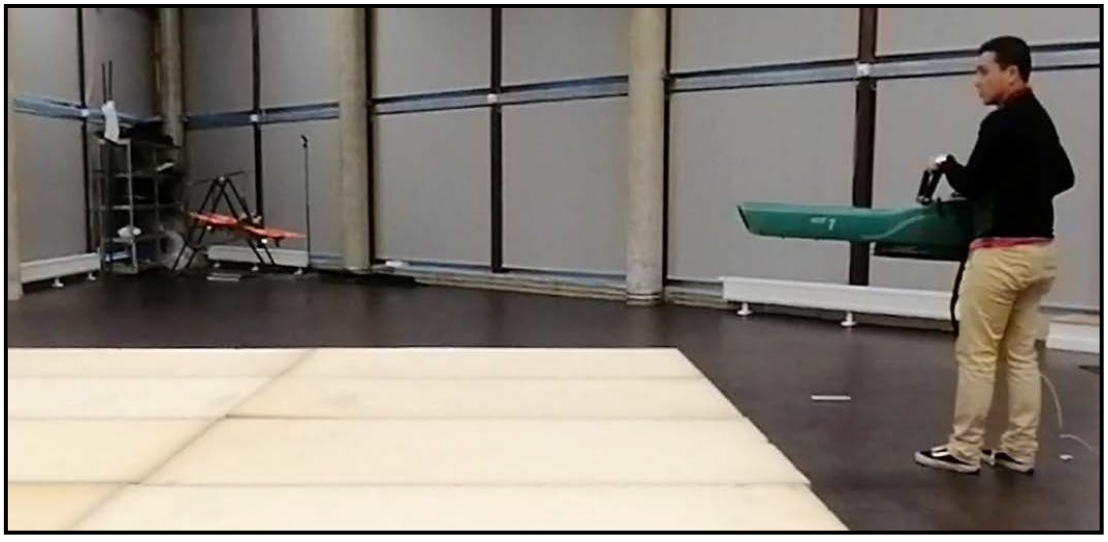
Autonomous robust control techniques for dynamic target tracking

Guillermo Julio Cesar Betancourt Vera

Par **Guillermo Julio Cesar BETANCOURT VERA**

Autonomous robust control techniques for dynamic target tracking

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



Soutenue le 28 juin 2021
Spécialité : Automatique et Robotique : Unité de recherche Heudyasic (UMR-7253)

D2619

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

HEUDIASYC, UMR 7253

AUTONOMOUS ROBUST CONTROL TECHNIQUES FOR DYNAMIC TARGET TRACKING

Thesis presented by

Guillermo Julio Cesar Betancourt Vera

to obtain the degree of Doctor

28/06/2021

Spécialité : Automatique et Robotique

Committee members:

Referees:	Sihem Tebbani	Professor at L2S
	Nicolas Marchand	DR CNRS at GIPSA-Lab
Examiners:	Reine Talj	HDR CNRS at HDS
	Didier Theilliol	Professor at CRAN
	Franck Ruffier	DR CNRS at ISM
	Lounis Adouane	Professor at HDS
Supervisors:	Pedro Castillo	HDR CNRS at HDS
	Rogelio Lozano	DR CNRS at HDS

Compiègne, France 2021

"Once you have tasted flight, you will forever walk the earth
with your eyes turned skyward, for there you have been,
and there you will always long to return."
— Leonardo da Vinci

To my lovely family...

Acknowledgements

I would like to deeply thank Pedro Castillo and Rogelio Lozano for giving me the opportunity to pursue my PhD under their supervision on an interesting interdisciplinary topic that included robots, dynamics, control, etc. This thesis would not have been possible without their continuous support, patience, advise, and availability. I can hardly express my gratitude in words to Pedro and Rogelio for letting me work in such an amazing lab that has been unique in many aspects. The availability of several advanced robots and equipments in our lab gave me an exceptional opportunity to work with many state-of-art platforms and to evaluate different aspects of my work in real world experiments. Thanks to them for all the time their spent to read and correct my articles and thesis and all their fruitful guidelines, without which this thesis would never have reached its current state.

I want to express also my gratitude to the institutions that supported this work from the beginning. To the Mexican *Consejo Nacional de Ciencia y Tecnlogía (CONACYT)* for the scholarship program that supported me during my PhD studies in France. A big thank you to the *Heudiasyc (Heuristic et DIAgnostique des SYstèmes Complexes)* laboratory and the *Ècole Doctorale Sciences pour l'Ingenieur at the Université Technologie de Compiègne (UTC)* that made this work possible with their PhD programs.

I would like to thank my examiners Professor Sihem Tebbani and the DR CNRS Nicolas Marchand that have honored me by accepting to read my thesis. Your insightful comments were undoubtedly a great asset for this thesis. My thanks are also for the HDR CNRS Reine Talj, Professors Didier Theilliol and Lounis Adouane as well the HDR CNRS Franck Ruffier for accepting to be part of my jury and giving very helpful commentaries of my thesis.

I have enjoyed each moment of my PhD life thanks to the amazing atmosphere at our lab. Every working day was full of interesting scientific and non-scientific discussions with such an awesome multifaceted group of people. I would like to immensely thank Guillaume Sanahuja, Gildas Bayard, Thierry Monglon, Alexis Offermann, Cristino de Souza, Hernan Abaunza and Belem Rojas with whom I had the opportunity to discuss, life, music, tv, science, games, religion, and many other interesting topics. A big thanks specially to Guillaume and Gildas for FL-AIR, an invaluable framework that helped me a lot to apply my methods on real and simulated robots. Their passion for robots and technology, their sense of humor and this endless knowledge on C++ have been an inspiration and a pleasure for me.

Thanks to Professor Pedro Gil and Vicente Balaguer for your support, knowledge and the great scientific discussions we had during our meetings. I hope to continue collaborating with you.

Acknowledgements

I would like to thank Sabine Vidal, Brigitte Azzimonti and Nathalie Alexandre from the administrative service of the laboratory and Sylvie Carlier from the administrative service of the *Ècole Doctorale Sciences pour l'Ingenieur at the Université Technologie de Compiègne (UTC)*, for taking care of the administrative aspects of my PhD.

I would like to thank a number of valuable friends outside my doctoral study, who has made living in France a very enjoyable and pleasant experience for me. My especial thanks to Nicolas Abdelnour, Louise le Masne, Maxime Juston, Garance Verrier, Aurélien Quelin, Laure Tangre, Hénoïk Willot, Santiago Venegas and Emeric Ostermeyer with whom I shared many great memories. Life in Compiègne could not be better thanks to your companionship, availability and sens of humor. I would also like to thank my roommates and friends Alexis Pagnoux, Alexis Nauton, Mokossia N'guessan, Quentin Duchemin, Maxime Beaugendre, Lola Lejeune, Matěj Křístek, Pierre Kidzié and Anna Tailliez for all the unforgettable moments that we spent together at home. A huge thanks for the amazing conversations that I had with each one of you, for teaching me the french culture, for being so patience when I was learning french and for the unforgettable and uncountable hours of music and sport sessions. All my gratitude to all of you. I could say that I found my second family with all of you.

My especial thanks to Ricardo Sanz for always proposing me such amazing hikes. I passed a wonderful time walking and discussing about everything with you. You made mountains became really important to me and that gave me peace and energy. My thanks to Oscar Beltran, my cheerful and supportive friend with whom I shared many ups and downs of PhD life. Thanks a lot for always believing on me and for being such a good friend when I needed it. Thanks to Jesus Solis, for the unmemorable discussions about science, technology and life. Even with a huge distance and seven hours of time difference, you made possible to helped me with my Linux issues. You are truly good friends that I can see as my brothers.

Last, but far from least, I would have never reached the place where I am now without the endless support, love and encouragement of my parents, Abraham and Alma. I am immensely thankful for their confidence in me, which has allowed me to discover many aspects of life independently. I would also like to deeply thank to my brothers Jesús, Abraham and Juan for all their back up and the nice moments we have spent together.

Compiègne, June 28, 2021 J. B.

Abstract

This thesis will approach the problem of tracking moving targets by means of an aerial drone. More precisely, 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.

The goal of this thesis is to develop control and navigation algorithms capable of robustly tracking dynamic targets. The contributions herein reported aim at presenting a unified and accessible analysis of control strategies for solving the stabilization and tracking problems. In addition, this thesis introduces autonomous and semi-autonomous navigation algorithms for tracking static and dynamic targets. Finally, for giving robustness with respect to a class of perturbations, a robust control architecture based on disturbance observer has been developed.

Control approaches for aerial vehicles: The first part of this work consisted on developing quadrotor controllers with the aim of robustly tracking trajectories and performing precise navigation tasks. It is well-known that, stabilizing and tracking control problems of a quadrotor vehicle has been studied through the last decades. Nevertheless,choosing the best control technique to solve these problems is still not an easy task. Therefore, the following control strategies were introduced:

– Nonlinear and linear backstepping.
– Simple nonlinear bounded control.
– Virtual control based on hyperbolic functions.
– Quaternion-based backstepping control.
– Finite time convergence based on sliding mode control.

Autonomous navigation algorithms: The next part of this thesis was focused on developing semi-autonomous and autonomous algorithms that allows to detect and track a mobile target, as well to generate desired trajectories such that, the controllers previously developed can track.

A path planning algorithm for quadcopters based on model predictive control was developed for the autonomous trajectory generation. The algorithm was solved in real time as a result of an online optimization of a cost function. Then, the optimal trajectories were tracked using a simple nonlinear bounded control.

Then, concerning the detection and tracking of a ground mobile target, an autonomous visual servoing navigation algorithm was proposed. The algorithm considers that the ground vehicle mimics a car driver such that its movements can be described as a vector field. In addition the ground robot is detected using properties from the well-known optical flow algorithm. Thus, the quadrotor tracks the ground vehicle using the quaternion-based backstepping control.

Lastly, a semi-autonomous virtual control architecture for controlling robots remotely was developed in cooperation with the team of virtual reality at the Heudiasyc Lab. The architecture is composed by a world-in-miniature located in a virtual environment and, a real aerial robot located in a fly arena. The real aerial robot tracks the desired positions and velocities, coming from the virtual environment, using a quaternion-based backstepping control.

Robust control scheme: The last part of this thesis was dedicated to the conception of a robust control architecture capable to counter-acted model uncertainties and a external perturbations such as wind gust and loss of effectiveness in rotors. This architecture can be seen as follows:

A robust control architecture based on disturbance estimator was introduced to give robustness to the closed-loop system with respect to nonlinear uncertainties and small external disturbances. The architecture performed experimental tests enhancing a simple feedback controller.

Then, improving the previously discussed architecture, a novel robust fault tolerant control scheme was obtained. The scheme has in addition a rotors fault observer to detect and estimate the loss of effectiveness in actuators. The robust controller can estimate and counteracted rotors faults in non hover and hover positions.

Ultimately, a bounded robust control architecture was developed from the disturbance observer and optimal control properties. Besides estimating and compensating unknown dynamics, this scheme improves its robustness by solving the physical constraints of the motors. The architecture was experimentally proved with several wind gust and loss of effectiveness in two rotors.

Résumé

Cette thèse étudie la problématique du suivi de cibles mobiles à l'aide d'un drone aérien. Plus précisément, une configuration quadrotor a été utilisée pour le développement d'algorithmes de navigation aérienne. En effet, ce type de plateforme est simple mécaniquement, polyvalent pour effectuer des manœuvres aggressives, et facilement disponible pour l'expérimentation.

L'objectif de cette thèse est de développer des algorithmes de contrôle et de navigation capables de suivre de manière robuste une cible dynamique. Les contributions rapportées ici présent une analyse unifiée et accessible des stratégies de contrôle pour résoudre les problèmes de stabilisation et de suivi. Cette thèse introduit également des algorithmes de navigation autonome et semi-autonome pour le suivi d'une cible statique et dynamique. Enfin, pour donner de la robustesse vis-à-vis d'une classe de perturbations, une architecture de contrôle robuste basée sur l'observateur de perturbations a été développée.

Techniques de contrôle des véhicules aériens : La première partie de ce travail a consisté à développer des contrôleurs de quadrotor dans le but de suivre de manière robuste des trajectoires et de réaliser des tâches de navigation précises. En effet, les problèmes de contrôle de stabilisation et de suivi d'un véhicule quadrotor ont été étudiés au cours des dernières décennies. Néanmoins, choisir la meilleure technique de contrôle pour résoudre ces problèmes n'est pas une tâche facile. Par conséquent, les stratégies de contrôle suivantes ont été introduites :

– Backstepping linéaires et non linéaires.
– Contrôle borné non linéaire.
– Contrôle virtuel basé sur des fonctions hyperboliques .
– Contrôle backstepping basé sur des quaternions.
– Convergence en temps fini basé sur le contrôle du mode glissant.

Algorithmes de navigation autonomes : La partie suivante de cette thèse s'est concentrée sur le développement d'algorithmes semi-autonome et autonome qui permettent de détecter et de suivre une cible mobile, ainsi que de générer des trajectoires désirées afin que les contrôleurs précédemment développés puissent suivre.

Résumé

Un algorithme de planification de trajectoire pour quadrotors basé sur un modèle de contrôle prédictif a été développé pour la génération autonome de trajectoires. L'algorithme a été résolu en temps réel grâce à une optimisation en ligne d'une fonction de coût. Ensuite, les trajectoires optimales ont été suivies à l'aide d'un simple contrôle borné non linéaire.

Ensuite, concernant la détection et le suivi d'une cible mobile au sol, une algorithme de navigation d'asservissement visuelle autonome a été proposé. L'algorithme considère que le véhicule terrestre imite un conducteur de voiture afin que ses mouvements soient décrits comme un champ vectoriel. De plus, le robot au sol est détecté en utilisant les propriétés de l'algorithme de flux optique bien connu. Ainsi, le quadrotor suit le véhicule au sol en utilisant la commande backstepping basée sur des quaternions.

Enfin, une architecture de contrôle virtuel semi-autonome permettant de contrôler des robots à distance a été développée en coopération avec l'équipe de réalité virtuelle du laboratoire Heudiasyc. L'architecture est composée d'un monde en miniature situé dans un environnement virtuel et d'un robot aérien réel situé dans une arène de vol. Le vrai robot aérien suit les positions et les vitesses désirent, provenant de l'environnement virtuel, à l'aide d'un contrôle backstepping basé sur le quaternion.

Schéma de contrôle robuste : La dernière partie de cette thèse était consacrée à la conception d'une architecture de contrôle robuste capable de contrer les incertitudes du modèle et les perturbations externes telles que les rafales de vent et la perte d'efficacité des rotors. Cette architecture peut être vue comme suit :

Une architecture de contrôle robuste basée sur un estimateur de perturbations a été introduite pour donner de la robustesse au système en boucle fermée vis-à-vis des incertitudes non linéaires et des petites perturbations externes. L'architecture a effectué des tests expérimentaux améliorant un contrôleur de rétroaction simple.

Ensuite, en améliorant l'architecture discutée précédemment, un nouveau schéma de contrôle robuste et tolérant aux pannes a été obtenu. Le schéma dispose en outre d'un observateur de défauts de rotors pour détecter et estimer la perte d'efficacité des actionneurs. Le contrôleur robuste peut estimer et contrecarrer les défauts des rotors en position non stationnaire et en position stationnaire.

Enfin, une architecture de contrôle robuste bornée a été développée à partir de l'observateur de perturbations et des propriétés de contrôle optimales. En plus de l'estimation et la compensation de dynamiques inconnues, ce schéma améliore sa robustesse en résolvant les contraintes physiques des moteurs. L'architecture a été prouvée expérimentalement avec plusieurs rafales de vent et une perte d'efficacité dans deux rotors.

Contents

List of Figures

List of Tables

Glossary

ADRC	Active Disturbance Rejection Control
AGV	Autonomous Ground Vehicle
CAVE	Cave Automated Virtual Environment
CoG	Center of Gravity
CoM	Center of Mass
DOBC	Disturbance Observer Based Control
DoF	Degrees of Freedom
DrEAM	Drone Exocentric Advanced Metaphor
Fl-AIR	Framework libre AIR
FoV	Field of View
FTC	Fault Tolerant Control
GESO	Generalized Extended State Observer
HSC	Hyperbolic Saturation Controller
LB	Linear Backstepping
LoE	Loss of Efficiency
MAS	Multi-rotor Aerial Systems
MPC	Model Predictive Control
NLB	Nonlinear Backstepping
NS	Nested Saturation
PI	Proportional Integrator
PID	Proportional Integral Derivative
PIO	Proportional-Integral Observer
PVTOL	Planar Vertical Take-off and Landing
qLPV	quasi-Linear Parameter Varying
SMC	Sliding Mode Control
UAV	Unmanned Aerial Vehicle
UAVs	Unmanned Aerial Systems
UDE	Uncertainty and Disturbance Observer
VTOL	Vertical Take-Off and Landing
WIM	World In Miniature

Chapter 1 Part I

1 Introduction

In the past decade Unmanned Aerial Vehicles (UAVs) have enjoyed considerable success in many applications such as search and rescue, monitoring, research, exploration, or mapping. Robotics research has advanced significantly the past years and it can now offer many elaborate solutions in artificial vision, path planning, decision-making, obstacle avoidance and environment modeling – all these works aiming at endowing mobile machines with autonomy [2]. The research and development in the field of Unmanned Aerial Systems (UAVs) is experiencing an increasing quantity of investment all around the world. Small UAVs are often used as experimental platforms to research on state estimation, control, robotic collaboration, etc; since their dynamic maneuvering and natural instability pose important challenges that are not present with other robotic platforms. In addition, the maneuverability of UAVs is combined with a low payload capability, which in turn requires the development of algorithms with constrained computation requirements. This fact has resulted in many research focused on testing and demonstrating the possibilities of several sensors on these platforms.

Nowadays, UAVs can take off, fly and land almost without any human intervention. The work developed in this thesis is focused on quadrotors or quadcopters vehicles. These robots are composed basically by four motors, an Inertial Measurement Unit (IMU) for obtaining the orientation of the vehicle, a position sensor such as a GPS and a Central Processing Unit (CPU) where the control algorithm is executed [3]. Furthermore, a quadcopter is capable of handling complex tasks in cramped and crowded environments, as well as it has a simple control mechanism compared to the other types of UAVs [4]. Due to the cost reduction, in the last years, potential applications of quadcotpers have emerged for civil and military uses, such as airborne surveillance [5], automated search and rescue [6], military information gathering, military offensive actions, electrical transmission line inspection [7] and automated forest fire surveillance [8], to cite a few. A common point to most of these applications is that they are defined with respect to a reference frame, where a target needs to be tracked.

For instance, in the airborne surveillance or in the automated search and rescue applications, a target must be defined to be tracked and this one can be static, slowly moving, or maneuvering at high speeds.

As a consequence of the aforementioned, there has been increasing interest in the quadrotor applications such as target-tracking, object detection and landing to drone station [9, 10, 11]. Especially, target-tracking is an essential task for quadrotors, as it has been proven to be an effective tool in search and rescue surveillance [12], following and providing aerial based video of sports events [13]. Also, it has become a relevant field for the correct development of many multidisciplinary applications, such as traffic supervision, autonomous robot navigation, mapping, and aerial photography.

1.1 Problem statement

The target tracking problem has an ever-growing number of civilian applications, ranging from traditional applications such as air traffic control and building surveillance to emerging applications such as supply chain management and wildlife tracking. For such tasks, and due the simplicity and versatility of multirotors, in recent years, a large part of the scientific community have focused their attention in these kind of vehicles in front of other types of UAVs [14]. Target tracking for UAV generally falls into two categories; cooperative tracking and non-cooperative tracking, respectively. In the cooperative tracking system, real time between the UAV and the target is required, such as a convoy [15]. Compared with cooperative tracking, non-cooperative tracking has less limitations. Instead of real-time interaction, in the non-cooperative tracking system, the UAV is equipped with a camera so that the information of the target can be obtained using vision-based method. For a successful tracking system, the target needs to be kept within the camera view and distinguished from the environment in real time by the onboard computer. Then the visual information derived from the processing of images is used to control the motion of UAV to track the target and keep it at the center of the image frame.

Autonomously tracking moving target using multirotors, as illustrated in Figure 1.1, is not an easy task and many challenges have to emerge in different stages. However, when using a quadrotor vehicle, this problem can be divided into three process; 1) taking off, 2) detection of the moving target, and 3) tracking the moving target. Since the quadrotors are dynamically unstable, under-actuated, and nonlinear systems, it is very challenging to design a controller for a quadrotors to track a moving target especially under the varying speed of the moving target and under the environment effects. In the process of tracking a moving target using a quadrotor, there are still several challenges: (1) the completion with high precision and reliability of the target tracking problem using a quadrotor, (2) the non-linearity and uncertainties in the quadrotor system and (3) the external perturbations that can occur when tracking a moving target.

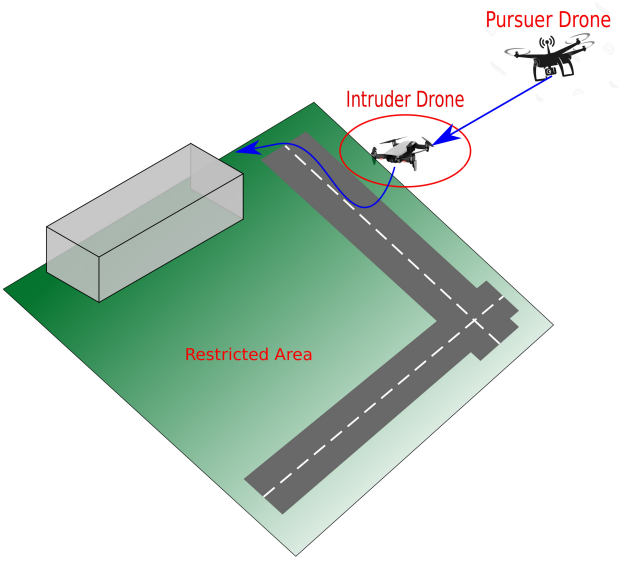


Figure 1.1 – Example of a target tracking problem: a pursuer drone follows an intruder quadrotor while flying in a restricted area.

The necessity of implementing novel and reliable autonomous navigation control methods for tracking moving targets, exploring the capabilities of robust control techniques and their applicability, defines the research framework from which this thesis has been developed. This thesis has a strong experimental and theoretical basis and the main research problems tackled during its realization have been: 1) a mathematical model of the aerial vehicle needs to be built such that it accurately describes the real vehicle but at the same time provides mathematical simplicity in order to aim at 2) the development of control algorithms that will deal with the stabilization and tracking problem, 3) Then, navigation methods must be designed to follow dynamic targets even in the presence of external disturbance, and consequently, 4) a robust control technique must to be exposed in order to overcome these external/internal perturbations.

1.2 Thesis objective

For tracking target tasks, there is a need for the aerial vehicle to anticipate the movements of the mobile target in order to be 'a step ahead'. This brings the interest to the level of the decision process: a target that is evasive is going to actively attempt to hide behind obstacles and use this advantage to continue evading. In order to anticipate to the movements of this target, the aerial vehicle must be able to predict such situations and to react or plan before they even happen in order to maintain the continuity of the target tracking.

One of the main difficulties when designing a control system for solving the target tracking problem is, undoubtedly, the good choice of the controller in the presence of disturbances. It should be remarked that disturbances are not necessarily external to the system itself, but they can be originated when sensing and/or actuating. Therefore, even if small, they can arise at any time.

