae Kim and Yoonsoo Kim. Moving ground target tracking in dense obstacle areas using uavs. *IFAC Proceedings Volumes*, 41(2):8552–8557, 2008.

[72] Steven AP Quintero, Francesco Papi, Daniel J Klein, Luigi Chisci, and Joao P Hespanha. Optimal uav coordination for target tracking using dynamic programming. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4541–4546. IEEE, 2010.

[73] Celine Teuliere, Laurent Eck, and Eric Marchand. Chasing a moving target from a flying uav. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4929–4934. IEEE, 2011.

[74] Huili Yu, Kevin Meier, Matthew Argyle, and Randal W Beard. Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles. *IEEE/ASME Transactions on Mechatronics*, 20(2):541–552, 2015.

[75] Jae-Keun Lee, Hahmin Jung, Huosheng Hu, and Dong Hun Kim. Collaborative control of uav/ugv. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2014 11th International Conference on*, pages 641–645. IEEE, 2014.

[76] Tiago Oliveira, A Pedro Aguiar, and Pedro Encarnacao. Moving path following for unmanned aerial vehicles with applications to single and multiple target tracking problems. *IEEE Transactions on Robotics*, 32(5):1062–1078, 2016.

[77] Peng Yao, Honglun Wang, and Zikang Su. Cooperative path planning with applications to target tracking and obstacle avoidance for multi-uavs. *Aerospace Science and Technology*, 54:10–22, 2016.

[78] Alexandre Borowczyk, Duc-Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous landing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40(9):2378–2385, 2017.

[79] Qi Lu, Beibei Ren, and Siva Parameswaran. Shipboard landing control enabled by an uncertainty and disturbance estimator. *Journal of Guidance, Control, and Dynamics*, 41(7):1502–1520, 2018.

[80] H. Goldstein. *Classical mechanics*, volume 4. Pearson Education India, 1962.

[81] P. Pounds, R. Mahony, J. Gresham, P Corke, and J. Roberts. Towards dynamically-favourable quad-rotor aerial robots. 2004.

[82] R. W. Prouty. *Helicopter performance, stability and control.* Krieger Publishing Company, 2002.

[83] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 2010.

[84] Pedro Castillo, Rogelio Lozano, and Alejandro E Dzul. *Modelling and Control of Mini-Flying Machines*, volume 1. Springer Science & Business Media, 2005.

[85] Bernard Etkin and Lloyd Duff Reid. *Dynamics of flight: stability and control*, volume 3. Wiley New York, 1996.

[86] Rogelio Lozano. *Unmanned aerial vehicles: Embedded control.* John Wiley & Sons, 2013.

[87] M.A. Clifford. Preliminary sketch of biquaernions. *Proceedings of the London Mathematical Society*, 4, 1873.

[88] A. Buchheim. A memoir on biquaternions. *American Journal of Mathematics*, 7:293–326, 1885.

[89] A.P. Kotelnikov. Screw calculus and some applications to geometry and mechanics. *Annuals of Imperial University of Kazan*, 1895.

[90] Romeo Ortega, Julio Antonio Loría Perez, Per Johan Nicklasson, and Hebertt J Sira-Ramirez. *Passivity-based control of Euler-Lagrange systems: mechanical, electrical and electromechanical applications.* Springer Science & Business Media, 2013.

[91] Cristian Souza, Guilherme Vianna Raffo, and Eugenio B Castelan. Passivity based control of a quadrotor uav. *IFAC Proceedings Volumes*, 47(3):3196–3201, 2014.

[92] Burak Yüksel, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. Reshaping the physical properties of a quadrotor through ida-pbc and its application to aerial physical interaction. In *2014 IEEE Int. Conf. on Robotics and Automation*, page 8p, 2014.

[93] Romeo Ortega and Iven Mareels. Energy-balancing passivity-based control. In *American Control Conference, 2000. Proceedings of the 2000*, volume 2, pages 1265–1270. IEEE, 2000.

[94] Fatima Oliva-Palomo, Anand Sanchez-Orta, Pedro Castillo, and Hussain Alazki. Nonlinear ellipsoid based attitude control for aggressive trajectories in a quadrotor: Closed-loop multi-flips implementation. *Control Engineering Practice*, 77:150–161, 2018.

[95] Guillaume Sanahuja and et al. Fl-air framework libre air. `https://uav.hds.utc.fr/software-flair/`. Accessed: 2018-10-13.

[96] Abdel-Razzak Merheb, Veysel Gazi, and Nilay Sezer-Uzol. Implementation studies of robot swarm navigation using potential functions and panel methods. *IEEE/ASME Trans. Mechatronics*, 21(5):2556–2567, 2016.

[97] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[98] Julio Betancourt-Vera, Pedro Castillo, Rogelio Lozano, and Boris Vidolov. Task assignment/trajectory planning for unmanned vehicles via hflc and pso. In *Unmanned Aircraft Systems (ICUAS), 2018 International Conference on*. IEEE, 2018.

[99] Dominic Jordan, Peter Smith, and Peter Smith. *Nonlinear ordinary differential equations: an introduction for scientists and engineers*, volume 10. Oxford University Press on Demand, 2007.

[100] Adel Belkadi, Hernan Abaunza, Laurent Ciarletta, Pedro Castillo, and Didier Theilliol. Distributed path planning for controlling a fleet of uavs: application to a team of quadrotors. *IFAC-PapersOnLine*, 50(1):15983–15989, 2017.

[101] Adel Belkadi, Laurent Ciarletta, and Didier Theilliol. Uavs fleet control design using distributed particle swarm optimization: A leaderless approach. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 364–371. IEEE, 2016.

[102] Adel Belkadi, Hernan Abaunza, Laurent Ciarletta, Pedro Castillo, and Didier Theilliol. Design and implementation of distributed path planning algorithm for a fleet of uavs. *IEEE-Transactions on Aerospace and Electronic Systems*.