This thesis will approach the problem of tracking moving targets by means of an aerial drone, in our case a quadrotor vehicle. The goal of this thesis is to develop control and navigation algorithms capable to robustly tracking dynamic targets. As a result, the following control techniques must be addressed: autonomous navigation algorithms for the detection and trajectory tracking of the moving target, control laws for stabilizing the quadrotor and tracking desired trajectories with finite time convergence and, an enhancing control algorithm for dealing with unknown model dynamics and external perturbations while satisfying motor's constraints.

The objectives of this thesis can be summarized as follows:

- Develop a control based navigation method that allows the quadrotor vehicle to detect and track a moving target.
- Develop a path planning algorithm based on model predictive control, capable of generating optimal trajectories based on target's position.
- Develop control algorithms for stabilizing a quadrotor while tracking desired trajectories with finite time convergence.
- Develop a robust control architecture capable of dealing with external disturbances and undesired dynamics such as rotors fault.

1.3 State of the art

Quadrotor control techniques

In control systems for multirotor aerial vehicles there are two different types of control, depending on the loop to which the controller is applied; the position controller and the attitude controller. In fact, there is no direct actuation control per se. Attitude control is the concept of pointing a craft in the desired direction. More specifically, it involves controlling the orientation of the robot on its three Euler angles (θ, ϕ, ψ) and controlling the robot's thrust in the z-axis of the robot frame of reference. This is performed due to the underactuation characteristics of the robot. Then, the outputs of this control are sent to a mixing-of-motors algorithm to generate the reference signals for the actuators of the MAVs. In these cases, the attitude control algorithm must take into account the number of motors and their respective control signal. In contrast, position control is the controller that gives the Euler angles as references to the attitude controller in order for the MAV to move in a desired trajectory [16].

For the last 10 years, attitude and position were popularly controlled by monitoring the altitude (z-axis) using feedback linearization followed by a linear/nonlinear controller to achieve the desired altitude. The values of the control torque can be found by replacing the controlled altitude in the attitude expression. This method for designing the control torque will be denoted *classical* approach in this work.

In addition, adaptive control has been studied to solve the problems of stabilization and tracking of the dynamics. In [17] a novel trajectory tracking scheme by combining two inner feedback linearization loops to stabilize the nominal quadrotor was presented. Moreover, a robust adaptive tracking control scheme based on a self-tuning regulator has been presented in [18]. This tracking method is implemented in the inner loop and a classical proportional integrator (PI) controller was employed in the outer loop such that the robustness of the whole system is guaranteed. Similarly, in [19] the altitude and attitude control laws for the quadrotor based on the passivity method were introduced; the simulation results demonstrate high efficiency and robustness with respect to the plant parametric uncertainty.

Furthermore, a time-varying backstepping technique is presented in [20], where a $3D$ control tracking law is studied and applied experimentally on a quadrotor. This control law is implemented by assuming that only the position and orientation parameters of the vehicles are known; whereas the linear and angular velocities are estimated using Luenberger observers. Later on, this work was improved in [21] by considering a non-linear time-varying version of the backstepping technique. This work takes all the non-linearities of the dynamics into account; resulting in a higher stability validated theoretically with numerical simulations.

Moreover, dynamic inversion is a new control technique developed in [22, 23] for the stabilization and dynamics tracking of a quadrotor vehicle. The tracking controller is composed of two loops; the dynamic inversion is implemented in the inner loop, where an internal dynamics stabilization is applied to the outer loop. The control objectives are enclosed in the form of constraint differential equation, and the resultant control law is obtained by inverting the constraint dynamics using Moore-Penrose Generalized Inverse (MPGI). Computer simulations were performed to exhibit the performance of the offered control technique in nominal and perturbed conditions. In addition to the dynamic inversion controllers, an integral backstepping controller with sliding mode control approach for a quadrotor was proposed in [24]. In this approach the control of the UAV is performed in the altitude and position control. The altitude control produces translational force, which is used to calculate virtual controls for the x and y movements and desired angles for pitch and roll. The proposed controller was tested in a Qball-X4 prototype. In the same sense, a hierarchical control was addressed in [25] based on sliding-mode and adaptive control techniques to deal with slow and fast time-varying wind conditions respectively. Simulations results show the validity of the proposed control strategy while tracking a time-parametrized straight-line and sinusoidal trajectory.

Recently, a new fashion to design the torque control of the vehicle is considering the vehicle model fully actuated by means of a virtual input. In this thesis, we will call the aforementioned as *virtual* approach. In this context, there are some works exploiting the differential flatness of the quadrotor model and then, with a trajectory generation algorithm, compute high-performance flight trajectories that are capable of moving a quadrotor from a large class of initial states to a given target point that will be reached at rest [26, 27, 28]. For example, in [29] a control scheme based on a virtual control input using a backstepping approach with Nussbaum function, a priori-bounded control torque for the rotational subsystem was designed to track the desired orientations generated by the translational subsystem.

In the same sense, a robust cost controller derived by Lyapunov stability theorem was addressed in [30]. The designed robust controller guarantees that the closed-loop system converge asymptotically stable and its robustness with respect to parameter uncertainties. Besides, in [31] the tracking control problem was solved using a two step approach. First, a translational control scheme that tracks the desired position trajectory is constructed assuming the translational dynamics to be fully actuated. The magnitude of the translational control input is used as the magnitude of the control thrust. Second, the unit vector representing the direction of the translational control input used as the desired thrust direction.

The application of quaternions for controlling aircraft systems was introduced in [32], where the authors used quaternion products to rotate control vectors between different reference frames. 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. Later on, quaternion-based techniques were also applied to quadrotor vehicles such as [33], where the authors proposed a quaternion high-order sliding mode algorithm to control attitude trajectory on spacecraft, while [34] 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 [35], where proportional-derivative (PD), linear quadratic regulartor LQR, and backstepping position and attitude controllers were compared with simulations. More recent contributions have explored other properties of quaternion-based dynamic systems. For example, [36] and [37] proposed fractional sliding-mode controllers using quaternions to stabilize the quadrotor attitude dynamics, while also considering finitetime convergence of the system dynamics based on a fractional-order system. In [38], researchers estimated quadrotor position and attitude information during aggressive trajectories by only employing cameras and inertial sensors, quaternion operations were used in the proposed algorithm.

Furthermore, a trajectory tracking control law with finite-time convergence on the nonlinear manifold SE(3) is more desirable since most of the existing finite-time control law only stabilizes attitude. The closed-loop systems under finite-time control usually demonstrates faster convergence rate, higher tracking precision, better disturbance rejection properties and robustness against uncertainties [39, 40, 41]. A continuous finite-time controller based on homogeneity was developed for robot manipulators in [42]. Following the same ideas, in [43] a finite-time attitude stabilizing result based on homogeneous method was presented. Latter on, this work was improved in [44] but in this case, the attitude was represented using Modified Rodriguez Parameters. A finite-time stabilization law with almost global convergence was developed for a rigid body using rotation matrices for representing attitude in [45, 46] but the method was not extended for the position dynamics.

Autonomous navigation algorithms

Planning algorithms

Trajectory planning problems for quadrotors were not widely studied using these approaches. This problem is mostly characterized as an optimization problem subject to constraints as velocity, bounded input, etc [47], [48]. Nevertheless, the path planning for autonomous navigation have been studied in many fields, including robotics, aerial vehicles, aquatics vehicles and space aircraft [49], [50], [51]. Yang et al. [52] treated the problem of time-optimal control of a quadcopter by using a nonlinear programming method coupled with a genetic algorithm. They succeeded to generate in simulation minimum time point-to-point trajectories under various technological constraints. Cowling et al. [52] presented an optimal trajectory planner with a linear control scheme to follow a reference trajectory. Authors exploited the differential flatness of the quadcopter to address the optimization problem within the output space.

In addition, in [53] the authors proposed a model predictive control based on trajectory tracking system for small unmanned helicopters. It is based on a linear model predictive controller and the simulations results show a good robustness for parameter uncertainty. In [54] authors addressed the stabilization with motion planning problem of a standard quadcopter. They showed that the system presents a flat output and exploited this fact in the treatment of the motion generation problem. Recently in [55] the authors introduce an architecture approach based on MPC for path planning in aerial vehicles. The approach considers the tactical formulation and reformulation of optimization parameters and constraints based on self-assessment and feasibility information from the MPC framework. The applicability of chosen approach has been demonstrated in simulative set-up for several planning scenarios.

A linear and nonlinear model predictive control scheme were presented in [56] to control a quadcopter platform with the goal to track various reference trajectories. The closed loop stability analysis of the linear and nonlinear control schemes were presented to show the similarities, as well as to highlight the differences. In [57] the authors develop a framework for real time, full-state feedback, unconstrained, nonlinear model predictive control that combines trajectory optimization and tracking control in a single, unified approach. The performance of the approach was validated on two different hardware platforms, an AscTec Firefly hexacopter and the ball balancing robot Rewero. More recent works as [26] present a trajectory generation algorithm that allows the fast computation of high-performance flight trajectories to move the quadcopter from a large class of initial states to a given target location. The algorithm was computed in a ground station and not on-board. In [58] the authors have been developed by solving an optimal control the minimum-energy paths trajectories. Only simulations were developed to validate the strategy established.

Aerial visual servoing

Regulation of visual relative position and yaw with a quadrotor, assuming 2D ground target, has been studied for landing [59], and saturated visual servoing [60], using spherical projection [61], also assuming that the target remains in the FoV, though [62] addressed how to handle constraints to enforce that, improving the control loop with a backstepping scheme in [63]. Moreover, simplified control design can be obtained with the virtual camera approach, [64], or exploiting the conventional perspective projection, [65], which represents image plane velocity at predefined height. In contrast, most approaches use the well-known eye-in-hand Image-Based Visual Servoing of [66], to derive an orthodox visual servoing of a ground target by quadrotors, [67]. However, neither approaches consider any physical constraint of the mobile robot and the quadrotor capabilities or limitations. Due to the complexity of airborne visual tracking, multiple processing units are considered to switch and balance computational load, [68]. Optical flow is a cautious alternative to provide damping at image plane, [69], given that it is not a derivative, similar to [68].

Advanced control schemes have been proposed, such as fractional control with conventional vision, [70], however the power of such scheme is not exploited due to its assumptions, which are based on conventional ideal conditions. The difficulty to implement the scheme arises since it considers attitude acceleration measurements, thus chattering is introduced in the second derivative of position control, unlike [71] where complex disturbance are compensated with continuous fractional control. Path tracking is also considered in [72] by formulating a constraint quadratic program problem given the field of view and maximum acceleration, however the numerical solution may not guarantee smooth enough trajectories, which may demand a high battery consumption attempting to reduce $\mathscr{L}_2$ norm.

Furthermore, a fuzzy engine is proposed to smooth out and produce allowable quadrotor trajectories from monocular camera, [73], however neglecting the physics of quadrotor and mobile robot, similar to [74], where a hybrid scheme is proposed with a mobile manipulator, with dual-task planning. More general sensory data are considered for drones to propose obstacle evasion and target, nevertheless assuming in addition perfect knowledge of obstacles and target for intersection planning in looking-ahead trajectories, [75], even in civil cluttered flying conditions, [76], all based on potential fields and civil aviation rulings, [77]. For missile drones, the look-ahead scheme is more critical, [78], similar to heavy oil tankers, [79], to avoid obstacles and converge to the target. In addition, it is customary to assume monocular camera based on the virtual camera approach that aligns the image plane collinear to the ground plane of the vehicle.

Semi-autonomous navigation

In extreme or unsafe environments (space exploration, tele-surgery, mine excavation, high pressure, high temperature, biological contamination and so on), traditional methods inevitably expose operators to danger on-site. Teleoperation systems are a promising solution when it is physically dangerous or impossible for an operator to be in a piece of a equipment or its neighborhood. Despite the concept of teleoperation was proposed decades ago [80], it has not been widely implemented in real applications. One of the major impediments is associated with the user's limited situational awareness [81]. It is defined as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [82].

New remote interaction solutions such as voice or video conferencing have reached a high level of sophistication and widespread use as an aid for the solution of the teleoperation problem. While the feeling of being present in a remote environment is clearly available with these systems, a complete immersion cannot be realized without the possibility of physical interaction with the remote environment. Haptic solutions, as an extension of visual and auditory modalities, refer to both kinesthetic and tactile information and include position, velocity, force, torque, vibration, etc. Using a teleoperation system with haptic feedback, the users can thus truly immerse themselves into a distant environment, i.e., modify it, and execute tasks without physically being present but with the feeling of being there. Several possible teleoperation schemes using haptic feedback to steer single or multiple robots have been proposed in the last decade [83, 84, 85, 86, 87, 88].

Within the robotics field, there has been great enthusiasm on multi-rotor vehicles due their characteristics: hovering capability, maneuverability and small size. Most of the applications designed for these vehicles concern, inspection and surveillance of places where can result inaccessible by humans. The possibility of letting aerial robots interacting with the environment, opens an additional wide set of potential applications for aerial robots like, i.e., maintenance, construction, cooperative grasping and transportation. Most of them require the physical interaction of the robots with the surrounding objects and uncertain and unknown environments that could make the fully-autonomous control of the vehicle practically unreliable. Thus, the teleoperation intervention becomes advisable to handle this kind of issues. In [89, 90, 91, 92] a bilateral teleoperation for Vertical Take off Landing (VTOL) vehicles, based on impedance controllers with haptic feedback and considering time-varying delays were carried out. Main results consist in providing the stability of the teleoperation loop with respect to bounded external forces (i.e. operator and environment forces). Simulation and experimental results were showed to verify the effectiveness and performance of the proposed schemes.

One important feature in a teleoperation system is the visual feedback of the remote environment. It comes often from one or more monocular cameras mounted on the robot, which lead to several problems as the limited field of view (FoV), lack of visualization and poor visibility.

New different aerial drones teleoperation, considering in a different manner the visual feedback, have been developed based on graphic interfaces [93, 94, 95, 96, 97, 98]. For instance, several simulations and experimental tests for the inspection of buildings using interfaces based on the head and gestures positions, and a wearable exoskeleton interface were presented in [93, 99]. The main strength of the proposed interaction methods is the ability to perform multi-modal interactions. In [100] an experimental study of gaze-based control modes for unmanned aerial vehicles with ten participants performing a simple flying task was performed. Several experimental tests were carried out. Furthermore, an approach based on Non-Invasive BCI device with expressive manner on face in remote presence using quadcopter control with the Emotive EPOC headset was implemented in [101]. Electroencephalogram signals were used in experimental tests to tele-operate the vehicle. A disadvantage for all of these interfaces is that the recognition algorithms must be precise and that the user needs to be close to the drone's camera in order to recognize the gestures or visual markers. In other words, it is required to have the vehicle in the line of sight (LoS).

Recent advances in the field of Virtual Reality (VR) have allowed to use this technology improving the teleoperation systems [102, 103]. For example, in [104] an immersive Augmented Reality (AR) environment for conduction remote maintenance via a robot has been proposed. The AR environment has been designed to enhance the operator's situational awareness of the robot as well as the maintenance tasks so as to reduce the cognitive load on the operator. However, neither simulation nor experimental tests were presented. In [105] a ground control station for drones based in an immersive virtual environment was developed. The environment simultaneously informs the operator about the position and condition of the vehicles. Hence, only simulation results were developed. Furthermore in [106] an interface based in a mixed reality environment and natural language for controlling UAVs was developed. Real experimental tests were carried out using a set of known virtual landmarks. Finally in [107] a wearable interface for drone teleoperation is presented. This interface was constituted from a data glove to allow the user to control the drone trajectory by hand motion and a haptic system used to augment their awareness of the environment surrounding the robot. Nevertheless, simulations and experimental tests were performed in line of sight.

Disturbance rejection robustness

With the increasing requirement of autonomous flight under different conditions, controlling a quadcopter stills an important challenge, for example, the position and attitude controls of the quadcopter are extremely sensitive to external disturbances, such as wind gusts. And therefore, the design of robust regulators become more and more attractive to outdoor missions and practical applications. In particular, robustness issues may be critical for quadcopter control since they can be subjected to undesired nonlinear dynamics and external disturbances. In order to weaken the effect of wind gusts, some authors have proposed complex control techniques to achieve stability. To deal with this problem, in [108, 109, 110] adaptive controllers with different structures were proposed and verified by simulations and experiments on test bench.

Disturbance Observer Based Control (DOBC) has been demonstrated to deal with disturbances without adding a significant computational requirement. These techniques are based on observing the inputs and the outputs of the plant and, based on the theoretical response that should be obtained, attributing everything inexplicable to a disturbance [111]. Such behavior can range from external disturbance to modeling errors and nonlinearities. Nevertheless, the provenance of the disturbance is not relevant for these methodologies. Among the most used DOBC techniques are the following: Disturbance Observer (DOBC) [112], which uses the theoretical transfer function to observe the theoretical input and the difference is the disturbance; Active Disturbance Rejection Control (ADRC) [113], which approximates the plant to a integrator chain and all the other behavior is disturbance; Generalized Extended State Observer (GESO) [114], which uses an observer with the model and assumes that the disturbance is constant, and the Uncertainty and Disturbance Observer (UDE) [115], which uses the model and the known plant state in order to obtain an approximation of the disturbance.

In addition to this, some works have been proposed to deal with variable delays [116] or even Networked control systems [117] in order to add uncertainties that could arise into the system and as well as improving the robustness in minimum phase systems [118, 119]. Likewise, these techniques have been successfully validated in multitude of real problems such as robotic manipulators [120], positioning of UAVs [121] or the magnetic levitator [122]. Among the presented techniques is worth to mentioning the UDE. In the state of art of this technique, there are many examples that have been validated experimentally when dealing with the quadrotor system. For instance, in [123], a comparison between the UDE and the classic PID to highlight the advantages and disadvantages of each one of the controllers is presented. Furthermore, a quadrotor slung load control, where the UDE compensates the variable mass, was proposed in [124]. And finally, in [125], the authors present a study of the UDE for controlling a quadrotor using the quaternion formalism.

Dealing with rotors failures

Fault Tolerant Control (FTC) techniques [126] are used to deal with failures that could arise from the system, preventing the performance degradation and making the control system safe and reliable. These techniques can be classified into passive and active [127]. In the first case, the controller works itself to minimize the effect of the failures but, it is not capable to overcome them. Nevertheless, in the second case, when the fault occurs, a self-reconfigurable controller is applied in order to maintain the desired performance of the system. These kinds of techniques generally use the following methodology [128]: the identification of the plant in regular operation, control tuning for the regular operation, detection and identification of the possible fault and, the control law reconfiguration.

For instance in [129], the authors proposed an active FTC based on a Gain-scheduled PID combined with an observer capable to detect and identify faults on a quadrotor.

A passive FTC was proposed in [130, 131] using a Sliding Mode Control (SMC) combined with a heuristic controller that was optimized for different possible faults scenarios and in [132] an extra sensor observer for thrust fault was presented. Recently, works on active FTC for quadcopter vehicles subjected to actuator partial faults can be found in [133, 134, 127]. In [133], [134] the faults were injected when the vehicle is flying at hover position. To the best of our knowledge, only a few experimental validation for a quadcopter flying in a non-hover position have been validated, see [127] and [135]. In [127], the authors developed a Fault Detection (FD) and accommodation algorithm using nonlinear adaptive estimation techniques. In addition, three observers were considered for the FD: a nonlinear adaptive observer and a linear Proportional-Integral Observer (PIO) applied to a Planar VTOL and a quasi-Linear Parameter Varying (qLPV) for a quadcopter.

Dealing with rotor's constraints

Despite several works found in the literature proposing robustness into the system, few of them bounds the control inputs for limiting the energy applied into the system. This is, because in some cases bounding, the control input the robustness is only guaranteed in a small range of external disturbances.

Bounding the control input is a challenge when implementing theoretical control results in real systems because, if the control signal exceeds the control inputs limits, it may lead to undesirable behavior of the system and damage its performance. The anti-windup technique is frequently used in wide range of applications to solve this problem. For example, the analysis of the aircraft control system with input saturation conducted in [136] has shown existence of hidden oscillations, which were compensated by an anti-windup scheme. In [137] the anti-windup approach was proposed and then applied for the aircraft control problem in [138, 139]. This anti-windup algorithm was based on the property of convergence for marginally stable linear plants with saturated inputs. Furthermore, in [140] an iterative thrust-mixing scheme based on the LQR approach, to compute the desired single rotor thrusts and a prioritizing motor-saturation, was validated in real-time tests. In [141], a cascade integration of Incremental Nonlinear Dynamic Inversion (INDI) for the attitude and position control of micro aerial vehicles is addressed. Wind tunnel experiments show that the vehicle can enter and leave the 10 m/s wind tunnel flow with only 21 cm maximum position deviation on average. In most of all these works, the desired collective thrust and body torques need to be converted into four single rotor thrusts, which can then be applied by knowing the mapping from motors commands to rotor thrusts, denoted as *thrust mapping*.

1.4 Thesis outline

This thesis is structured into four Chapters. Chapter II provides an introduction to the classical modeling quadrotor approaches. Fundamental concepts regarding the Euler-Lagrange, Newton-Euler and quaternion formalism are revisited.

A unified and accessible analysis, placing popular control algorithms into a proper context are addressed in Chapter III. The chapter is divided as follows: a comparison between popular and virtual control algorithms and a meticulous analysis is introduced in this chapter. Furthermore, a quaternion-based backstepping control is developed in the chapter. Finally, the finite-time convergence using sliding mode control is revisited. Experimental and numerical simulations results are provided to verify the performance of each control technique.

Part IV establishes some autonomous and semi-autonomous navigation algorithms that are combined with the controllers developed in Chapter III. This chapter is divided as follows: a path planning algorithm using model predictive control for the trajectory generation, a visual servoing control for the tracking and target detection and finally, a virtual control architecture for controlling robots remotely using a immersed virtual environment. Experimental results are provided in order show the good performance of the proposed algorithms as well their limitations.

Finally, Chapter V takes a slightly different perspective, as it provides a comprehensive and unified guide to understand disturbance rejection based observers from the theoretical and experimental point of view. This chapter is divided into three sections: the introduction of a robust control scheme for counteracting undesired dynamics, the enhanced robust architecture for counteracting loss of effectiveness in rotors, and a robust bounded control to handle motors constraints while counteracting external perturbations. Several numerical and experimental results are exposed to demonstrate the good performance of the developed architectures.

Chapter 2 Part II

2 Modeling quadcopter vehicle approaches

Before stepping into the quadrotor control and navigation design, the mathematical model of the vehicle must be studied. The aim of this chapter is to present popular modeling approaches that exploit the whole properties of the vehicle. Considering all the parameters that fulfill the complete dynamic model of an aerial robot, e.g., the aeroelastic flutter effect, internal dynamics of the rotors, etc, are particularly important for control purposes. However, it comes when a whole set of changing variables that sometimes becomes unmanageable. Therefore, it is interesting to consider also simplified models constituted by a minimum number of states and inputs, but retaining the main features that must be considered when designing control laws for a real aircraft.

Most of the work developed for representing the dynamics of aerial vehicles are based on the Euler angles convention. This method is intuitive and easy to implement when simplifications are considered, e.g. non acrobatic maneuvers (slow movements, small angles). However, when more difficult tasks or applications are involved, the mathematical representation of the robot using this approach encounters some problems such as, complicated non-linear mathematical expressions, singularities, and gimbal lock.

Other option for representing the rotation of rigid bodies is the quaternion algebra. This kind of formalism has considerable advantages in comparison with the Euler angles method. For instance: lack of discontinuities and gimbal lock, and provision of mathematical simplicity. Moreover, quaternion multiplication makes less computational overhead in comparison to Euler angles because of its vector representation. In addition, quaternions need less memory space in comparison to Euler angles.

The models for the aerial vehicle developed in this thesis assume the following:

- The structure is supposed rigid.
- The structure is supposed symmetrical.
- The CoG and the body fixed frame origin are assumed to coincide.
- The propellers are supposed rigid.
- Thrust and drag are proportional to the square of propeller's speed.

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 approaches based on Euler angles are presented. An explanation of the quaternion-based modeling methodology is detailed in Section 2.3 and finally, some conclusions are discussed in Section 2.4.

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 [142]. 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 can be stated [143]:

$$\begin{aligned} f_i &= C_T \rho A_p r^2 \omega_i^2 \\ \tau_i &= C_Q \rho A_p r^3 \omega_i^2 \end{aligned} \tag{2.1}$$

where f_i represents the total thrust produced by rotor $i = 1, \cdots, 4$, 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 [144], see Figure 2.1.

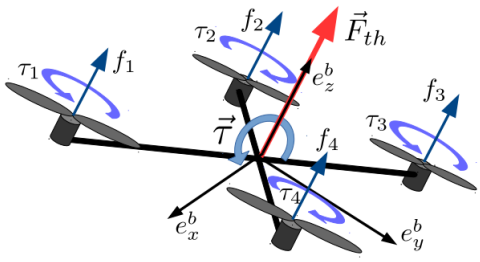


Figure 2.1 – Propeller forces and torques acting on a quadrotor.

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

$$\vec{F}_{th} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} k_T \omega_i^2 \end{bmatrix}, \ \vec{\tau} = \begin{bmatrix} \tau_\theta \\ \tau_\phi \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l k_T \left(\omega_1^2 - \omega_1^2 - \omega_3^2 + \omega_4^2\right) \\ l k_T \left(\omega_1^2 + \omega_1^2 - \omega_3^2 - \omega_4^2\right) \\ \sum_{i=1}^{4} k_Q \omega_i^2 (-1)^2 \end{bmatrix}, \tag{2.2}$$

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

2.2 Classical modeling approaches

In this section the three most popular techniques for modeling aerial vehicles will be introduced: the Euler-Lagrange, the Newton-Euler and the Euler-Lagrange based on quaternion formalism.

2.2.1 Euler-Lagrange approach

The model is obtained by representing the aircraft as a rigid body evolving in a $3-D$ space due to the main thrust and three torques: pitch, roll and yaw as depicted in Figure 2.2.

Let the generalized coordinates of the aerial vehicle be expressed by

$$\vec{p} = (x, y, z, \psi, \theta, \phi) \in \mathbb{R}^6 \tag{2.3}$$

where $\vec{\xi} = (x, y, z) \in \mathbb{R}^3$ denotes the position of the center of mass of the quadcopter relative to a fixed inertial frame $\mathscr{I}$, and $\vec{\eta} = (\psi, \theta, \phi) \in \mathbb{R}^3$ are the Euler angles, ψ is the yaw angle around the $z-$axis, θ is the pitch angle around the modified $y-$axis, and ϕ is the roll angle around the modified $x-$axis, which represent the orientation of the quadcopter.

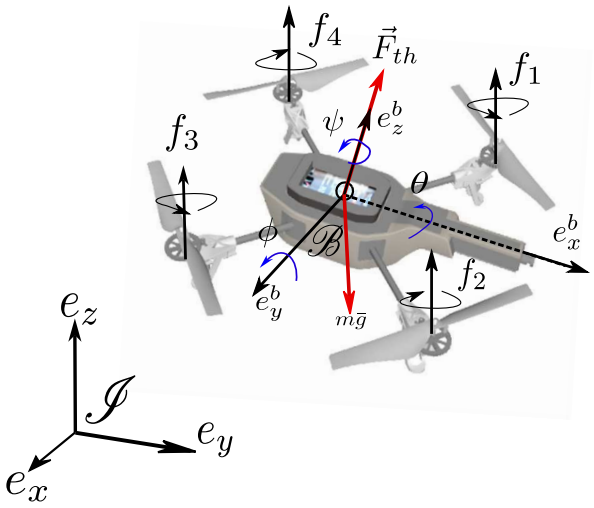


Figure 2.2 – Coordinate systems acting in the aerial vehicle where f_i denotes the force applied of each motor, $\vec{F}_{th}$ symbolizes the total thrust force and θ, ϕ, ψ are the Euler angles on each axis.

Defining the Lagrangian

$$L(\vec{p}, \dot{\vec{p}}) = T_{trans} + T_{rot} - U, \tag{2.4}$$

where $T_{trans} = \frac{m}{2}\dot{\vec{\xi}}^T\dot{\vec{\xi}}$ is the translational kinetic energy, $T_{rot} = \frac{1}{2}\vec{\Omega}^T I\vec{\Omega}$ means the rotational kinetic energy, $U = m\vec{g}\vec{\xi}$ represent the potential energy of the rotorcraft, $\vec{\Omega}$ signifies the angular velocity, I the inertia matrix, and $\vec{g} = [0\ 0\ -g]$ depicts the acceleration vector due the gravity.

The angular velocity vector $\vec{\Omega}$ resolved in the body frame is related to the generalized velocities $\dot{\vec{\eta}}$ by means of the kinematic relationship [145]

$$\dot{\vec{\eta}} = W_v^{-1}\vec{\Omega}, \tag{2.5}$$

where

$$W_v = \begin{bmatrix} -\sin\theta & 0 & 1 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ \cos\theta\cos\psi & -\sin\psi & 0 \end{bmatrix}, \ \vec{\Omega} = \begin{bmatrix} \dot{\phi} - \dot{\psi}\sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \\ \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi \end{bmatrix} \tag{2.6}$$

Therefore $T_{rot} = \frac{1}{2}\dot{\vec{\eta}}^T \mathbb{J}\dot{\vec{\eta}}$ with $\mathbb{J} = \mathbb{J}(\vec{\eta}) = W_v^T I W_v$ and

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_z z \end{bmatrix}$$

Note, the matrix $\mathbb{J} = \mathbb{J}(\vec{\eta})$ acts as the inertia matrix for the full rotational kinetic energy of the quadcopter expressed in terms of the generalized coordinates $\vec{\eta}$. Then, the dynamics equations describing the behavior of the full aerial vehicle are obtained from Euler-Lagrange equations with external generalized forces

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{\vec{p}}} - \frac{\partial \mathscr{L}}{\partial \vec{p}} = \begin{bmatrix} \vec{F}_\xi \\ \vec{\tau} \end{bmatrix}, \tag{2.7}$$

where $\vec{F}_\xi = R\vec{F}_{th} \in \mathbb{R}^3$ is the translational force applied to the quadcopter due to main thrust directed out of the top of the vehicle with $\vec{F}_{th} = [0\ 0\ u_1]^T$ with $u_1 = \left\|\vec{F}_{th}\right\|$. R denotes the rotational matrix $R(\psi,\theta,\phi) \in SO(3)$ representing the orientation of the quadcopter relative to a fixed inertial frame:

$$R = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta S_\psi & C_\phi C_\psi + S_\psi S_\theta S_\phi & C_\phi S_\psi S_\theta - S_\phi C_\psi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{2.8}$$

where C_θ and S_θ stand for $\cos\theta$ and $\sin\theta$, respectively. Developing (2.7), the Euler-Lagrange equation for the translation and attitude motion can be written as

$$m\ddot{\vec{\xi}} = \vec{F}_\xi - m\vec{g} \tag{2.9}$$

$$\mathbb{J}\ddot{\vec{\eta}} = \vec{\tau} - C(\vec{\eta},\dot{\vec{\eta}})\dot{\vec{\eta}} \tag{2.10}$$

where $C(\vec{\eta},\dot{\vec{\eta}}) = \dot{\mathbb{J}} - \frac{1}{2}\frac{\partial}{\partial\vec{\eta}}(\dot{\vec{\eta}}^T\mathbb{J})$ refers to the Coriolis term, which contains the gyroscopic and centrifugal terms associated with the $\vec{\eta}$ dependence of $\mathbb{J}$. Expanding (2.10) is not an easy task and in several works, the full inertia matrix $\mathbb{J}$ is considered as diagonal and the Coriolis matrix is neglected.

Let us consider the case where the $\mathbb{J}$ is considered diagonal and $C(\vec{\eta},\dot{\vec{\eta}})$ is neglected. Thus, to simplify, it is possible to define

$$\vec{\tilde{\tau}} = \begin{bmatrix} \tilde{\tau}_\theta \\ \tilde{\tau}_\phi \\ \tilde{\tau}_\psi \end{bmatrix} = \mathbb{J}^{-1}(\vec{\tau} - C(\vec{\eta},\dot{\vec{\eta}})). \tag{2.11}$$

Finally, we obtain from (2.9) and (2.10) considering (2.11)

$$\begin{aligned} m\ddot{x} &= -u_1 \sin\theta, & \ddot{\theta} &= \tilde{\tau}_\theta \\ m\ddot{y} &= u_1 \cos\theta \sin\phi, & \ddot{\phi} &= \tilde{\tau}_\phi \\ m\ddot{z} &= u_1 \cos\theta \cos\phi - mg, & \ddot{\psi} &= \tilde{\tau}_\psi \end{aligned} \tag{2.12}$$

where x and y are coordinates in the horizontal plane, z is the vertical position, and $\tilde{\tau}_\psi$,$\tilde{\tau}_\theta$, and $\tilde{\tau}_\phi$ are the yawing moment, pitching moment, and rolling moment, respectively.
For the case when the Coriolis and the inertial matrix for the $\vec{\eta}$ dynamics in (2.7) are considered, therefore, rewriting the attitude dynamics yields

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

then $\frac{\partial \vec{\Omega}}{\partial \vec{\eta}} = W_\eta$, $\vec{\Omega}^T I \frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}} = [b_1\ b_2\ b_3]$ with

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

differentiating $\vec{\Omega}^T I \frac{\partial \vec{\Omega}}{\partial \dot{\vec{\eta}}}$,

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

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 $\vec{\Omega}^T I \frac{\partial \vec{\Omega}}{\partial \vec{\eta}} = [h_1 \ h_2 \ h_3]$, where

$h_1 = 0,$

$h_2 = -I_{xx}\left(\dot{\psi}\dot{\phi}C_\theta - \dot{\psi}^2 S_\theta C_\theta\right) - I_{yy}\left(\dot{\psi}\dot{\theta}S_\theta S_\phi C_\phi + \dot{\psi}^2 S_\theta C_\theta S_\phi^2\right) - I_{zz}\left(\dot{\psi}^2 S_\theta C_\theta C_\phi^2 - \dot{\psi}\dot{\theta}S_\theta S_\phi C_\phi\right),$

$h_3 = I_{yy}(-\dot{\theta}^2 S_\phi C_\phi - \dot{\psi}\dot{\theta}C_\theta S_\phi^2 + \dot{\psi}\dot{\theta}C_\theta C_\phi^2 + \dot{\psi}^2 C_\theta^2 S_\phi C_\phi) I_{zz}(-\dot{\psi}^2 C_\theta^2 S_\phi C_\phi + \dot{\psi}\dot{\theta}C_\theta S_\phi^2 - \dot{\psi}\dot{\theta}C_\theta C_\phi^2 + \dot{\theta}^2 S_\phi C_\phi),$

such that

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

Then, regrouping all the terms developed above and using (2.10), thus, $\mathbb{J}$ is expressed as

$$\mathbb{J}(\vec{\eta}) = \begin{bmatrix} I_{xx}S_\theta^2 + I_{yy}C_\theta^2 S_\phi^2 + I_{zz}C_\theta^2 C_\phi^2 & C_\theta C_\phi S_\phi (I_{yy} - I_{zz}) & -I_{xx}S_\theta \\ C_\theta C_\phi S_\phi (I_{yy} - I_{zz}) & I_{yy}C_\phi^2 + I_{zz}S_\phi^2 & 0 \\ -I_{xx}S_\theta & 0 & I_{xx}, \end{bmatrix} \tag{2.13}$$

and

$$C(\vec{\eta}, \dot{\vec{\eta}}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.14}$$

where

$$\begin{aligned}
c_{11} &= I_{xx}\dot{\theta}S_\theta C_\theta + I_{yy}(-\dot{\theta}S_\theta C\theta S_\phi^2 + \dot{\phi}C_\theta^2 S_\phi C\phi) - I_{zz}(\dot{\theta}S_\theta C_\theta C_\phi^2 + \dot{\phi}C_\theta^2 S_\phi C_\phi), \\
c_{12} &= I_{xx}\dot{\psi}S_\theta C_\theta - I_{yy}(\dot{\theta}S_\theta S_\phi C_\phi + \dot{\phi}C_\theta S_\phi^2 - \dot{\phi}C_\theta C_\phi^2 + \dot{\psi}S_\theta C_\theta S_\phi^2) \\
&\quad + I_{zz}(\dot{\phi}C_\theta S_\phi^2 - \dot{\phi}C_\theta C_\phi^2 - \dot{\psi}S_\theta C_\theta C_\phi^2 + \dot{\theta}S_\theta S_\phi C_\phi) \\
c_{13} &= -I_{xx}\dot{\theta}C_\theta + I_{yy}\dot{\psi}C_\theta^2 S_\phi C_\phi - I_{zz}\dot{\psi}C_\theta^2 S_\phi C_\phi, \\
c_{21} &= -I_{xx}\dot{\psi}S_\theta C_\theta + I_{yy}\dot{\psi}S_\theta C_\theta S_\phi^2 + I_{zz}\dot{\psi}S_\theta C_\theta C_\phi^2, \\
c_{22} &= -I_{yy}\dot{\phi}S_\phi C_\phi + I_{zz}\dot{\phi}S_\phi C_\phi, \\
c_{23} &= I_{xx}\dot{\psi}C_\theta + I_{yy}(-\dot{\theta}S_\phi C_\phi + \dot{\psi}C_\theta C_\phi^2 - \dot{\psi}C_\theta S_\phi^2) + I_{zz}(\dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2 + \dot{\theta}S_\phi C_\phi), \\
c_{31} &= -I_{yy}\dot{\psi}C_\theta^2 S_\phi C_\phi + I_{zz}\dot{\psi}C_\theta^2 S_\phi C_\phi, \\
c_{32} &= -I_{xx}\dot{\psi}C_\theta + I_{yy}(\dot{\theta}S_\phi C_\phi + \dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2) - I_{zz}(\dot{\psi}C_\theta S_\phi^2 - \dot{\psi}C_\theta C_\phi^2 + \dot{\theta}S_\phi C_\phi), \\
c_{33} &= 0.
\end{aligned} \tag{2.15}$$

Although (2.9) and (2.10) were developed taking into account a multicopter with four rotors, these equations are also valid for other aerial configurations as long as the forces and torques are rewritten.

2.2.2 Newton-Euler approach

The general motion of a rigid body in space is a combination of translational and rotational motions. Consider a rigid body moving in inertial space, undergoing both rotations and translations, see Figure 2.2. Following same ideas as in section 2.2.1, let us define now an earth fixed frame $\mathscr{I}$ and a body frame $\mathscr{B}$. The center of mass and the body frame are assumed to coincide. Using Euler angles parametrization, the airframe orientation in space is given by a rotation matrix R from $\mathscr{B}$ to $\mathscr{I}$. Using the Newton-Euler formalism [146, 147, 148, 149], the dynamics of a rigid body under external forces applied to the center of mass and expressed on earth fixed frame is

$$\begin{aligned} \dot{\vec{\xi}}(t) &= \vec{v}(t), & m\dot{\vec{v}}(t) &= R(t)\vec{F}_{th}(t) - m\vec{g}, \\ \dot{R}(t) &= R(t)\hat{\vec{\Omega}}(t), & I\dot{\vec{\Omega}} &= -\vec{\Omega}(t) \times I\vec{\Omega}(t) + \vec{\tau}(t) \end{aligned} \tag{2.16}$$

where $\hat{\vec{\Omega}}(t)$ introduces the skew-symmetric matrix of $\vec{\Omega}(t)$, and I represents the constant inertia matrix.

Considering the total forces and torques from (2.2), and introducing them into (2.16), it follows that,

$$\begin{aligned} m\ddot{x} &= -\sin\theta u_1, \\ m\ddot{y} &= \cos\theta \sin\phi u_1 \\ m\ddot{z} &= \cos\theta\cos\phi u_1 - mg \end{aligned} \tag{2.17}$$

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ \ddot{\psi} \end{bmatrix} = J^{-1}\left(\begin{bmatrix} \tau_\theta \\ \tau_\phi \\ \tau_\psi \end{bmatrix} - \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \right). \tag{2.18}$$

Equations (2.17) and (2.18) describes the translational and rotational dynamics, respectively. Remark that making the same assumptions as before, an inertial diagonal matrix and that the Coriolis matrix is neglected, it is possible to obtain the same equations as in (2.12).

2.3 Quadrotor quaternion model based on Euler-Lagrange

For consistency reasons, and for building the mathematical background for following the proposed modeling, it is essential to understand quaternions and their associated algebra [150]. Quaternions were proposed by Hamilton in the nineteen century as a three-dimensional version of complex numbers (one real and one imaginary part). However, they can be also written as a vector with 4 scalar components, q_0, q_1, q_2, q_3, where q_1, q_2, q_3 correspond to the distance along the quaternion bases vector of i, j, k and q_0 is considered the scalar part of the quaternion.

They are also known as a $4-$dimensional hyper-complex number, since they can be represented as one real plus three imaginary numbers, $\boldsymbol{q} \triangleq q_0 + q_1 i + q_2 j + q_3 k$, where $i, j, k \in \mathbb{I}$, such that $i^2 = j^2 = k^2 = ijk = -1$ 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 $\vec{\boldsymbol{q}} = [q_1 \ q_2 \ q_3]^T$. In literature, one of the well-known expression to denote a quaternion relays on the representation given by the French bankern Olinde Rodriguez. He expanded the Euler's Formula to include 3−dimentional rotations using quaternions. This expression is known nowadays as the Euler-Rodrigues formula which is the exponential mapping of the axis-angle representation $\vec{n}, \varsigma$ of a rotation defined as

$$\boldsymbol{q} = \exp^{\frac{1}{2}\varsigma\vec{n}} = \cos(\varsigma/2) + \vec{n}\sin(\varsigma/2) \tag{2.19}$$

notice that $\|\boldsymbol{q}\| = 1$, thus $\boldsymbol{q}$ can be called a unit quaternion. Operations with quaternions are not solved with standard mathematical tools, therefore, the basisc quaternion operations must be defined.

Quaternion algebra

Product

Consider two quaternion $\boldsymbol{p}$ and $\boldsymbol{q}$ that are in $\mathbb{H}$. The quaternion product can be expressed as

$$\begin{aligned} \boldsymbol{p} \otimes \boldsymbol{q} &= (p_0 + p_1 i + p_2 j + p_3 k)(q_0 + q_1 i + q_2 j + q_3 k) \\ &= (p_0 q_0 - \vec{p} - \vec{q}, q_0 \vec{p} + \vec{p} \times \vec{q}). \end{aligned} \tag{2.20}$$

Remark that $\boldsymbol{p} \otimes \boldsymbol{q} \neq \boldsymbol{q} \otimes \boldsymbol{p}$.

Conjugate

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

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

Norm

The norm of a quaternion is defined as

$$\begin{aligned} \|\boldsymbol{q}\|^2 = N^2(\boldsymbol{q}) &= q_0^2 + q_1^2 + q_2^2 + q_3^2 \\ &= \boldsymbol{q} \otimes \boldsymbol{q}^* \end{aligned} \tag{2.22}$$

Inverse

For any non-null quaternion there exists an inverse quaternion such that

$$\boldsymbol{q}^{-1} := \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|} \tag{2.23}$$

Vector rotation

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

$$\vec{p}' = \boldsymbol{q}^{-1} \otimes \vec{p} \otimes \boldsymbol{q} = \boldsymbol{q}^* \otimes \vec{p} \otimes \boldsymbol{q} \tag{2.24}$$

Derivative

The derivative of any unit quaternion can be defined as follows

$$\begin{aligned} \boldsymbol{q}(t) &= q_0(t) + \vec{q}(t) = \cos\left(\frac{\theta}{2}\right) + \vec{u}\sin\left(\frac{\theta}{2}\right) \\ \boldsymbol{q}(\Delta t) &= q_0(\Delta t) + \vec{q}(\Delta t) = \cos\left(\frac{\Delta t}{2}\right) + \vec{u}\sin\left(\frac{\Delta t}{2}\right) \end{aligned} \tag{2.25}$$

where $\boldsymbol{q}(\Delta t)$ is a quaternion that represents an infinitesimal change in the angle while keeping the same unit vector $\vec{u}$. Thus the derivative of a unit quaternion can be expressed as

$$\begin{aligned} \frac{d}{dt}\boldsymbol{q}(t) &= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(1 + (\frac{\vec{u}}{2})\Delta\theta)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{1}{2}\frac{\boldsymbol{q}(t)\vec{u}\Delta\theta}{\Delta t} \\ &= \frac{1}{2}\boldsymbol{q}(t) \otimes \vec{\omega}(t). \end{aligned} \tag{2.26}$$

where $\vec{\omega}(t) = \dot{\theta}\vec{u}$ corresponds to the angular velocity vector associated with the quaternion $q(\Delta t)$.

2.3.1 Quadrotor quaternion dynamical model

Let us consider that the center of mass of the quadcopter is aligned with the origin of the body. The vector $\vec{\xi}$ goes from the origin of the inertial frame to the body frame and it is described as $\vec{\xi} = [x\ y\ z]^T$, $\dot{\vec{\xi}} = [\dot{x}\ \dot{y}\ \dot{z}]^T$, where $\dot{\vec{\xi}}$ defines the translational velocity seen from the inertial frame. Inspired on the Lagrangian methodology, the translational kinetic energy is defined as $E_{kt} = \frac{1}{2}m\dot{\vec{\xi}}^T\dot{\vec{\xi}}$ and the potential energy $U_p = mge_z$ where the mass of the vehicle is described by m, e_z is a unitary vector with the form $e_z = [0\ 0\ 1]^T$ and g is the term associated with the gravitational acceleration.

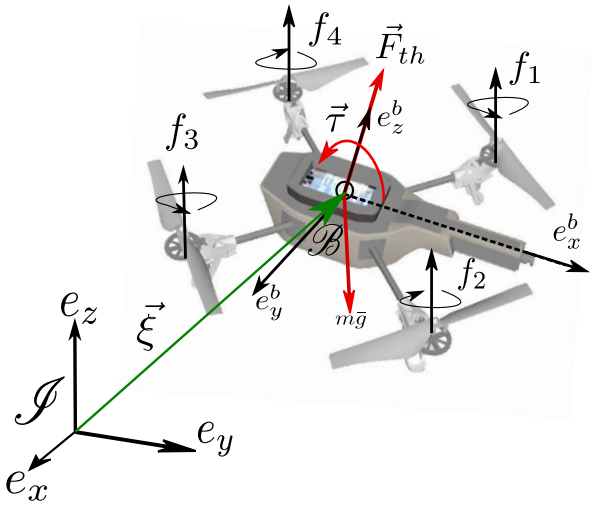


Figure 2.3 – Quadcopter free body diagram

Thus, using (2.7) the Euler-Lagrange equation for the translational system is expressed as

$$\frac{d}{dt}\left(\frac{\partial L_T}{\partial \dot{\vec{\xi}}}\right) - \frac{\partial L_T}{\partial \vec{\xi}} = \vec{F}_{th} \tag{2.27}$$

where $L_T = E_{kt} - U_p$ is the Lagrangian of the system, $\vec{\xi} \in \mathbb{R}^3$ are the generalized coordinates and $\vec{F}_{th}$ are the external forces acting on the quadcopter, respectively. Then, developing (2.27) it follows that

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = \vec{F}_{th} \tag{2.28}$$

The external forces acting on the quadrotor are the lift forces, which are produced by the rotation of the motors and propellers and its expression is defined as before stated, $\vec{F}_{th} = [0\ 0\ u_1]^T$. Remark that if the quadrotor changes its orientation (e.g., tracking a desired reference), the weight vector, $[0\ 0\ -mg]^T$ is always pointing downward, this means that the vector needs to be expressed in the inertial frame. However, the lift force changes its orientation as the vehicle does it. This implies that the lift force needs to be defined in the body frame. Then, considering the aforementioned, the external forces can be expressed as a function of the attitude of the vehicle as

$$\vec{F}_{\xi} = \boldsymbol{q}^* \otimes \vec{F}_{th} \otimes \boldsymbol{q} \tag{2.29}$$

where $\boldsymbol{q} = [q_0\ q_1\ q_2\ q_3]^T$ is the quaternion which defines the attitude performance of the quadrotor. Using the quaternion algebra [150], the translation dynamic model (2.28) can be rewritten as

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{bmatrix} = \begin{bmatrix} (-q_0 q_2 + q_1 q_3)u_1 \\ (q_0 q_1 + q_2 q_3)u_1 \\ (q_0^2 - q_1^2 - q_2^2 + q_3^2)u_1 - mg \end{bmatrix}. \tag{2.30}$$

For the rotational part, the Lagrangian can be written considering the kinetic rotational energy as

$$L_{rot} = \frac{1}{2}\vec{\Omega}^T I \vec{\Omega} \tag{2.31}$$

where $\vec{\Omega}$ is the angular velocity referred in the body frame. Remembering that the quaternion used in this analysis has the constraint $c = \boldsymbol{q}^T \otimes \boldsymbol{q} - 1 = 0$, then, the Euler-Lagrange equation for the attitude system is represented by

$$\frac{d}{dt}\left(\frac{\partial L_{rot}}{\partial \dot{\vec{\Omega}}}\right) - \frac{\partial L_{rot}}{\partial \vec{\Omega}} = \vec{\tau} + \lambda \frac{\partial c}{\partial \vec{\Omega}} \tag{2.32}$$

with λ representing the Lagrange multiplier. Using the property (2.26), the Lagrangian described by the equation (2.31) can be rewritten as follows

$$\begin{aligned} L_{rot} &= 2(G(\boldsymbol{q}) \otimes \dot{\boldsymbol{q}})^T \otimes I \otimes (G(\boldsymbol{q}) \otimes \dot{\boldsymbol{q}}) \\ &= 2\dot{\boldsymbol{q}}^T \otimes (G^T(\boldsymbol{q}) \otimes I \otimes G(\boldsymbol{q})) \otimes \dot{\boldsymbol{q}} \end{aligned} \tag{2.33}$$

or

$$\begin{aligned} L_{rot} &= 2(-G(\boldsymbol{q})\otimes\dot{\boldsymbol{q}})^T\otimes I\otimes(-G(\boldsymbol{q})\otimes\dot{\boldsymbol{q}}) \\ &= -2\boldsymbol{q}^T\otimes(G^T(\dot{\boldsymbol{q}})\otimes I\otimes G(\dot{\boldsymbol{q}}))\otimes\boldsymbol{q} \end{aligned} \tag{2.34}$$

where

$$G(\boldsymbol{q}) = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}$$

which are two equivalent equations that will help us to get the Euler-Lagrange equation. Thus, the expression $\frac{\partial L_{rot}}{\partial\vec{\Omega}}$ in (2.32) can be rewritten using (2.34) as

$$\begin{aligned} \frac{\partial L_{rot}}{\partial\boldsymbol{q}} &= -4(G^T(\dot{\boldsymbol{q}})\otimes I\otimes G(\dot{\boldsymbol{q}}))\otimes\boldsymbol{q} \\ &= -2G^T(\dot{\boldsymbol{q}})\otimes I\otimes(2G(\dot{\boldsymbol{q}})\otimes\boldsymbol{q}) \\ &= -2G^T(\dot{\boldsymbol{q}})\otimes I\vec{\Omega}, \end{aligned} \tag{2.35}$$

then, developing $\frac{\partial L_{rot}}{\partial\dot{\vec{\Omega}}}$ in (2.32) using (2.33)

$$\begin{aligned} \frac{\partial L_{rot}}{\partial\dot{\boldsymbol{q}}} &= 4(G^T(\boldsymbol{q})\otimes I\otimes G(\boldsymbol{q}))\otimes\dot{\boldsymbol{q}} \\ &= 2G^T(\boldsymbol{q})\otimes I\otimes 2G(\boldsymbol{q})\otimes\dot{\boldsymbol{q}} \\ &= 2G^T(\boldsymbol{q})\otimes I\vec{\Omega}. \end{aligned} \tag{2.36}$$

Finally, the temporal derivative of (2.36) is expressed as

$$\frac{d}{dt}\frac{\partial L_{rot}}{\partial\dot{\boldsymbol{q}}} = 2G^T(\dot{\boldsymbol{q}})\otimes I\vec{\Omega} + 2G^T(\boldsymbol{q})\otimes I\dot{\vec{\Omega}}. \tag{2.37}$$

For getting into the final dynamic model of the quadrotor, the analysis of the generalized torques must be carried out. Considering the D'Alambert's principle but on its rotational form: the virtual work of a rigid body around an angle γ of an axis ϖ and a torque $\vec{T}$ can be described as follows

$$\delta W = (\varpi\cdot\vec{T})\delta_\gamma \tag{2.38}$$

Notice that if $\|\varpi\| = 1$, thus, it can be referred as a quaternion and the infinitesimal rotation can be represented as

$$\boldsymbol{q} + \delta_q = \boldsymbol{q}\otimes\boldsymbol{q}_\delta \tag{2.39}$$

subject to $|\boldsymbol{q}| = 1$, $|\boldsymbol{q}_\delta|$ and $|\delta_q| << 1$. Multiplying the conjugate $\boldsymbol{q}^*$ in both sides of (2.39), then

$$\begin{aligned} \boldsymbol{q}^*\otimes\boldsymbol{q} + \boldsymbol{q}^*\otimes\delta_q &= \boldsymbol{q}^*\otimes\boldsymbol{q}\otimes\boldsymbol{q}_\delta \\ [1\ \ 0] + \boldsymbol{q}^*\otimes\delta_q &= \boldsymbol{q}_\delta. \end{aligned} \tag{2.40}$$

Using the definition of the temporal derivative of a quaternion as stated in (2.25) and if infinitesimal change are considered, δ_q can be expressed as

$$\delta_q = \left[\cos\left(\frac{\delta_\gamma}{2}\right) \;\; \varpi \sin\left(\frac{\delta_\gamma}{2}\right)\right]^T \tag{2.41}$$

and the following approximation is valid

$$\boldsymbol{q}_\delta = \left[1 \;\; \varpi\frac{\delta_\gamma}{2}\right]. \tag{2.42}$$

Then using (2.42) in (2.40)

$$[1 \;\; 0] + \boldsymbol{q}^* \otimes \delta_q = \left[1 \;\; \varpi\frac{\delta_\gamma}{2}\right] \tag{2.43}$$

which implies that $\boldsymbol{q}^* \otimes \delta_q$ is associated with the complex component of $\boldsymbol{q}_\delta$ and therefore the following expression is considered valid

$$\bar{o}\frac{\delta_\gamma}{2} = \boldsymbol{q}^* \otimes \delta_q. \tag{2.44}$$

The virtual work can be rewritten using (2.44) in (2.39)

$$\delta W = 2(\boldsymbol{q}^* \otimes \delta_q) \otimes \vec{\tau} \tag{2.45}$$

According to the quaternion algebra [150], the expression (2.45) can be rewritten as

$$\delta W = 2\vec{\tau}^\top (G(\boldsymbol{q}) \otimes \delta_q) \tag{2.46}$$

Thus, the virtual work is described as follows

$$\delta W = (2G^T(\boldsymbol{q}) \otimes \vec{\tau}) \otimes \delta_q \tag{2.47}$$

Therefore, the generalized torques are

$$\vec{T} = 2G(\boldsymbol{q}) \otimes \vec{\tau} \tag{2.48}$$

Using (2.35), (2.36) and (2.48) in (2.32) and multiplying by $G(\boldsymbol{q})$

$$4G(\dot{\boldsymbol{q}}) \otimes G^T(\dot{\boldsymbol{q}}) \otimes I\vec{\Omega} + 2G(\boldsymbol{q}) \otimes G^T(\boldsymbol{q}) \otimes I\dot{\vec{\Omega}} = 2G(\boldsymbol{q}) \otimes G^T(\boldsymbol{q}) \otimes \vec{\tau} + \lambda \otimes G(\boldsymbol{q}) \otimes \boldsymbol{q} \tag{2.49}$$

Knowing that $G(\dot{\boldsymbol{q}}) \otimes G^T(\dot{\boldsymbol{q}}) = 1$, $G(\boldsymbol{q}) \otimes G^T(\boldsymbol{q}) = 1$ and that $G(\boldsymbol{q}) \otimes \boldsymbol{q} = 0$, then

$$2I\dot{\vec{\Omega}} + 2I\vec{\Omega} = 2\vec{\tau} \tag{2.50}$$

Observe that the expression (2.50) has the form of the Newton's equation that describes the dynamics of a rigid body subject an external torques.

Thus, the rotational dynamic model of the quadrotor can be rewritten as

$$\begin{aligned} \dot{\boldsymbol{q}} &= -\frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} \\ I\dot{\vec{\Omega}} &= -\vec{\Omega} \times I\vec{\Omega} + \vec{\tau} \end{aligned} \tag{2.51}$$

Therefore, the whole quadcopter dynamic model can be described using (2.30) and (2.51) as

$$\begin{bmatrix} \dot{\vec{\xi}} \\ \ddot{\vec{\xi}} \\ \dot{\boldsymbol{q}} \\ \dot{\vec{\Omega}} \end{bmatrix} = \begin{bmatrix} \dot{\vec{\xi}} \\ \boldsymbol{q} \otimes \frac{\vec{F}_{th}}{m} \otimes \boldsymbol{q}^* + \vec{g} \\ -\frac{1}{2}\boldsymbol{q} \otimes \vec{\Omega} \\ I^{-1}\left(-\vec{\Omega} \times I\vec{\Omega} + \vec{\tau}\right) \end{bmatrix} \tag{2.52}$$

2.3.2 Decoupling the vehicle dynamics

As can be verified in (2.52), the dynamics of the quadcopter are completely coupled by means of the orientation of $\vec{F}_{th}$ depending on the vehicle's attitude $\boldsymbol{q}$. Nevertheless, using an appropriate approach and some properties of unit quaternions, the quadcopter can be easily controlled despite its underactuated nature.

Let us assume that (2.51) is stabilized using a control action $\vec{\tau}$, then the quaternion attitude will converge to $\boldsymbol{q}_0 = 1 + [0\ 0\ 0]^T$ while the axis-angle orientation δ and its angular velocity $\vec{\Omega}$ will converge to zero. Given a desired attitude trajectory defined by a desired quaternion $\boldsymbol{q_d}$ and its angular velocity $\vec{\Omega}_d$, (2.51) can be defined in terms of the quaternion error $\boldsymbol{q_e} \triangleq \boldsymbol{q_d}^* \otimes \boldsymbol{q}$ as

$$\begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{e}} \\ \dot{\vec{\Omega}}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{q_e} \otimes \vec{\Omega}_e \\ I^{-1}\left(-\vec{\Omega}_e \times I\vec{\Omega}_e + \vec{\tau}\right) \end{bmatrix}, \tag{2.53}$$

if $\vec{\tau}$ is correctly designed in terms of the attitude error, then, $\boldsymbol{q_e} \to \boldsymbol{q}_0$ implying that $\boldsymbol{q} \to \boldsymbol{q_d}$.

From (2.30) and considering the desired force (2.29), $\vec{F}_\xi = \boldsymbol{q} \otimes \vec{F}_{th} \otimes \boldsymbol{q}^*$ can be easily designed such that $\vec{\xi}$ and $\dot{\vec{\xi}}$ converge to zero. If a position error is defined as $\vec{\xi}_e = \vec{\xi} - \vec{\xi}_d$, where $\vec{\xi}_d$ represents a desired position for the vehicle, then the translational dynamics error can be written as

$$\begin{bmatrix} \dot{\vec{\xi}}_e \\ \ddot{\vec{\xi}}_e \end{bmatrix} = \begin{bmatrix} \dot{\vec{\xi}}_e \\ \frac{\vec{F}_\xi}{m} - \vec{g} \end{bmatrix} \tag{2.54}$$

Consequently, if an adequate controller is designed for $\vec{F}_\xi$, the error position will converge to zero, meaning that the quadcopter can be stabilized at any desired position.

From (2.54), it yields that the translational model can be seen as a virtual fully actuated system where $\vec{F}_\xi$ can be designed to point at any direction. Let us define a desired force vector $\vec{U} \in \mathbb{R}^3$ with respect to the frame $\mathscr{I}$, which stabilizes the system (2.54). Given the direction and magnitude of such force vector, the attitude can be controlled using $\vec{\tau}$ such that the direction of $\vec{F}_\xi$ is aligned with $\vec{U}$, orientating the quadcopter thrust in the required direction to control the translational dynamics. Therefore, the desired quaternion $\boldsymbol{q_d}$ is derived from the shortest rotation between these both vectors.

Recalling the Euler-Rodrigues formula from (2.19), $\boldsymbol{q_d}$ is defined as

$$\boldsymbol{q_d} = \exp^{\frac{1}{2}\varsigma\vec{n}} = \cos(\varsigma/2) + \vec{n}\sin(\varsigma/2), \tag{2.55}$$

where $\vec{n}$ and ς denote the axis and the angle of shortest rotation between $\vec{F}_{th}$ and $\vec{U}$, respectively. Defining $\vec{v}$ and $\vec{\mu}$ as normalized vectors of $\vec{F}_{th}$ and $\vec{U}$ where, notice that $\vec{v} = [0\ 0\ 1]^T$ is constant. The cross and the scalar products between these two vectors are defined as, see Figure 2.4

$$\vec{\mu} \times \vec{v} = S(\vec{\mu})\vec{v} = \vec{\mu}\,\vec{v}\sin(\varsigma)\vec{n}, \tag{2.56}$$

$$\vec{\mu} \cdot \vec{v} = \vec{\mu}\,\vec{v}\cos(\varsigma). \tag{2.57}$$

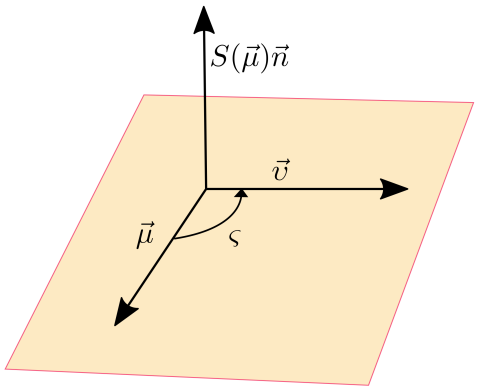


Figure 2.4 – Illustration of the Cross product of two vectors

Remark from (2.55), the Euler-Rodrigues formula uses the half angle, thus the well-known trigonometry half angles formulas are required

$$\cos\left(\frac{\varsigma}{2}\right) = \pm\sqrt{\frac{1+\cos(\varsigma)}{2}} \qquad \sin\left(\frac{\varsigma}{2}\right) = \pm\sqrt{\frac{1-\cos(\varsigma)}{2}} \tag{2.58}$$

Thus, the desired attitude that aligns the thrust vector to the control direction can obtained from (2.55) using (2.56), (2.57) and (2.58) as

$$\boldsymbol{q_t} = \pm \begin{bmatrix} \sqrt{\frac{1+\vec{\mu}\,\vec{v}}{2}} \\ \sqrt{\frac{1-\vec{\mu}\,\vec{v}}{2}} S(\vec{\mu})\vec{v} \end{bmatrix} \tag{2.59}$$

where

$$S(\vec{\mu}) = \begin{bmatrix} 0 & -\mu_3 & \mu_2 \\ \mu_3 & 0 & -\mu_1 \\ -\mu_2 & \mu_1 & 0 \end{bmatrix}$$

Notice that since $\vec{v}$ is always aligned with the vertical axis of the body frame $\mathscr{B}$ and the direction of (2.59) is defined by a cross product, then $\boldsymbol{q}_t$ rotates the vehicle in the xy plane. An additional rotation around the $z-$axis can be added by introducing

$$\boldsymbol{q_d} \triangleq \boldsymbol{q_t} \otimes \boldsymbol{q_z}, \tag{2.60}$$

where $\boldsymbol{q_z}$ represent the desired rotation over the $z-$axis.

2.4 Modeling approaches conclusions

The aim of this chapter was to give an introduction to the main mathematical concepts for getting into the dynamic modeling of a quadrotor vehicle, using one of the three existing approaches in literature: Newton-Euler, Euler-Lagrange and quaternion formalism. Even this kind of UAVs are inherently underactuated, an approach using the quaternion formalism was introduced such that its dynamic equations can be analyzed and treated as a virtual fully actuated system.

Most robotics research works, particularly in UAVs, have been relied on Euler angles convention. However, the presence of important non-linearities, undesired effects, e.g. gimball-lock, and an inherent complexity when multiple rotations and transltations are present, may hamper the development in the performance of the control algorithms and its applications. Nevertheless, although the quaternion formalism can seem to be less intuitive and difficult to conceptualize, their application can really simplify dynamic models, and help in the design of better controllers.

Chapter 3 Part III

3 Control approaches for aerial vehicles

Controlling a quadcotper vehicle has been extensively studied in the past decade with the recent increase on the power computation for embedded systems. These systems are now able to perform the computations needed for a variety of control techniques, with lower cost of sensors and actuators. These types of control algorithms are applied to the position and the attitude ofUAVs. This chapter is devoted to present a brief overview evaluation of popular control algorithms for Multi-rotor Aerial Systems (MAS).

The main objective of this chapter is to provide a unified and accessible analysis, placing the classical and quaternion models of the quadcopter as well the study of its control methods into a proper context. In addition to provide the basis for beginner users working in aerial vehicles, this chapter contributes in presenting a comprehensive analysis of the implementation for the nonlinear and linear backstepping, nested saturation, virtual fully actuated controllers as well the sliding mode control and the quaternion-based nonlinear backtepping approaches. These techniques are selected and compared to evaluate the performance of the aircraft by simulations and experimental studies.

The content of this chapter is organized as follows: In section 3.1, a brief overview for controlling the attitude and translation system of the quadrotor is presented. Four control laws are developed with their respective analysis of stability. Then, a quaternion-based backstepping control is addressed in section 3.2 to demonstrate the advantages of using this kind of controller on aerial vehicles. Finally in section 3.3, the finite-time convergence is discussed.

3.1 Comparison of stabilization and tracking control algorithms

In this section, we present a comparison between the well-known control algorithms relying on the Euler Angles, to solve the stabilization and tracking control problems. The aim of this section is to introduce the main control algorithms used in the literature for controlling a quadrotor vehicle and show their properties and capabilities.

Typically in the literature, when controlling a quadrotor vehicle, it is assumed the $z-$axis can be controlled by some linear or nonlinear controller using specifically F_{Th}. Usually, this controller is based on a feedback linearization technique as presented in [151, 152, 153]. This *classical* procedure is often used because the idea is to guarantee the vehicle keeps at hover, and later control their displacement in the plane x or y making an under-actuated subsystem.

Control of altitude and yaw angle

Following aforementioned ideas, the controller for the $z-$axis is often proposed as

$$u_1 = \frac{m}{\cos\theta\cos\phi}(g + r_1) \tag{3.1}$$

where r_1 is always designed to achieve $z \to z_d$, for instance $r_1 = -k_1(z - z_d) - k_2\dot{z}$, where k_1, k_2 represent positive constants and z_d denotes the desired altitude. Then rewriting (2.12) using (3.1) and assuming $\cos\theta\cos\phi \neq 0$, that is, $\theta, \phi \in (-\pi/2, \pi/2)$, it yields

$$m\ddot{x} = -(r_1 + mg)\frac{\tan\theta}{\cos\phi} \tag{3.2}$$

$$m\ddot{y} = (r_1 + mg)\tan\phi \tag{3.3}$$

$$\ddot{z} = -k_2\dot{z} - k_1(z - z_d) \tag{3.4}$$

Note from (3.4), $\exists$ a time T large enough such that for $t > T$, the altitude error $(z - z_d) \to 0$ such that $z \approx z_d$ can be considered the equilibrium state of (3.4). Thus we can rewrite (3.2) and (3.3) as follows

$$\begin{aligned} \ddot{x} &\approx -g\frac{\tan\theta}{\cos\phi} \\ \ddot{y} &\approx g\tan\phi \end{aligned} \tag{3.5}$$

For simplifying the analysis, let us consider $J = I_{3\times3}$. Thus, system (2.12) can be expressed using (3.1) and (3.5) as

$$\begin{aligned} \ddot{x} &\approx -g\frac{\tan\theta}{\cos\phi} & \ddot{\theta} &\approx \tilde{\tau}_\theta \\ \ddot{y} &\approx g\tan\phi & \ddot{\phi} &\approx \tilde{\tau}_\phi \\ & & \ddot{\psi} &\approx \tilde{\tau}_\psi \end{aligned} \tag{3.6}$$

where $\tilde{\tau}_\psi = -a_{\psi_1}\dot{\psi} - a_{\psi_2}(\psi - \psi_d)$ is the proposed control torque to stabilize the yaw angle and $\tilde{\tau}_\phi$, $\tilde{\tau}_\theta$ will be defined later. The control gains a_{ψ_1}, a_{ψ_2} are positive constants chosen to ensure stable, well-damped response of the quadrotor. From (3.6) it follows that if ψ_d and z_d are constants, then ψ and z converge. Therefore, $\dot{\psi}$ and $\ddot{\psi} \to 0$, which implies that $\psi \to \psi_d$.

3.1.1 Nonlinear Backstepping algorithm

Backstepping is suitable for strict-feedback systems that are also known as *lower triangular* and it does not require that the resulting input-output dynamics be linear. The main idea is to use some of the state variables as “virtual controls" or “pseudo controls", and depending on the dynamics of each state, design intermediate control laws. The Backstepping design is a recursive procedure where a Lyapunov function is derived for the entire system (3.6).

Control of lateral position and roll angle

Define the error e_1 as $e_1 = y_1 - y_1^d$, where y_1^d is the reference. Let us propose the following positive function $V_1 = \frac{k_{y_1}}{2}e_1^2$, with $k_{y_1} > 0$ denoting a constant, then taking the derivative with respect to time and proposing $y_2^v = y_2^d - e_1$ as a virtual control input, it follows that

$$\dot{V}_1 = k_{y_1}e_1(y_2 - y_2^v - e_1) = -k_{y_1}e_1^2 + k_{y_1}e_1(y_2 - y_2^v)$$

where $\dot{y}_1^d = y_2^d$. Define the error $e_2 = y_2 - y_2^v$ thus, the previous yields $\dot{V}_1 = -k_{y_1}e_1^2 + k_{y_1}e_1e_2$. Propose the second positive function $V_2 = \frac{k_{y_2}}{2}e_2^2$ with $k_{y_2} > 0$ constant. Taking its derivative with respect to time and proposing $\delta_1^v = -\dot{y}_2^v + \frac{k_{y_1}}{k_{y_2}}e_1 + e_2$ representing a second virtual control input, implies that

$$\dot{V}_2 = -k_{y_2}e_2^2 - k_{y_1}e_1e_2 + k_{y_2}e_2(\delta_1^v - g\tan\phi_1)$$

Defining the error $e_3 = \delta_1^v - g\tan\phi_1$, $\dot{V}_2$ yields

$$\dot{V}_2 = -k_{y_2}e_2^2 - k_{y_1}e_1e_2 + k_{y_2}e_2e_3$$

Propose the third positive function $V_3 = \frac{k_{y_3}}{2}e_3^2$, with $k_{y_3} > 0$ constant. Differentiating V_3 and defining $\delta_2^v = \dot{\delta}_1^v + \frac{k_{y_2}}{k_{y_3}}e_2 + e_3$, it follows that

$$\dot{V}_3 = k_{y_3}e_3^2 - k_{y_2}e_3e_2 + k_{y_3}e_3(\delta_2^v - g(1+\tan^2\phi_1)\phi_2)$$

Defining the error $e_4 = \delta_2^v - g(1+\tan^2\phi_1)\phi_2$, then the above yields

$$\dot{V}_3 = -k_{y_3}e_3^2 - k_{y_2}e_3e_2 + k_{y_3}e_3e_4$$

Proposing $V_4 = \frac{k_{y_4}}{2} e_4^2$ as positive function, with $k_{y_4} > 0$ defining a constant, and computing its derivative, it appears that

$$\dot{V}_4 = k_{y_4} e_4 (\dot{\delta}_2^v - g(1+\tan^2\phi_1)(\tilde{\tau}_\phi + 2\phi_2^2 \tan\phi_1))$$

Proposing a control law given by

$$\tilde{\tau}_\phi = \frac{1}{g(1+\tan^2\phi_1)} (\dot{\delta}_2^v + \frac{k_{y_3}}{k_{y_4}} e_3 + e_4) - 2\phi_2^2 \tan\phi_1 \tag{3.7}$$

implies that $\dot{V}_4 = -k_{y_4} e_4^2 - k_{y_3} e_4 e_3$.

Finally, define the following candidate Lyapunov function with the form $V_T = V_1 + V_2 + V_3 + V_4$. Therefore

$$\dot{V}_T = -k_{y_1} e_1^2 - k_{y_2} e_2^2 - k_{y_3} e_3^2 - k_{y_4} e_4^2 \leq 0.$$

Expressing (3.7) with respect to the state variables, it follows that

$$\begin{aligned} \tilde{\tau}_\phi = \frac{1}{g(1+\tan^2\phi_1)} \Big(& \dddot{y}_2^d - 4\ddot{y}_2^d - \bar{k}_{y_1} \dot{y}_2^d + \bar{k}_{y_2}(y_2 - y_2^d) + \bar{k}_{y_3}(y_1 - y_1^d) - \bar{k}_{y_4} g \tan\phi_1 \\ & -4g(1+\tan^2\phi_1)\phi_2 - 2\phi_2^2\phi_1 \Big) \end{aligned} \tag{3.8}$$

where

$$\begin{aligned} & \bar{k}_{y_1} = \bar{k}_{y_4} = k_\phi + 6, \qquad && \bar{k}_{y_2} = 2k_\phi + 4 \\ & \bar{k}_{y_3} = k_\phi + \frac{k_{y_3} k_{y_1}}{k_{y_4} k_{y_2}} + 1, \qquad && k_\phi = \frac{k_{y_1}}{k_{y_2}} + \frac{k_{y_2}}{k_{y_3}} + \frac{k_{y_3}}{k_{y_4}} \end{aligned} \tag{3.9}$$

Control of forward position and pitch angle

From (3.6) and (3.8), we obtain $\phi \to 0$, then from (3.5) and (3.6) we get

$$\begin{aligned} \ddot{x} &\approx -g\tan\theta \\ \ddot{\theta} &= \tilde{\tau}_\theta \end{aligned} \tag{3.10}$$

Using a procedure similar to the one proposed for the roll control, we obtain

$$\begin{aligned} \tilde{\tau}_\theta = \frac{1}{g(1+\tan^2\theta_1)} \Big(& \dddot{x}_2^d - 4\ddot{x}_2^d - \bar{k}_{x_1} \dot{x}_2^d + \bar{k}_{x_2}(x_2 - x_2^d) + \bar{k}_{x_3}(x_1 - x_1^d) - \bar{k}_{x_4} g \tan\theta_1 \\ & -4g(1+\tan^2\theta_1)\theta_2 - 2\theta_2^2\theta_1 \Big) \end{aligned} \tag{3.11}$$

where

$$\begin{aligned} \bar{k}_{x_1} &= \bar{k}_{x_4} = k_\theta + 6, & \bar{k}_{x_2} &= 2k_\theta + 4 \\ \bar{k}_{x_3} &= k_\theta + \frac{k_{x_3} k_{x_1}}{k_{x_4} k_{x_2}} + 1, & k_\theta &= \frac{k_{x_1}}{k_{x_2}} + \frac{k_{x_2}}{k_{x_3}} + \frac{k_{x_3}}{k_{x_4}} \end{aligned} \tag{3.12}$$

3.1.2 Control algorithm based on nested saturation

The goal for controllers based on saturation functions is to impose a bound in the control input. The control algorithm is conceived from a linear systems (chain of integrators) nevertheless, it was also demonstrated that it can be applied for nonlinear systems. Their procedure, in contrast to the backstepping approach, starts in the last state (where the control input is located) until the first one. The stability analysis is assured using a series of positive functions.

The first methodology was proposed in [154] designing the controller with nested saturation functions for linear systems and applied to nonlinear systems. Nevertheless after analysis, it can be observed that the controller structure using saturation functions imposes bounds on each state of the system, such that, some inequalities can be applied, for example $\tan\theta \approx \theta$ and $\cos\phi \approx 1$, see [155, 147]. From this assumption, the longitudinal and lateral parts of (3.6) can be rewritten as follows

$$\begin{aligned} \dot{x}_1 &= x_2 & \dot{y}_1 &= y_2 \\ \dot{x}_2 &\approx -g\theta_1 & \dot{y}_2 &\approx g\phi_1 \\ \dot{\theta}_1 &= \theta_2 & \dot{\phi}_1 &= \phi_2 \\ \dot{\theta}_2 &\approx \tilde{\tau}_\theta & \dot{\phi}_2 &\approx \tilde{\tau}_\phi \end{aligned} \tag{3.13}$$

Roll control (ϕ, y)

A controller stabilizing the lateral system of (3.13) can be denoted by

$$\tilde{\tau}_\phi = -\sigma_a(\phi_2 + \sigma_b(\cdot)) \tag{3.14}$$

where σ_i defines a generalized saturation function such that $|\sigma_i(\cdot)| \leq i$ for positive constant $i : a, b, c, d$. The argument in $\sigma_b(\cdot)$ will be defined later to assure convergence of the states. Let us define a positive function $V_1 = \frac{1}{2}\phi_2^2$, then its derivative is defined by

$$\dot{V}_1 = \phi_2\dot{\phi}_2 = \phi_2\tilde{\tau}_\phi = -\phi_2\sigma_a(\phi_2 + \sigma_b(\cdot)) \tag{3.15}$$

if $\left|\phi_2\right| \geq b \Rightarrow \dot{V}_1 \leq 0$. Then, $\exists\ t_1$ such that for $t \geq t_1\ \left|\phi_2(t)\right| \leq b$, implying that $\left|\phi_2 + \sigma_b(\cdot)\right| \leq 2b$. Choosing $a \geq 2b$, then (3.14) can be rewritten $\forall\ t > t_1$ as

$$\tilde{\tau}_\phi = -\phi_2 - \sigma_b(\cdot). \tag{3.16}$$

Define $\nu_1 = \phi_1 + \phi_2$, then, $\dot{\nu}_1 = -\sigma_b(\cdot)$. Propose a positive function $V_2 = \frac{1}{2}\nu_1^2$ and imposing $\sigma_b(\cdot) = \sigma_b(\nu_1 + \sigma_c(\cdot))$ then, its derivative is determined by

$$\dot{V}_2 = \nu_1\dot{\nu}_1 = -\nu_1\sigma_b(\nu_1 + \sigma_c(\cdot)) \tag{3.17}$$

if $|\nu_1| \geq c \Rightarrow \dot{V}_2 \leq 0$. Then, $\exists\ t_2 \geq t_1$ such that for $t \geq t_2$ $|\nu_1(t)| \leq c$ and $|\nu_1 + \sigma_c(\cdot)| \leq 2c$. Choosing $b \geq 2c$, then (3.16) can be rewritten $\forall\ t > t_2$ as

$$\tilde{\tau}_\phi = -\phi_2 - \nu_1 - \sigma_c(\cdot). \tag{3.18}$$

Define $\nu_2 = \nu_1 + \phi_1 - y_2/g = \phi_2 + 2\phi_1 - y_2/g$, thus $\dot{\nu}_2 = -\sigma_c(\cdot)$. Propose the positive function $V_3 = \frac{1}{2}\nu_2^2$ with $\sigma_c(\cdot) = \sigma_c(\nu_2 + \sigma_d(\cdot))$, then its derivative is described by

$$\dot{V}_3 = \nu_2\dot{\nu}_2 = -\nu_2\sigma_c(\nu_2 + \sigma_d(\cdot)) \tag{3.19}$$

if $|\nu_2| \geq d \Rightarrow \dot{V}_3 \leq 0$. Then, $\exists\ t_3 \geq t_2$ such that $\forall\, t \geq t_3$ $|\nu_2(t)| \leq d$ and $|\nu_2 + \sigma_d(\cdot)| \leq 2d$. Choosing $c \geq 2d$, then (3.18) can be rewritten $\forall\ t > t_3$ as

$$\tilde{\tau}_\phi = -\phi_2 - \nu_1 - \nu_2 - \sigma_d(\cdot). \tag{3.20}$$

Define $\nu_3 = \phi_2 + 3\phi_1 + 3\frac{y_2}{g} + \frac{y_1}{g}$ then $\dot{\nu}_3 = -\sigma_d(\cdot)$. Propose $V_4 = \frac{1}{2}\nu_3^2$ and $\sigma_d(\cdot) = \sigma_d(\nu_3)$, taking the derivative of V_4, it follows that

$$\dot{V}_4 = \nu_3\dot{\nu}_3 = -\nu_3\sigma_d(\nu_3) \leq 0 \tag{3.21}$$

The previous implies that $\nu_3 \to 0$, then from (3.19) it follows that $\nu_2 \to 0$. From (3.17) implies that $\nu_1 \to 0$, similarly from (3.15), $\phi_2 \to 0$. From definition of ν_1 it follows that $\phi_1 \to 0$. From definition of ν_2 this implies that $y_2 \to 0$. And finally, from definition of ν_3 it can be deduced that $y_1 \to 0$.

Rewriting $\tilde{\tau}_\phi$, it yields

$$\tilde{\tau}_\phi = -\sigma_a\left(\phi_2 + \sigma_b\left(\phi_1 + \phi_2 + \sigma_c\left(\phi_2 + 2\phi_1 + y_2\sigma_d\left(\phi_2 + 3\phi_1 + 3\frac{y_2}{g} + \frac{y_1}{g}\right)\right)\right)\right) \tag{3.22}$$

Pitch control (θ, x)

As before, we assume that the control strategy will insure a very small bound on $|\,\theta\,|$ such that $\tan(\theta) \approx \theta$. Therefore, using a procedure similar to the one proposed for the roll control, it is possible to express the pitch control as follows

$$\tilde{\tau}_\theta = -\sigma_e\left(\theta_2 + \sigma_f\left(\theta_1 + \theta_2 + \sigma_g\left(\theta_2 + 2\theta_1 + x_2 + \sigma_h\left(\theta_2 + 3\theta_1 - 3\frac{x_2}{g} - \frac{x_1}{g}\right)\right)\right)\right) \tag{3.23}$$

3.1.3 Linear backstepping

This controller is conceived using the same procedure proposed in the nonlinear backstepping methodology, but instead of using the nonlinear system, a linear system is required. This methodology is useful for beginners with the goal to better understand its procedure. The obtained control law is quite different to the one obtained with the nonlinear system. For our study, it is assumed that the vehicle is moving with small angles, therefore the nonlinear system (3.10) can be represented as (3.6).

Forward position control

Rewriting first equation in (3.6)

$$\dot{x}_1 = x_2 \tag{3.24}$$

propose $V_1 = \frac{1}{2}x_1^2$, then $\dot{V}_1 = x_1 x_2$. if $x_2 \rightarrow -k_1 x_1$ with k_1 is a positive constant, then $\dot{V}_1 = -k_1 x_1^2$. Define the following error with the form

$$\zeta_2 = x_2 - x_2^v \tag{3.25}$$

with $x_2^v = -k_1 x_1 = \alpha_1$ defining the first virtual input. Rewriting (3.24) and the second equation of (3.6) with (3.25) and using $x_1 = \zeta_1$

$$\begin{aligned} \dot{\zeta}_1 &= \zeta_2 + \alpha_1 \\ \dot{\zeta}_2 &= -g\theta_1 - \dot{\alpha_1} \end{aligned} \tag{3.26}$$

Define a positive definite function $V_2 = V_1 + \frac{\zeta_2}{2}^2$, then

$$\dot{V}_2 = \dot{V}_1 + \zeta_2\dot{\zeta}_2 = \dot{V}_1 + \zeta_2(-g\theta_1 - \dot{\alpha_1}) \tag{3.27}$$

if $(-g\theta_1 - \dot{\alpha_1}) \rightarrow -k_2\zeta_2$ with k_2 is a positive constant, then, $\dot{V}_2 = \dot{V}_1 - k_2\zeta_2^2$. Define the error

$$\zeta_3 = -g\theta_1 - \alpha_2 \tag{3.28}$$

with $\alpha_2 = k_2\zeta_2 - \dot{\alpha}_1$ denoting the second virtual input. Rewriting (3.26) with the previous equation, it follows that

$$\begin{aligned} \dot{\zeta}_1 &= \zeta_2 + \alpha_1 \\ \dot{\zeta}_2 &= -g\theta_1 - \dot{\alpha}_1 \\ \dot{\zeta}_3 &= -g\theta_2 - \dot{\alpha}_2 \end{aligned} \tag{3.29}$$

Define the positive definite function $V_3 = V_2 + \frac{\zeta_3^2}{2}$, then

$$\dot{V}_3 = \dot{V}_2 + \zeta_3\dot{\zeta}_3 = \dot{V}_2 + \zeta_3(-g\theta_2 - \dot{\alpha}_2) \tag{3.30}$$

if $(-g\theta_2 - \dot{\alpha}_2) \rightarrow -k_3\zeta_3$, with k_3 is a positive constant, then $\dot{V}_3 = \dot{V}_2 - k_3\zeta_3^2$. Define the error

$$\zeta_4 = -g\theta_2 - \alpha_3 \tag{3.31}$$

where $\alpha_3 = (g\theta_2)^{\nu} = k_3\zeta_3 - \dot{\alpha}_2$ is the third virtual input. Rewriting (3.29) with $\dot{\zeta}_4$, it follows that

$$\begin{aligned}
\dot{\zeta}_1 &= \zeta_2 + \alpha_1 \\
\dot{\zeta}_2 &= -g\theta_1 - \dot{\alpha}_1 \\
\dot{\zeta}_3 &= -g\theta_2 - \dot{\alpha}_2 \\
\dot{\zeta}_4 &= -g u_2 - \dot{\alpha}_3
\end{aligned} \tag{3.32}$$

Propose the candidate Lyapunov function $V_4 = V_3 + \frac{\zeta_4^2}{2}$, then

$$\dot{V}_4 = \dot{V}_3 + \zeta_4\dot{\zeta}_4 = \dot{V}_3 + \zeta_4(-g u_2 - \dot{\alpha}_3) \tag{3.33}$$

Propose the control input

$$\tilde{\tau}_\theta = \frac{1}{g}(k_4\zeta_4 - \dot{\alpha}_3) \tag{3.34}$$

with $k_4 > 0$ is a constant. Then

$$\dot{V}_4 = \dot{V}_3 - k_4\zeta_4^2 = -k_1\zeta_1^2 - k_2\zeta_2^2 - k_3\zeta_3^2 - k_4\zeta_4^2 < 0 \tag{3.35}$$

The above implies system (3.32) goes to zero implying that (3.6) is globally asymptotically stable. Rewriting (3.34) with respect to the state variables, it follows that

$$\tilde{\tau}_\theta = -\frac{\bar{k}_1}{g}x_1 - \frac{\bar{k}_2}{g}x_2 + \bar{k}_3\theta_1 + \bar{k}_4\theta_2 \tag{3.36}$$

where

$$\begin{aligned}
\bar{k}_1 &= k_1k_2k_3k_4 \\
\bar{k}_2 &= k_1k_2(k_3 + k_4) + k_3k_4(k_1 + k_2) \\
\bar{k}_3 &= k_1(k_2 + k_3 + k_4) + k_2(k_3 + k_4) + k_3k_4 \\
\bar{k}_4 &= k_1 + k_2 + k_3 + k_4
\end{aligned} \tag{3.37}$$

Lateral position control

For controlling the lateral position of the system (3.6), and using a similar procedure as the one developed for the pitch control, the backstepping roll control input can be obtained as

$$\tilde{\tau}_\phi = \frac{\bar{k}_{\phi_1}}{g}y_1 + \frac{\bar{k}_{\phi_2}}{g}y_2 + \bar{k}_{\phi_3}\phi_1 + \bar{k}_{\phi_4}\phi_2 \tag{3.38}$$

3.1.4 Fully actuated approach

In this approach, the goal is to transform the underactuated system into a virtual fully actuated system. This can be done by imposing a desired attitude R_d that will be related with the virtual control input $\vec{U}$. This allows us to use control inputs τ_ϕ, τ_θ to only control the attitude system, i.e., $R \to R_d$. Motivated by the works developed by [156, 157], simple smooth bounded controllers that can easily be implemented in VTOL aircraft with parallel motors for tracking set-points and time-varying trajectories are proposed.

The methodology is explained as follows; propose a vector control input, $\vec{U}$, containing the virtual control laws to stabilize the translational states in the VTOL vehicle. These virtual control inputs will be related to the desired orientation matrix, R_d, of the vehicle and with its main control input, $\vec{F}_{th} = [0\ 0\ u_1]^T$, as follows

$$\vec{U} = u_1 R_d \vec{e}_z \tag{3.39}$$

In our case of study, the virtual control laws will be designed for the vertical, horizontal and longitudinal axis of the quadrotor, with the form $\vec{U} = [U_x, U_y, U_z]^T$, $\vec{e}_z = [0\ 0\ 1]^T$ and similarly $R_d \in \mathbb{R}^{3\times3}$, is defined as

$$R_d = \begin{bmatrix} S_{\theta_d}C_{\psi_d} & C_{\psi_d}S_{\theta_d}S_{\phi_d} - C_{\phi_d}S_{\psi_d} & S_{\theta_d}C_{\phi_d}C_{\psi_d} + S_{\phi_d}S_{\psi_d} \\ C_{\theta_d}S_{\psi_d} & C_{\phi_d}C_{\psi_d} + S_{\theta_d}S_{\phi_d}S_{\psi_d} & S_{\theta_d}C_{\phi_d}S_{\psi_d} - S_{\phi_d}C_{\psi_d} \\ -S_{\theta_d} & C_{\theta_d}S_{\phi_d} & C_{\theta_d}C_{\phi_d} \end{bmatrix}.$$

with θ_d, ψ_d ϕ_d representing the desired attitude. Next step is to rewrite the original translational dynamic system, in our case it is represented by equations (2.12). Thus the new system is described as

$$m\ddot{\vec{\xi}} = \vec{F}_{th} R - mg\vec{e}_z \tag{3.40}$$

where $\vec{\xi} = [x\ y\ z]^T$, $R \in \mathbb{R}^{3\times3}$ has the same form of R_d except that replacing the desired angles with $\phi\ \theta\ \psi$. Notice from (3.39) that, u_1 is the main control input. Therefore, using (3.39) into first equation of (3.40)

$$\begin{aligned} m\ddot{\vec{\xi}} &= u_1 R\vec{e}_z - mg\vec{e}_z + u_1 R_d\vec{e}_z - u_1 R_d\vec{e}_z \\ &= u_1(R - R_d)\vec{e}_z - mg\vec{e}_z + u_1 R_d\vec{e}_z \end{aligned}$$

Define $\vec{\omega} = u_1(R - R_d)\vec{e}_z$, thus

$$m\ddot{\vec{\xi}} = \vec{\omega} - mg\vec{e}_z + \vec{U}. \tag{3.41}$$

The desired orientation can be found using (3.39) and considering $\psi_d \doteq 0$, then

$$\theta_d = \arctan\left(\frac{U_x}{U_z}\right) \tag{3.42}$$

$$\phi_d = \arctan\left(\frac{U_y}{\sqrt{U_x^2 + U_z^2}}\right) \tag{3.43}$$

Notice from (3.39) that $\left\|\vec{U}\right\| = \|u_1 R_d \vec{e}_z\|$ gives an orthogonal vector as resultant of the right side of the equation, therefore

$$u_1 = \sqrt{U_x^2 + U_y^2 + U_z^2}. \tag{3.44}$$

Define the control objective

$$\lim_{t\longrightarrow\infty} \left\|\vec{\xi}(t) - \vec{\xi}_d(t)\right\| = 0. \tag{3.45}$$

where $\vec{\xi}_d = [x_d \ y_d \ z_d]^T$ are the longitudinal and lateral references. Denoting the tracking error as $\vec{p} = [\vec{p}_1 \ \vec{p}_2]^T$ such that

$$\vec{p}_1 = \vec{\xi} - \vec{\xi}_d \qquad \vec{p}_2 = \dot{\vec{\xi}} - \dot{\vec{\xi}}_d \tag{3.46}$$

Differentiating (3.46) and by means of (3.41)

$$\begin{aligned} \dot{\vec{p}}_1 &= \vec{p}_2 \\ \dot{\vec{p}}_2 &= \frac{\vec{\omega}}{m} - g\vec{e}_z + \frac{\vec{U}}{m} - \ddot{\vec{\xi}}_d. \end{aligned} \tag{3.47}$$

where $\ddot{\vec{\xi}}_d$ means the desired acceleration.

Control based on hyperbolic functions

Let us propose $\vec{U}$ in (3.47) as

$$\vec{U} = m(-\sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) - \sigma_2(K_2\vec{p}_2) + g\vec{e}_z + \ddot{\vec{\xi}}_d) \tag{3.48}$$

where $K_1, K_2 \in \mathbb{R}^{3\times 3}$ are diagonal positive matrices constant, σ_i means a saturation function with the form $\sigma(\varsigma) = \bar{\sigma}\tanh(\varsigma)$ and $\bar{\sigma}$ means a bounded constant. Hence, the bounded control $\vec{U}$ in (3.48) makes system (3.47) globally asymptotically stable.

Proof: First, to prove the asymptotic convergence of solution $\vec{p}(t)$ to the origin, observe from (3.39) and (3.41) that $\|\bar{\omega}\| \leq 2\bar{F}_{th}$. This implies that the closed-loop system using (3.47) and (3.48) satisfies the Lipschitz condition[1], and hence has a unique solution over $[0, T]$ with $T \geq 0$. Moreover, if T is bounded, then $\vec{p}(t)$ is bounded $\forall\ t \in [0, T]$. This proof is inspired in [159].

Furthermore, if $\lim_{t\longrightarrow\infty} \|\vec{\omega}(t)\| = 0$ then $\lim_{t\longrightarrow\infty} \left\|\vec{p}(t)\right\| = 0\ \forall\ \vec{p}(0)$, that is, for every ϵ_1, for every $\vec{p}(0)$, there exists ϵ_2, T_{ϵ_1}, $T_{\epsilon_2} > 0$ and $T_{\epsilon_1} \geq T_{\epsilon_2}$ such that

$$\|\vec{\omega}(t)\| < \epsilon_2,\ \forall\, t \geq T_{\epsilon_2} \Rightarrow \left\|\vec{p}(t)\right\| < \epsilon_1,\ \forall\, t \geq T_{\epsilon_1}. \tag{3.49}$$

Inspired by the work developed by [157], let us propose the following candidate Lyapunov function

$$V_1(\vec{p}) = \sum_{i=1}^{2} \int_0^{(k_{1_i} p_{1i} + k_{2_i} p_{2i})} \sigma_{1i}(\varsigma_i) d\varsigma_i + \frac{1}{2}\vec{p}_2^T K_1 \vec{p}_2 \tag{3.50}$$

[1] See [158], Theorem 3.1 and page 446

where k_{1_i}, p_{1i}, k_{2_i}, p_{2i}, σ_{1i} and ς_i for $i = 1,2$ are the elements of, K_1, K_2, $\vec{p}_1$, $\vec{p}_2$, σ_1 and ς respectively. The time derivative of $V_1(\vec{p})$ along (3.47) and (3.48) is

$$\begin{aligned}
\dot{V}_1 &= \dot{\vec{p}}_1^{\,T} K_1 \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) + \dot{\vec{p}}_2^{\,T} K_2 \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) + \vec{p}_2^{\,T} K_1 \dot{\vec{p}}_2 \\
&= -\sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) K_2 \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) - \sigma_2(K_2\vec{p}_2) K_2 \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \\
&\quad + \frac{\vec{\omega}^T}{m} K_2 \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) - \sigma_2(K_2\vec{p}_2) K_1 \vec{p}_2 + \frac{\vec{\omega}^T}{m} K_1 \vec{p}_2. \\
&\leq -\lambda_{min}(K_2) \left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\|^2 + \lambda_{max}(K_2) \left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\| \left\| \sigma_2(K_2\vec{p}_2) \right\| \\
&\quad - \lambda_{min}(K_1) \vec{p}_2^{\,T} \sigma_2(K_2\vec{p}_2) + \frac{\vec{\omega}^T}{m} (K_1\vec{p}_2 + K_2\sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2)).
\end{aligned} \tag{3.51}$$

Using Young's inequality [1], the second term of (3.51), satisfies

$$\begin{aligned}
\lambda_{max}(K_2) \left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\| \left\| \sigma_2(K_2\vec{p}_2) \right\| \leq & \frac{\lambda_{max}(K_2)}{2} \left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\|^2 \\
& + \frac{\lambda_{max}(K_2)}{2} \left\| \sigma_2(K_2\vec{p}_2) \right\|^2
\end{aligned} \tag{3.52}$$

Furthermore, there exists $\nu > 0$ such that $\left\| \sigma_2(K_2\vec{p}_2) \right\| \leq \nu \lambda_{max}(K_2) \left\| \vec{p}_2 \right\|$, for all $\vec{p}_2$, then the third term of (3.52) satisfies

$$\left\| \sigma_2(K_2\vec{p}_2) \right\|^2 \leq \nu \lambda_{max}(K_2) \vec{p}_2^{\,T} \sigma_2(K_2\vec{p}_2) \tag{3.53}$$

Substituting (3.53) into (3.52), yields

$$\left\| \sigma_1(K_1\vec{p}_1 + k_2\vec{p}_2) \right\| \left\| \sigma_2(K_2\vec{p}_2) \right\| \leq \frac{\left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\|^2}{2} + \frac{\nu\lambda_{max}}{2} \vec{p}_2^{\,T} \sigma_2(K_2\vec{p}_2) \tag{3.54}$$

From (3.54), the derivative $\dot{V}_1$ in (3.51) satisfies

$$\dot{V}_1 \leq -W_1(\vec{p}, \vec{\omega}) \tag{3.55}$$

where

$$W_1(\vec{p}, \vec{\omega}) = c_1 \left\| \sigma_1(K_1\vec{p}_1 + K_2\vec{p}_2) \right\|^2 + c_2 \vec{p}_2^{\,T} \sigma_2(K_2\vec{p}_2) - \frac{\|\vec{\omega}\|}{m} (\lambda_{max}(K_1) \left\| \vec{p}_2 \right\| + c_3) \tag{3.56}$$

with $c_1 = \frac{2\lambda_{min}(K_2) - \lambda_{max}(K_2)}{2}$, $c_2 = \frac{2\lambda_{min}(K_1) - \nu\lambda_{max}^2(K_2)}{2}$, $c_3 = \lambda_{max}(K_2)\bar{\sigma}_1$ with $\bar{\sigma}_1 \geq \|\sigma_1(\cdot)\|$. In order to ensure c_1, c_2, $c_3 > 0$, the matrices K_1, K_2 are chosen as follows:

$$2\lambda_{min}(K_2) > \lambda_{max}(K_2); \qquad 2\lambda_{min}(K_1) > \nu\lambda_{max}(K_2)^2 \tag{3.57}$$

(3.55) is used to get (3.49). Let the ball $\mathscr{B}_{\epsilon_1} = \{\vec{p} \in \mathbb{R}^3 \mid \left\| \vec{p} \right\| \leq \epsilon_1\}$, and show that $\vec{p}(t)$ approaches $\mathscr{B}_{\epsilon_1}$ after the time T_{ϵ_1}. Let $\alpha_1 = \min_{\|\vec{p}\|=\epsilon_1} V_1(\vec{p})$, then $\alpha_1 > 0$. Take $\beta_1 \in (0, \alpha_1)$ and define the set $\Omega_{\beta_1} = \{\vec{p} \in \mathscr{B} \mid V_1(\vec{p}) \leq \beta_1\}$, then Ω_{β_1} is in the interior of $\mathscr{B}_{\epsilon_1}$. As $V_1(\vec{p})$ is continuous and $V_1(0) = 0$, there exists γ_1 such that $0 < \gamma_1 < \epsilon_1$ and $\beta_2 = \max_{\|\vec{p}\|=\gamma_1} V_1(\vec{p}) < \beta_1$.

Let the ball $\beta_{\gamma_1} = \{\vec{p} \in \Omega_{\beta_1} \mid \|\vec{p}\| \leq \gamma_1\}$, and take $\alpha_2 = \min_{\|\vec{p}\|=\gamma_1} V_1(\vec{p})$, then we have $0 < \alpha_2 < \beta_2 < \beta_1 < \alpha_1$. Consequently, let the ball $\beta_{\gamma_2} = \{\vec{p} \in \mathscr{B}_{\gamma_1} \mid \|\vec{p}\| \leq \gamma_2\}$ where γ_2 satisfies $0 < \gamma_2 < \gamma_1$ and $0 < \min_{\|\vec{p}\|=\gamma_2} V_1(\vec{p}) \leq \max_{\|\vec{p}\|=\gamma_2} V_1(\vec{p}) < \alpha_2$, then from the above construction we have

$$\mathscr{B}_{\gamma_2} \subset \mathscr{B}_{\gamma_1} \subset \Omega_{\beta_1} \subset \mathscr{B}_{\epsilon_1} \tag{3.58}$$

and the Lyapunov function $V_1(\vec{p})$ satisfies

$$\begin{aligned} \max_{\|\vec{p}\|=\gamma_2} V_1(\vec{p}) &< \min_{\|\vec{p}\|=\gamma_1} V_1(\vec{p}) \\ &< \max_{\|\vec{p}\|=\gamma_1} V_1(\vec{p}) < \beta_1 < \alpha_1. \end{aligned} \tag{3.59}$$

In view of (3.58) it is sufficient to show that $\vec{p}(t)$ approaches $\mathscr{B}_{\gamma_2}$ after the finite time T_{ϵ_1}. Then, the proof is completed. ■

As the quadrotor is composed of mechanical and electrical parts, constraints are needed in order to respect the limits of its actuators and mechanical movements. Using (3.44) and (3.48) it yields

$$\begin{aligned} u_1^2 &= U_x^2 + U_y^2 + U_z^2 \\ u_1^2 &\leq (\sigma_{11} + \sigma_{21})^2 m^2 + (\sigma_{12} + \sigma_{22})^2 m^2 + (\sigma_{13} + \sigma_{23} + g)^2 m^2. \end{aligned} \tag{3.60}$$

In order to ensure $u_1 \leq \bar{u}_1$, where $\bar{u}_1$ is a thrust bounded constant defined by the properties of the actuators, and from (3.60), then the following condition is necessary

$$g - \bar{\sigma}_{13} - \bar{\sigma}_{23} > 0, \tag{3.61}$$

$$(\bar{\sigma}_{11} + \bar{\sigma}_{21})^2 + (\bar{\sigma}_{12} + \bar{\sigma}_{22})^2 + (\bar{\sigma}_{13} + \bar{\sigma}_{23} + g)^2 \leq \frac{\bar{u}_1^2}{m^2}. \tag{3.62}$$

Equation (3.43) can be rewritten as

$$\theta_d = \tan\left(\frac{-\bar{\sigma}_{11} - \bar{\sigma}_{21}}{-\bar{\sigma}_{13} - \bar{\sigma}_{23} + g}\right)^{-1} \leq \bar{\theta}_d. \tag{3.63}$$

Rewriting (3.61) as

$$\begin{aligned} -\bar{\sigma}_{13} - \bar{\sigma}_{23} &< -\frac{g}{2} \\ |\bar{\sigma}_{13} + \bar{\sigma}_{23}| &< \frac{g}{2}, \end{aligned} \tag{3.64}$$

and equation (3.62) by

$$|\bar{\sigma}_{11} + \bar{\sigma}_{21}|^2 + |\bar{\sigma}_{12} + \bar{\sigma}_{22}|^2 + |\bar{\sigma}_{13} + \bar{\sigma}_{23} + g|^2 \leq \frac{|\bar{u}_1^2|}{|m^2|}, \tag{3.65}$$

such that equation (3.63) is rewritten as

$$\tan\left(\frac{|\bar{\sigma}_{11} + \bar{\sigma}_{21}|}{|\bar{\sigma}_{13} + \bar{\sigma}_{23} + g|}\right)^{-1} \leq \bar{\theta}_d. \tag{3.66}$$

Developing the third term of (3.65)

$$|\bar{\sigma}_{11}+\bar{\sigma}_{21}|^2+|\bar{\sigma}_{12}+\bar{\sigma}_{22}|^2+|\bar{\sigma}_{13}+\bar{\sigma}_{23}|^2+2|\bar{\sigma}_{13}+\bar{\sigma}_{23}||g|+|g|^2\leq\frac{\bar{u}_1^2}{m^2}. \tag{3.67}$$

Assumption 1. *As the virtual control U_z has to compensate the weight of the vehicle (mg) to track a desired altitude ($z\longrightarrow z_d$.), then, it is possible to say that $U_z > U_x$ and $U_z > U_y$.*

Based on the Assumption 1 and considering that the vehicle is moving in the $x-$axis, thus, using (3.64) into (3.67) we obtain

$$\begin{aligned}
|\bar{\sigma}_{11}+\bar{\sigma}_{21}|^2+|\frac{g}{2}|^2+2|\frac{g}{2}||g|+|g|^2&\leq\frac{|\bar{u}_1^2|}{|m^2|}\\
|\bar{\sigma}_{11}+\bar{\sigma}_{21}|^2+\frac{g^2}{4}+2g^2&\leq\frac{\bar{u}_1^2}{m^2}\\
|\bar{\sigma}_{11}+\bar{\sigma}_{21}|^2+\frac{9}{4}g^2&\leq\frac{\bar{u}_1^2}{m^2}\\
|\bar{\sigma}_{11}+\bar{\sigma}_{21}|&\leq\sqrt{\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2}.
\end{aligned} \tag{3.68}$$

Rewriting equation (3.63) with (3.68)

$$\frac{\sqrt{\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2}}{|\bar{\sigma}_{13}+\bar{\sigma}_{23}|+g}\leq\tan(\bar{\theta}_d) \tag{3.69}$$

From the last equation

$$|\bar{\sigma}_{13}+\bar{\sigma}_{23}|\geq\left(\frac{\sqrt{\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2}}{\tan(\bar{\theta}_d)}\right)-g. \tag{3.70}$$

Now, for ϕ_d, it is possible to rewrite equation (3.43) using the inequalities (3.68) and (3.70)

$$\begin{aligned}
\frac{|\bar{\sigma}_{12}+\bar{\sigma}_{22}|}{\sqrt{\left(\sqrt{\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2}\right)^2+\left(\sqrt{\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2}-g\right)^2}}&\leq\tan\bar{\phi}_d\\
\frac{|\bar{\sigma}_{12}+\bar{\sigma}_{22}|}{\sqrt{\frac{\tan^2\bar{\theta}_d\left(\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2\right)+\left(\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2-g^2\right)}{\tan^2\bar{\theta}_d}}}&\leq\tan\bar{\phi}_d
\end{aligned} \tag{3.71}$$

Then,

$$|\bar{\sigma}_{12}+\bar{\sigma}_{22}|\leq\tan\bar{\phi}_d\left(\sqrt{\frac{\tan^2\bar{\theta}_d\left(\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2\right)+\left(\frac{\bar{u}_1^2}{m^2}-\frac{9}{4}g^2-g^2\right)}{\tan^2\bar{\theta}_d}}\right) \tag{3.72}$$

Following the inequalities given by (3.68), (3.70) and (3.72) it is possible to achieve the constraints $u_1 \leq \bar{u}_1$, $\theta \leq \bar{\theta}_d$ and $\phi \leq \bar{\phi}_d$.

To guarantee the orientation angle θ, ϕ tracks its desired value θ_d, ϕ_d in the sense that $\lim_{t\to\infty} \left\| \vec{\eta(t)} - \vec{\eta_d}(t) \right\| = 0$, where $\vec{\eta} = [\theta \ \phi]^T$ and $\vec{\eta_d} = [\theta_d \ \phi_d]^T$, such that

$$\lim_{t \longrightarrow \infty} \|\vec{\omega}(t)\| = 0. \tag{3.73}$$

Then, it is possible to rewrite attitude subsystem describing the pitch movement in equation (2.11) as in the following form

$$\begin{aligned} \dot{\vec{\eta}}_1 &= \vec{\eta}_2 \\ J\dot{\vec{\eta}}_2 &= \vec{\tau} \end{aligned} \tag{3.74}$$

where $\vec{\tau} = [\tilde{\tau}_\theta \ \tilde{\tau}_\phi]^T$, and $J \in \mathbb{R}^{2x2}$ is a diagonal constant inertial matrix. Let us propose a control input of the form

$$\bar{\tau} = -\mathcal{K}_1 \eta_1 - \mathcal{K}_2 \eta_2 \tag{3.75}$$

where $\vec{\eta}_1 = (\vec{\eta} - \vec{\eta}_d)$ and $\vec{\eta}_2 = (\dot{\vec{\eta}} - \dot{\vec{\eta}}_d)$, $\mathcal{K}_1$ and $\mathcal{K}_2$ are positive diagonal matrices that must be well chosen to ensure (3.73) and $\dot{\vec{\eta}}_d = [\dot{\theta}_d \ \dot{\phi}_d]^T$ rate is defined as

$$\begin{aligned} \dot{\theta}_d &= \frac{\dot{U}_x U_z - U_x \dot{U}_z}{U_z^2 + U_x^2} \\ \dot{\phi}_d &= \frac{\dot{U}_y U_x^2 + \dot{U}_y U_z^2 - U_y \dot{U}_x U_x - U_y \dot{U}_z U_z}{\sqrt{U_x^2 + U_z^2}\left(U_y^2 + U_x^2 + U_z^2\right)} \end{aligned} \tag{3.76}$$

with

$$\begin{aligned} \dot{\vec{U}} = &-\frac{1}{m}((1 - \sigma_1 \tanh(K_1 \vec{p}_1 + K_2 \vec{p}_2)^2) K_1 \vec{p}_2 + K_2((1 - \sigma_1 \tanh(K_1 \vec{p}_1 + K_2 \vec{p}_2)^2) \\ &+ (1 - \sigma_2 \tanh(K_2 \vec{p}_2)^2))\dot{\vec{p}}_2). \end{aligned}$$

Take $\mathcal{K}_2^2 > 4\mathcal{K}_1$. Then, the roots of the characteristic equation of the subsystem (3.74), (3.75) denoted by α, β are real with

$$\beta = \left(-\frac{\mathcal{K}_2}{2} - \sqrt{\frac{\mathcal{K}_2^2}{4} - \mathcal{K}_1} \right) < \alpha = \left(-\frac{\mathcal{K}_2}{2} + \sqrt{\frac{\mathcal{K}_2^2}{4} - \mathcal{K}_1} \right) < 0. \tag{3.77}$$

Let $\gamma > 0$ be an arbitrary constant. Now, if $\gamma\mathcal{K}_2$ replaces $\mathcal{K}_2$ and $\gamma^2\mathcal{K}_1$ replaces $\mathcal{K}_1$, then β in (3.77) is placed by $\gamma\beta$ and α by $\gamma\alpha$. Hence, it can easily be shown that for the given $\vec{\tau} > 0$ and any fixed $r_\eta > 0$ one can select a pair $\mathcal{K}_1$, $\mathcal{K}_2$ such that $\left\|\eta_0\right\| \leq r_\eta$ implies $|\eta_i(t)| \leq \vartheta_i r_\eta \exp(\alpha t)$ where $0 \leq \vartheta_i$, $i = 1,2$ depend on η_{i0}, and in particular

$$|\mathcal{K}_1 \eta_1(t) + \mathcal{K}_2 \eta_2(t)| \leq (\mathcal{K}_1 \vartheta_1 + \mathcal{K}_2 \vartheta_2) r_\eta < \vec{\tau}. \tag{3.78}$$

3.1.5 Numerical and experimental results

This section presents numerical and experimental results to validate the control strategies developed in (3.11), (3.22), (3.36), (3.48). The robot used to validate this overview of control laws is quadcopter system emulating a PVTOL vehicle for simplifying further analysis.

The scenario consists in moving the longitudinal axis of the vehicle as following: the vehicle starts at $x(0) = -2m$ and $z(0) = 1m$, then, three set-points defined by $x_{r_1} = 2$, $x_{r_2} = 4$ and $x_{r_3} = 5$ in meters are imposed as desired values. For better illustrating the graphs obtained in simulation and experimental results, we denote as LB, NLB, NS, the Linear/NonLinear Backstepping and the Nested Saturation controllers, respectively, called also *classical* approaches. The HSC means the Hyperbolic Saturation Controller based on the fully actuated approach, named also *virtual* method. Parameters for numerical and practical validation are shown in Table 3.1.

Table 3.1 – Parameters values used in the control laws in (3.11), (3.22), (3.36) and (3.48) used in simulation and experimental validation.

Controller	Control parameters	Sim. Values	Exp. Values
LB	k_1	1	1.31
	k_2	1.2	1
	k_3	0.8	0.8
	k_4	0.1	0.1
NLB	$\bar{k}_1$	0.04	0.04
	$\bar{k}_2$	0.035	0.0025
	$\bar{k}_3$	0.4	0.4
	$\bar{k}_4$	0.1	0.1
NS	σ_a	0.15	0.15
	σ_b	0.22	0.22
	σ_c	0.1	0.1
	σ_d	0.3	0.3
HSC	σ_{11}	0.5	0.5
	σ_{12}	0.1	0.2
	σ_{21}	0.5	0.5
	σ_{22}	0.35	0.4
	k_{11}	5	1
	k_{22}	0.9	1.3
	k_{11}	3	0.3
	k_{22}	0.55	0.6

Numerical results

The performance of the PVTOL vehicle when using controllers (3.11), (3.22), (3.36) and (3.48) tracking the set-points previously defined is shown in Figures 3.1 and 3.3. Note from Figure 3.1 that, the convergence using controllers without bounded functions are faster than the others that use saturation functions.

This performance is 'normal' because the saturation functions are chosen so small to guarantee a convergence from any initial condition. To increase the speed of convergence of these controllers, the bound of the saturation functions must be increased.

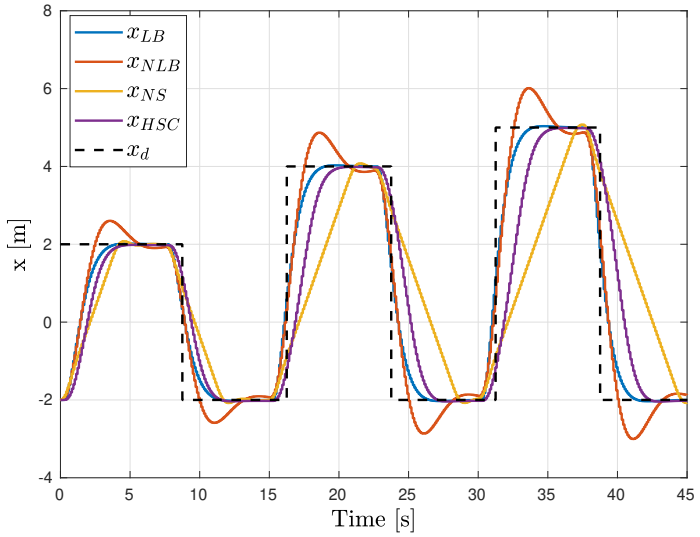


Figure 3.1 – Longitudinal position performance when comparing the four control algorithms.

Besides, as depicted in Figure 3.2, the θ angle responses for controllers NS and HSC are bounded, even if initial conditions are far from the desired position. In addition, for the controllers LB and NLB the angular position is bigger than for the NS and HSC controllers and its magnitude is related to the error position. Notice that only for the HSC algorithm the angular position tracks the imposed reference.

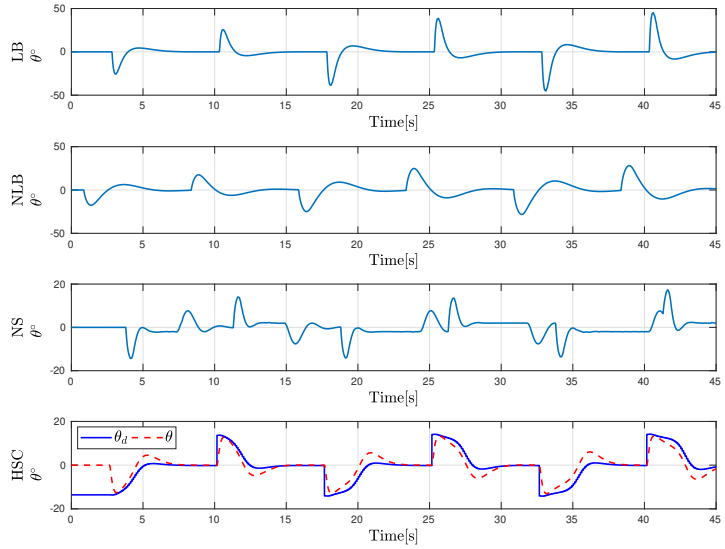


Figure 3.2 – θ angle response when applying four controllers into the PVTOL dynamics. θ_d denotes the desired angle computed in (3.43).

Figure 3.3 is the most important graph to analyze the performance of the four controllers. In this figure, z performance is depicted showing the advantage of using a virtual controller from the classical one. Observe here that the controllers using classical procedure their altitude performance is degraded when tracking the references. In practical validation, this is observed as a drop in the altitude performance (z decreases). This drop is related to how far the desired value is.

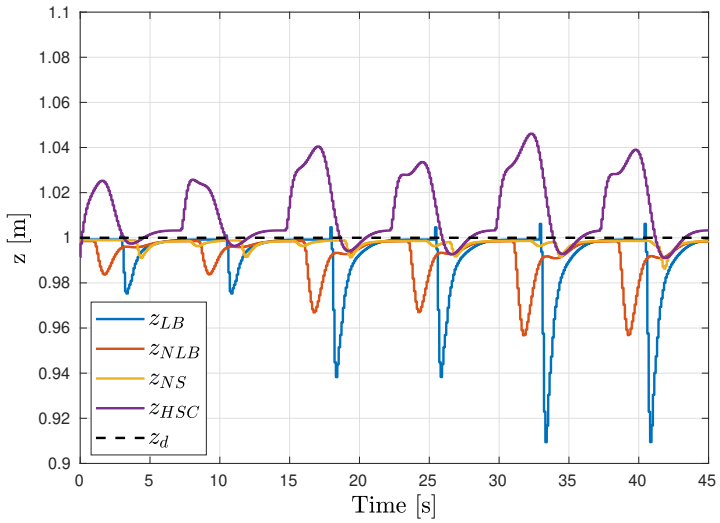


Figure 3.3 – Vertical position behavior when comparing the four control algorithms.

Similarly, the performance of the vehicle comparing the four control algorithms in the $x-z$ plane is displayed in Figure 3.4. In one hand, it can be verified that when using control algorithms based on *classical* method, the behaviour of the vehicle in the $z-$axis is compromized. On the other hand, controllers based on the *virtual* approach compensate this error by means of the correction term F_2 in the total thrust.

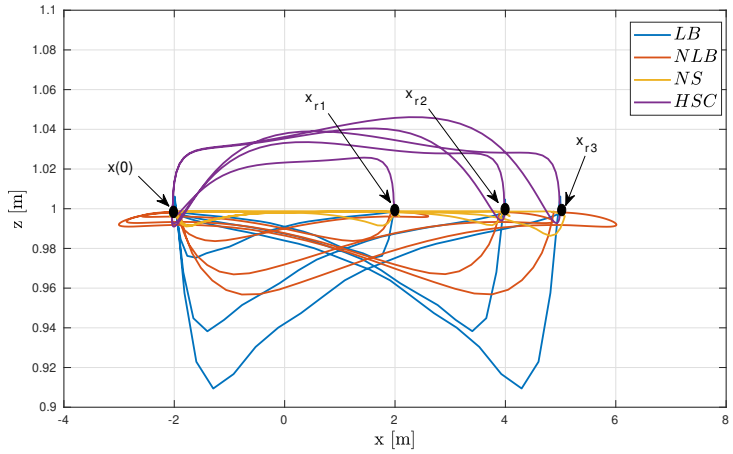


Figure 3.4 – Vehicle's performance evolution in the $x-z$ plane when applying the four control algorithms into the PVTOL dynamics. From figure x_{r_i} for $i:1,2,3$ describes the desired way-point.

In Figures 3.5 and 3.6 the control input responses for the four controllers are depicted. Notice from Figure 3.5 that u_1 response is different for the HSC controller with respect to the others that decreases when the desired point is modified. Notice that for the HSC controller u_1 increases to compensate the lateral displacement. Nevertheless, in Figure 3.6 the u_2 performance is quite different. Notice here that u_2 for the HSC controller is smaller with respect to the others one.

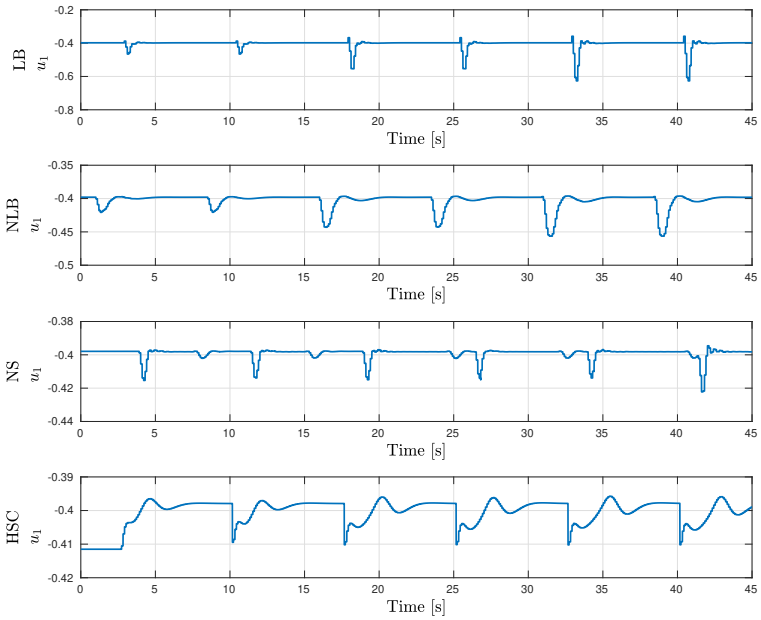


Figure 3.5 – u_1 behavior when applying the controllers (3.11), (3.26), (3.36) and (3.48).

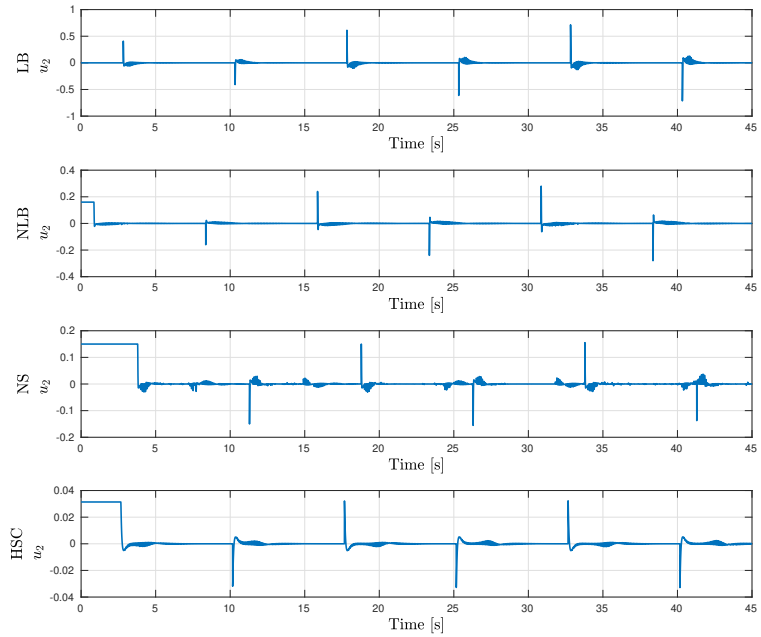


Figure 3.6 – Torque control u_2 response when applying the four control algorithms.

Experimental results

The control algorithms (3.11), (3.26), (3.36) and (3.48) were validated in real time in a quadcopter vehicle evolving as a PVTOL vehicle. For this, front motors (f_{L_2}, f_{R_2}) and rear motors (f_{L_1}, f_{R_1}) in the aerial vehicle are considered producing only a force in the front and rear side of the vehicle, as depicted in Figure 3.7. A video with the experimental results can be seen at: https://youtu.be/Fi4uUbQRrgw

Notice from this figure that $f_1 = f_{L_1} + f_{R_1}$ and $f_2 = f_{L_2} + f_{R_2}$. Vertical displacement on the z axis is produced when increasing or decreasing, by the same magnitude the speed in the motors, while pitch moment is produced with the difference between f_1 and f_2. This quadcopter vehicle evolving as a PVTOL is an AR Drone 2. Its firmware was modified to work under the software *Fl-AIR - Framework libre AIR* which is open source and runs a Linux-based operating system, capable of implementing a wide range of control schemes, see [160]. An OptiTrack motion capture system was used to estimate the vehicle's position, while its internal Inertial Measurement Unit (IMU) measures its orientation and angular rates.

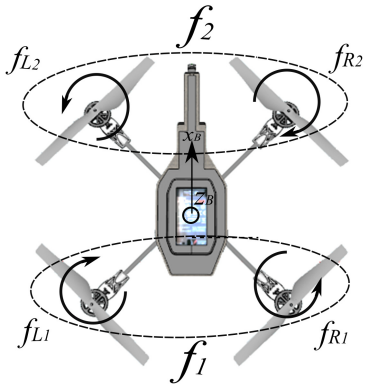


Figure 3.7 – Quadcopter configuration evolving its longitudinal plane.

Conditions for flight tests are the same as used in the simulation section, i.e. the initial conditions are $x(0) = -2m$ and $z(0) = 1m$ while $x_{r_1} = 2$, $x_{r_2} = 4$ and $x_{r_3} = 5$ in meters are the references. In Figures 3.8- 3.11 the state responses when applying the controllers are illustrated. Notice from Figure 3.8, as in the simulation case, the convergence of controllers based on saturation functions are quite slower due to the bound of these functions that helps to maintain a relative small error with respect to the others controllers. Nevertheless, the LB controller has a faster convergence but in this case, implying more energy and therefore a bigger attitude response as we will see in the next figures.

The angular performance of the quadcotper system when using the four control algorithms is depicted in Figure 3.9. Remark that for the virtual control algorithm a desired angle is imposed and tracked satisfactorily. Similarly in this figure observe that θ-response is bigger for those controllers that are mainly designed without saturation functions. From this figure it is possible to make a relation between the fast convergence of controllers and their attitude response. This means that, faster the convergence is achieved, then, bigger the attitude response.

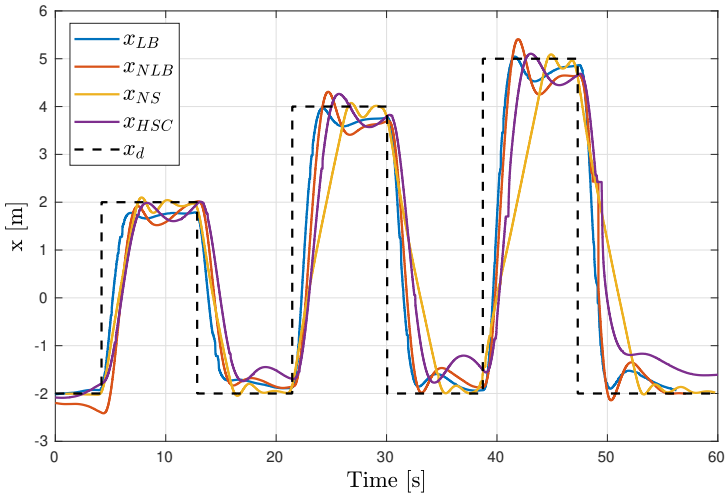


Figure 3.8 – Longitudinal behavior when applying experimentally the four control algorithms into the aerial vehicle.

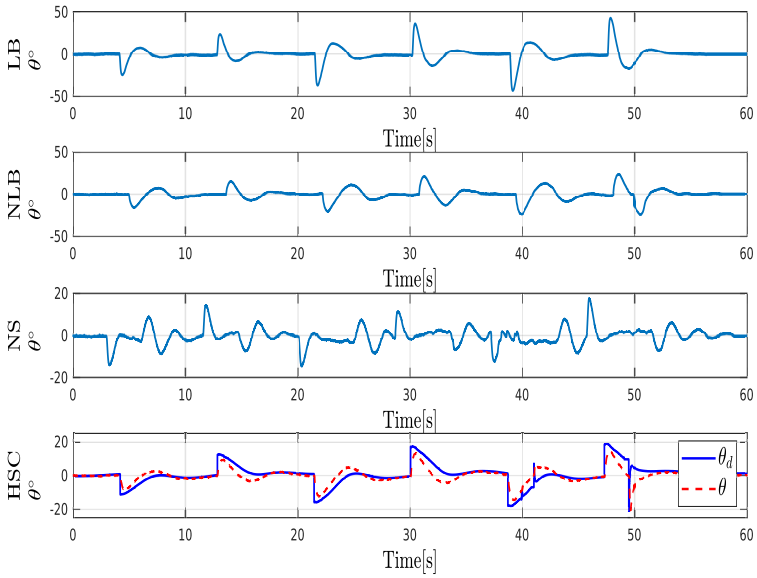


Figure 3.9 – θ angle response when the four control algorithms where applied in real time. θ_d describes the desired angle.

Observe in Figure 3.10, that the magnitude of the error in the $z-$axis is directly related to the $x-$axis error. This means that if the desired x-position is far, then, the error in the vertical axis increases. And as a consequence, the magnitude of the θ angle is affected as can be verified in Figure 3.9. Therefore, as observed from this same figure, it is possible to use small values for the saturation functions or furthermore reduce the control gains of the LB and NLB controllers, however it will compromise the performance of the system reducing the convergence of the states to the desired positions.

Moreover, a big angle response should increase the vertical error in the vehicle when using control algorithms based on the *classical* approach as can be seen in Figure 3.11. Besides, when using controllers based on the *virtual* approach or controllers using bounded functions, this error is minimized as shown in Fig. 3.11. Notice the performance of the LB and NLB controllers. Due the longitudinal error increases, the altitude response is affected in order to fulfill the control objective. It is well known that this altitude error can be fixed re-tuning the control gains for every new desired position, however that was out of the scope of this thesis.

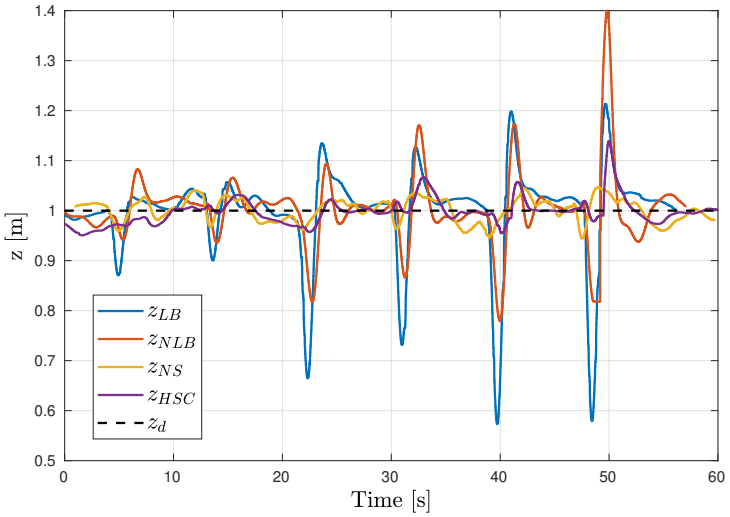


Figure 3.10 – Vertical position comparing the four control algorithms.

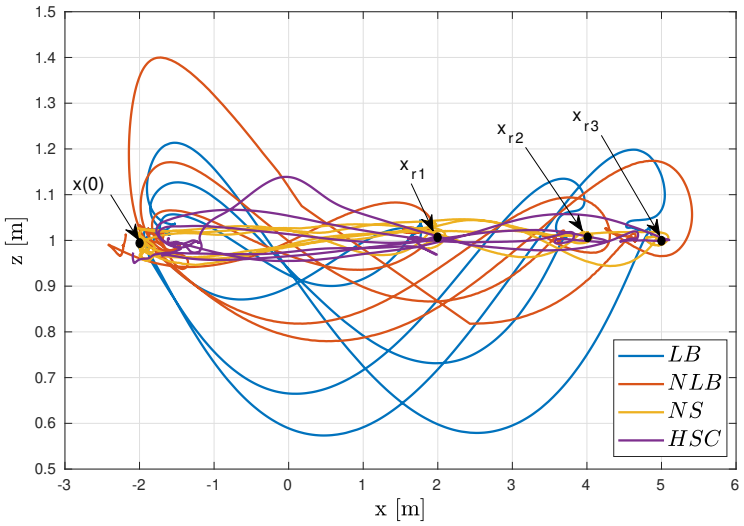


Figure 3.11 – Vehicle's performance evolution in the $x-z$ plane when the four control algorithms are applied in real time.

In Figures 3.12 and 3.13 the control input responses are illustrated. Notice that they have obtained similar performance as in the case of simulations results. In the next section, we introduce a discussion in terms of the control inputs using the indices performance and qualitative variables to expose our conclusions for the four control algorithms.

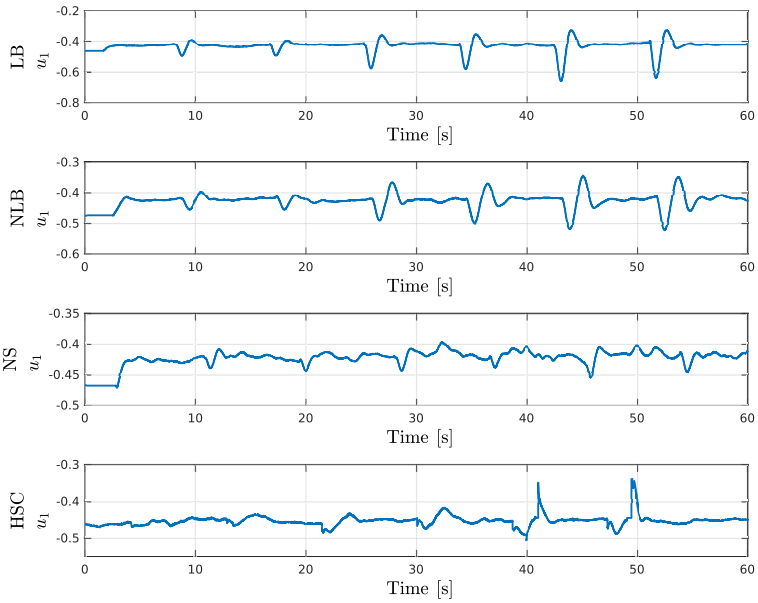


Figure 3.12 – Thrust control u_1 response for the four control algorithms applied to the aerial vehicle.

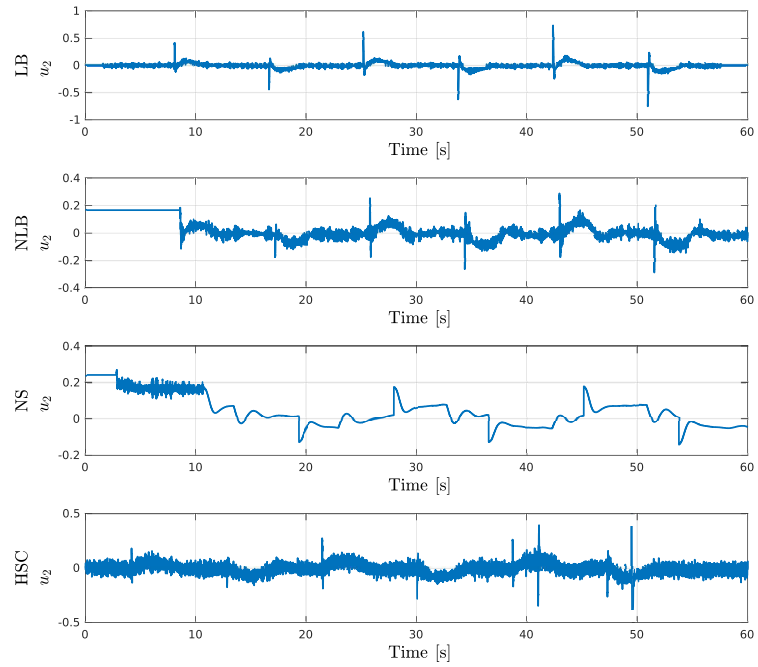


Figure 3.13 – Torque control response, u_2, obtained during the flight tests.

3.1.6 Discussion

We have carried out an analysis of the performance of the four control algorithms. For this purpose, the performance indices and a qualitative comparison of the controller are obtained. The performance indices are the Integral Square Error (ISE), the Integral Absolute Error (IAE), the Integral time Squared Error (ITSE) and the Integral time Absolute Error (ITAE). A summary of this analysis is presented in Tables 3.2 - 3.4.

From the steady state regime experimental results, in Table 3.2 it can be observed that the Nested Saturation and Hyperbolic Saturation controllers achieve the best performance in terms of the vertical error. The HSC presents less error than NS. However, the HSC algorithm includes a bigger computational effort with respect to the other algorithms due the derivatives of the desired angle θ_d. It can be analyzed from the design of the HSC controller that imposes desired angles, therefore, this increases the time of convergence to the desired translational values. Nevertheless, it could be an advantage because when imposing desired angles, we can ensure converge even if the desired position are far away. Observe also from tables that the Nested Saturation controller is quite small with respect to the other approaches due mainly to the values of the bound of the saturation functions. The linear backstepping controller presents a small error in the longitudinal axis and faster convergence time as can be observed in tables. Nonetheless, according to Table 3.2 this control algorithm has a bigger vertical error and by consequence control effort u_2 required is also bigger.

Table 3.2 – Performance indices of the $z-$axis

Controllers/Indices	IAE	ISE	ITAE	ITSE
LB	2.97	0.48	99.48	17.98
NLB	2.40	0.30	82.38	12.33
NS	1.16	0.03	35.63	1.02
HSC	1.09	0.04	30.04	1.35

Table 3.3 – Performance indices of the $x-$axis

Controllers/Indices	IAE	ISE	ITAE	ITSE
LB	60.21	221.56	1453	7132
NLB	64.53	241.37	1937	7780
NS	77.96	290.13	2324	10188
HSC	83.51	308.68	266.76	14752

Regarding the design, tuning and the implementation effort, the linear backstepping and nested saturation algorithms are easier to tune because their structure can be seen as Proportional Derivative controllers. In contrast, the NLB and the HSC are tedious to implement since both present several complex derivatives and they have more parameters to tune. In this context, Table 3.4 presents a behavior comparison between the four algorithms including numerical and practical validation and other qualitative indicators.

Table 3.4 – Qualitative comparison of the control algorithms. The best result is denoted with 1 and the worst with 4.

	LB	NLB	NS	HSC
Converge to the reference	1	4	3	2
Computational effort	1	3	2	3
Control effort	4	3	1	1
Design & tuning effort	1	4	2	3
Real implementation	1	4	2	3

Concerning the methods to design these control laws, one of the advantages of the *virtual* method is the possibility to design any control law (adaptive control, sliding mode control, etc) in F_1 and F_2 that could significantly improve the performance of the system. Here, the challenge will be to design control laws capable to provide desired angles respecting the mechanical constraints of the vehicle for avoiding singularities in the control input u_2. The disadvantage, as previously mentioned, is the complexity of the implementation of the algorithms due of the first and second derivative of F_1 and F_2. In contrast, the *classical* method gives the advantage of an easy fashion to get the analysis and design of the control algorithms based on the assumption that the vertical axis is controlled. However, the control effort of the inputs u_1 and u_2 could be larger if the desired positions are far from the initial conditions.

3.2 Quaternion-based backstepping control

In this section, a backstepping quaternion control is designed based on the proposed dynamic model (2.52) developed in Section 2.3. According to this section and in particular, subsection 2.3.2, it is possible to decouple the whole dynamic model of the vehicle in terms of a desired attitude $\boldsymbol{q_d}$, which is described by a desired thrust vector force $\vec{U} \in \mathscr{I}$. Let us consider the rotational dynamic model of the quadcopter represented by (2.51) as the associated quaternion for the tracking attitude error defined by (2.53). Therefore, the thrust vector force $\vec{U}$ must be established in terms of the translational system (2.54).

Backstepping position controller

Following the procedure of the bacsktepping control technique, as in the case of using Euler-angles, let us define the tracking error define the error e_{x_1} as $e_{x_1} = x_1 - x_1^d$, where x_1^d is the reference. Let us propose the following positive function $V_1 = \frac{\alpha_{x_1}}{2} e_{x_1}^2$, with $\alpha_{x_1} > 0$ denoting a constant, then taking the derivative with respect to time and proposing $x_2^v = x_2^d - e_{x_1}$ as a virtual control input, it follows that

$$\dot{V}_1 = \alpha_{x_1} e_{x_1} (x_2 - x_2^v - e_{x_1}) = -\alpha_{x_1} e_{x_1}^2 + \alpha_{x_1} e_{x_1} (x_2 - x_2^v)$$

where $\dot{x}_1^d = x_2^d$. Define the error $e_{x_2} = x_2 - x_2^v$ thus, the previous yields $\dot{V}_1 = -\alpha_{x_1} e_{x_1}^2 + \alpha_{x_1} e_{x_1} e_{x_2}$. Propose the second positive function $V_2 = \frac{\alpha_{x_2}}{2} e_{x_2}^2$ with $\alpha_{x_2} > 0$ constant.

Differentiating with respect to time

$$\dot{V}_2 = \alpha_{x_2} e_{x_2} (U_x - \dot{x}_2^v)$$

Proposing $U_x = -e_{x_2} + \dot{x}_2^v - \frac{\alpha_1}{\alpha_2} e_{x_1} = -e_{x_2} + \dot{x}_2^d - x_2 + x_2^d - \frac{\alpha_{x_1}}{\alpha_{x_2}} e_{x_1}$, the, defining the Lyapunov function $V_T = V_1 + V_2$ and differentiating with respect to time, it follows that

$$\begin{aligned} \dot{V}_T &= \dot{V}_1 + \dot{V}_2 = -\alpha_{x_1} e_{x_1}^2 + \alpha_{x_1} e_{x_1} e_{x_2} - \alpha_{x_2} e_{x_2}^2 - \alpha_1 e_{x_1} e_{x_2} \\ \dot{V}_T &= -\alpha_{x_1} e_{x_1}^2 - \alpha_{x_2} e_{x_2}^2 < 0 \end{aligned} \tag{3.79}$$

The procedure to get the virtual control inputs U_y and U_z is exactly the same as for U_x. Thus without lost of generality, it is possible to define the control vector of the form

$$\vec{U} = \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix} = \begin{bmatrix} -e_{x_2} + \dot{x}_2^d - x_2 + x_2^d - \frac{\alpha_{x_1}}{\alpha_{x_2}} e_{x_1} \\ -e_{y_2} + \dot{y}_2^d - y_2 + y_2^d - \frac{\alpha_{y_1}}{\alpha_{y_2}} e_{y_1} \\ -e_{z_2} + \dot{z}_2^d - z_2 + z_2^d - \frac{\alpha_{z_1}}{\alpha_{z_2}} e_{z_1} + g \end{bmatrix} \tag{3.80}$$

where α_{i_j} are positive constants. Therefore, the thrust can be defined as

$$u_1 = \frac{\sqrt{U_x^2 + U_y^2 + U_z^2}}{m}. \tag{3.81}$$

Then, from (2.56) and (2.57) let us define the unit vector associated with $\vec{U}$ as

$$\vec{\mu} = \frac{\vec{U}}{\|\vec{U}\|} = \frac{1}{\sqrt{U_x^2 + U_y^2 + U_z^2}} \begin{bmatrix} U_x \\ U_y \\ U_z \end{bmatrix}. \tag{3.82}$$

The desired quaternion $\boldsymbol{q_d}$ can be computed using (2.59) and (2.60) as follows

$$\boldsymbol{q_d} = \begin{bmatrix} q_{d0} \\ \vec{q}_d \end{bmatrix} = \begin{bmatrix} q_{d0} \\ q_{d1} \\ q_{d2} \\ q_{d3} \end{bmatrix}, \qquad q_{d0} = \sqrt{\frac{1 + \vec{v}\,\vec{\mu}}{2}}, \qquad \vec{q}_d = \sqrt{\frac{1 + \vec{v}^T\,\vec{\mu}}{2}} S(\vec{\mu})\,\vec{v} \tag{3.83}$$

where $\vec{v} = [0\ 0\ 1]^T$ and $S(\vec{\mu})$ is defined as

$$S(\vec{\mu}) = \begin{bmatrix} 0 & -\mu_3 & \mu_2 \\ \mu_3 & 0 & -\mu_1 \\ -\mu_2 & \mu_1 & 0 \end{bmatrix}.$$

Consequently, choosing proper control gains α_{i_j} in (3.80) such that (3.81) aligns $\vec{F}_\xi$ with $\vec{U}$, then, the position error will converge to zero and therefore, the translational quadrotor system (2.54) can be stabilized in any desired position.

Once the desired quaternion is defined, the control of the attitude system and its stability analysis must be addressed. For that, let us propose the following change of coordinates

$$\vec{\mathcal{Z}}_1 = \begin{bmatrix} 1 - |q_{e0}| \\ \vec{q}_e \end{bmatrix} \tag{3.84}$$

where $\vec{q}_e = [q_{e1}\ q_{e2}\ q_{e3}]^T$ and $\boldsymbol{q_e} = \boldsymbol{q_d^*} \otimes \boldsymbol{q}$. Then computing its derivative with respect to time

$$\begin{bmatrix} -\operatorname{sgn} q_{e0} \left(q_{d0}\dot{q}_0 + \dot{q}_{d0} q_0 + \dot{\vec{q}}_d^T \vec{q} + \vec{q}_d^T \dot{\vec{q}} \right) \\ -\dot{\vec{q}}_d q_0 - \vec{q}_d \dot{q}_0 + \dot{q}_{d0} \mathbb{I}_3 \vec{q} + q_{d0} \mathbb{I}_3 \dot{\vec{q}} - S(\dot{\vec{q}}_d)\vec{q} - S(\vec{q}_d)\dot{\vec{q}} \end{bmatrix} \tag{3.85}$$

Considering the case of regulation problem, it follows that $\dot{q}_{d0}$, $\dot{\vec{q}}_d \to 0$, then

$$\dot{\vec{\mathcal{Z}}}_1 \begin{bmatrix} -\operatorname{sgn} q_{e0} \left(q_{d0}\dot{q}_0 + \vec{q}_d^T \dot{\vec{q}} \right) \\ q_{d0}\dot{\vec{q}} - \vec{q}_d \dot{q}_0 - S(\vec{q}_d)\dot{\vec{q}} \end{bmatrix} \tag{3.86}$$

From (2.51), it follows that (3.86) can be rewritten as

$$\dot{\vec{\mathcal{Z}}}_1 = \frac{1}{2} \begin{bmatrix} -sgn(q_{e0}) \left(q_{d0}\vec{q}^T + \vec{q}_d^T (q_0 \mathbb{I}_3 + S(\vec{q})) \right) \\ q_{d0}(q_0 \mathbb{I}_3 + S(\vec{q})) - \vec{q}_d \vec{q}^T - S(\vec{q}_d)(q_0 \mathbb{I}_3 + S(\vec{q})) \end{bmatrix} \tag{3.87}$$

Remark that the first element of (3.87) can be expressed as $\dot{\vec{\mathcal{Z}}}_1(1) = \operatorname{sgn}(q_{e0})\vec{q}_e^T$ and the second element can be rewritten using the property $S(a)S(b) = ba^T - a^T b\mathbb{I}$ as

$$\dot{\vec{\mathcal{Z}}}_1(2) = \frac{1}{2} \left(q_{d0}\vec{q}_d \mathbb{I}_3 + q_{d0} S(\vec{q}) + \vec{q}^T \vec{q}_d - S(\vec{q}_d) q_0 \mathbb{I}_3 - S(\vec{q}_d) S(\vec{q}) \right) \vec{\Omega} \tag{3.88}$$

Defining

$$S(\vec{q}_e) = q_{d0} S(\vec{q}) + \vec{q}^T q_{d0} - S(\vec{q}_d) q_0 \mathbb{I}_3 - \vec{q}\vec{q}_d^T \tag{3.89}$$

then, $\dot{\vec{\mathcal{Z}}}_1$ is expressed as

$$\dot{\vec{\mathcal{Z}}}_1 = \frac{1}{2} M(\boldsymbol{q_e}) \vec{\Omega} \tag{3.90}$$

where

$$M(\boldsymbol{q_e}) = \begin{bmatrix} \operatorname{sgn}(q_e 0)\vec{q}_e^T \\ q_{e0}\mathbb{I}_3 + S(\vec{q}_e) \end{bmatrix}.$$

Let us propose a positive function of the form $V_1 = \frac{1}{2}\vec{\mathcal{Z}}_1^T \vec{\mathcal{Z}}_1$ and differentiating with respect to time is given by

$$\dot{V}_1 = \frac{1}{2} \vec{\mathcal{Z}}_1^T M(\boldsymbol{q_e}) \vec{\Omega} \tag{3.91}$$

The main idea behind the backstepping technique is to propose a virtual control input such that $\dot{V}_1 \leq 0$, thus, let us introduce

$$\vec{\Omega}^v = -K_1 M^T(\boldsymbol{q_e}) \vec{\mathcal{Z}}_1 \tag{3.92}$$

then, the derivative of V_1 can be expressed as

$$\dot{V}_1 = -\frac{1}{2} \vec{\mathcal{Z}}_1^T M(\boldsymbol{q_e}) K_1 M({}^T \boldsymbol{q_e}) \vec{\mathcal{Z}}_1 \leq 0 \tag{3.93}$$

Proposing the second error as $\vec{\mathcal{Z}}_2 = \vec{\Omega} - \vec{\Omega}^v$ and differentiating with respect to time,

$$\begin{aligned} \dot{\vec{\mathcal{Z}}}_2 &= \dot{\vec{\Omega}} - \dot{\vec{\Omega}}^v \\ \dot{\vec{\mathcal{Z}}}_2 &= I^{-1}(-\vec{\Omega} IS(\vec{\Omega}) + \vec{\tau} - I\dot{\vec{\Omega}}^v) \end{aligned} \tag{3.94}$$

The system in terms of the errors $\vec{\mathcal{Z}}_1$ and $\vec{\mathcal{Z}}_2$ and its derivatives can be expressed as

$$\begin{aligned} \dot{\vec{\mathcal{Z}}}_1 &= \frac{1}{2} M(\boldsymbol{q_e})(\vec{\Omega}^v + \vec{\mathcal{Z}}_2) \\ \dot{\vec{\mathcal{Z}}}_2 &= I^{-1}(-\vec{\Omega} IS(\vec{\Omega}) + \vec{\tau} - I\dot{\vec{\Omega}}^v) \end{aligned} \tag{3.95}$$

Proposing a Lyapunov function of the form $V_T = \frac{1}{2}\vec{\mathcal{Z}}_1^T \vec{\mathcal{Z}}_1 + \frac{1}{2}\vec{\mathcal{Z}}_2^T \vec{\mathcal{Z}}_2$ and its derivative with respect to time given by

$$\begin{aligned} \dot{V}_T &= \vec{\mathcal{Z}}_1^T \dot{\vec{\mathcal{Z}}}_1 + \vec{\mathcal{Z}}_2^T \dot{\vec{\mathcal{Z}}}_2 \\ &= \frac{1}{2}\vec{\mathcal{Z}}_1^T (M(\boldsymbol{q_e})\vec{\Omega}^v + M(\boldsymbol{q_e})\vec{\mathcal{Z}}_2) + \vec{\mathcal{Z}}_2^T (I^{-1}(-\vec{\Omega} IS(\vec{\Omega}) + \vec{\tau} - I\dot{\vec{\Omega}}^v)) \end{aligned} \tag{3.96}$$

Choosing $\vec{\tau}$ as

$$\vec{\tau} = \vec{\Omega} IS(\vec{\Omega}) + I\dot{\vec{\Omega}}^v - K_2\vec{\mathcal{Z}}_2 - \frac{1}{2}M(\boldsymbol{q_e})^T \vec{\mathcal{Z}}_1 \tag{3.97}$$

then, it can be see that using (3.97) system (3.95) is stabilized and therefore, the full dynamic quadcopter model described by (2.52) is asymptotically stable

$$\dot{V}_T = -\frac{1}{2}\vec{\mathcal{Z}}_1^T M(\boldsymbol{q_e}) K_1 M^T(\boldsymbol{q_e})\vec{\mathcal{Z}}_1 - \vec{\mathcal{Z}}_2^T K_2 \vec{\mathcal{Z}}_2 \leq 0. \tag{3.98}$$

The final control input equation using the quaternion backstepping control can be expressed in terms of states as follows

$$\begin{aligned} \vec{\tau} = &\begin{bmatrix} 0 & -\Omega_x & 0 \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} - \begin{bmatrix} K_{2x} & 0 & 0 \\ 0 & K_{2y} & 0 \\ 0 & 0 & K_{2z} \end{bmatrix} \\ &\left(\begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} - \left(\begin{bmatrix} K_{1x} & 0 & 0 \\ 0 & L_{1y} & 0 \\ 0 & 0 & K_{1z} \end{bmatrix} \times \frac{1}{2} \begin{bmatrix} \text{sgn}(q_{e0})q_{e1} & q_0 & -q_3 & q_2 \\ \text{sgn}(q_{e0})q_{e2} & q_3 & q_0 & -q_1 \\ \text{sgn}(q_{e0})q_{e3} & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 1-|q_{e0}| \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \right) \right) \\ &- \frac{1}{2} \begin{bmatrix} \text{sgn}(q_{e0})q_{e1} & q_0 & -q_3 & q_2 \\ \text{sgn}(q_{e0})q_{e2} & q_3 & q_0 & -q_1 \\ \text{sgn}(q_{e0})q_{e3} & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 1-|q_{e0}| \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \end{aligned} \tag{3.99}$$

with K_{1i} and K_{2i} where $i \in [x,\ y,\ z]$ are positive constants, Ω_i represents the angular velocity and I stands for the inertia tensor.

3.2.1 Experimental test

In this section, the validation of the control algorithm (3.99) based on the quadcopter quaternion model (2.52) is addressed. For this experimental test, an AR Drone 2 was employed using the Fl-AIR framework open source code. The scenario consists in the following: the vehicle is located in a random hover position $\vec{\xi}(0) = [0,\ 0,\ 1.2m]^T$ and thus, a circular desired trajectory $\vec{\xi}_d = [a\cos(t),\ a\sin(t),\ 1.2m]^T$ must be tracked. The desired quaternion around the $z-$axis is defined by $\boldsymbol{q_z} = [\cos(\psi_d/2),\ 0,\ 0, \sin(\psi_d/2)]^T$ where $\psi_d = \arctan\left(\frac{\xi_{dy}}{\xi_{dx}}\right)$, and the desired quaternion is given by $\boldsymbol{q_d} = \boldsymbol{q_t} \otimes \boldsymbol{q_z}$. The control parameters used in this test are depicted in Table 3.5.

The performance of the vehicle when tracking a circular desired trajectory is presented in Figures 3.14 to 3.18. Notice from Figures 3.14 and 3.15 the well behavior of the proposed control algorithm. It can be observed that the x and y axes are well tracked with a small ϵ error in the $z-$axis. The vehicle's front is pointing towards the sense of the trajectory according to the desired ψ_d angle.

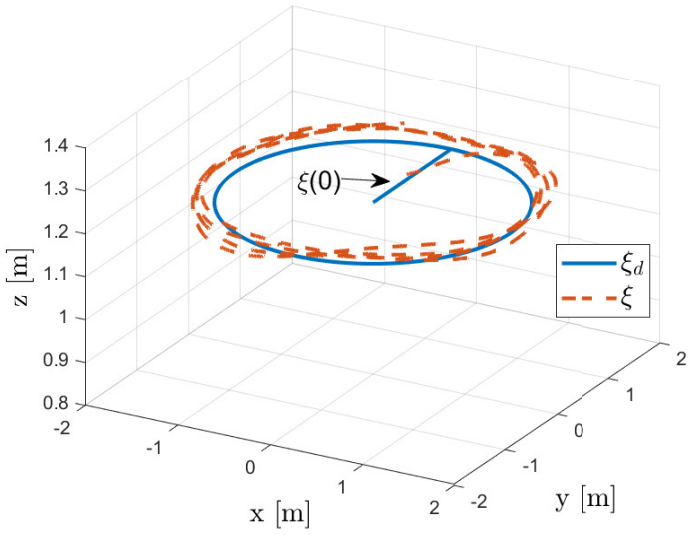


Figure 3.14 – Performance of the quadcpter during the flight test

Table 3.5 – Control gains parameters used in experimental tests

Parameter	x	y	z
α_1	0.1	0.1	0.1
α_2	0.8	0.8	0.8
K_2	0.2	0.2	0.2
L_1	30	30	10

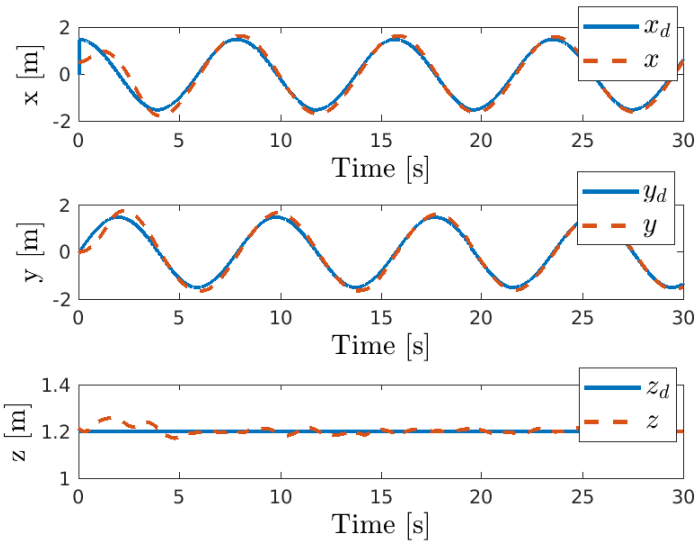


Figure 3.15 – Quadcotper position tracking

The desired attitude trajectory, computed by (2.60) and (3.83), is illustrated in Figure 3.16. Notice the well performance of q_0 and q_3 which are related with the desired ψ_d angle. Moreover, 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.

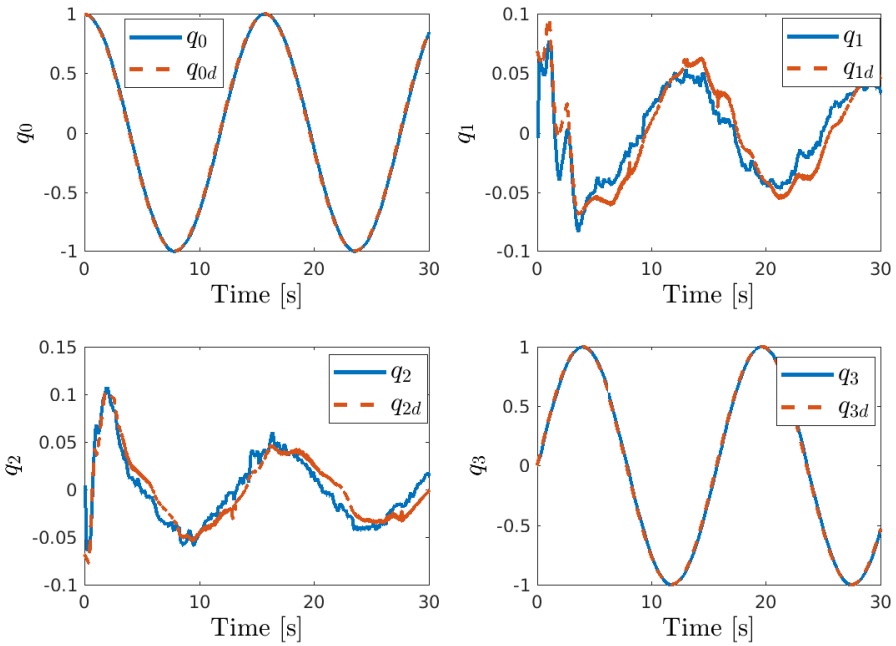


Figure 3.16 – Quadcopter attitude quaternion performance when tracking the desired pose trajectory

Figures 3.17 and 3.18 present the performance of the designed controllers (3.80), (3.81) and (3.99) when tracking a circular desired trajectory. Let us recall that this controller compute the torques and forces required to stabilize the vehicle in the desired references.

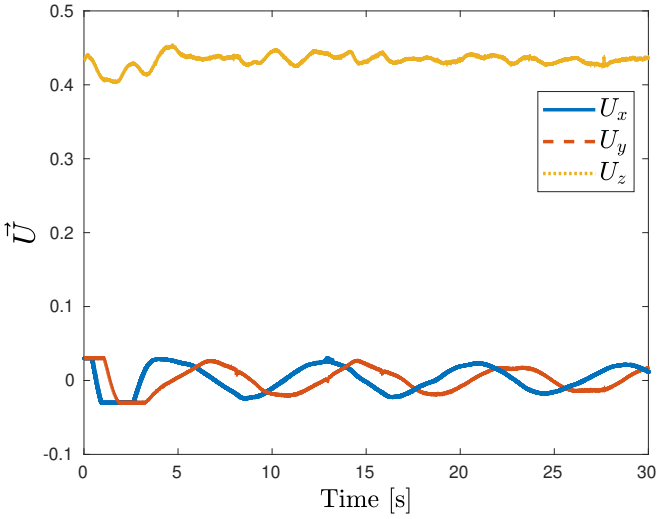


Figure 3.17 – Performance of the control input forces

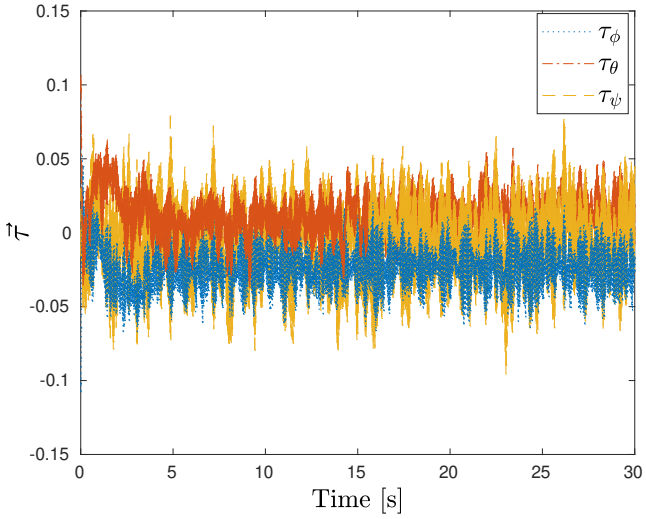


Figure 3.18 – Performance of the rotational controller

To better illustrate the vehicle's rotational behavior, the equivalent Euler angles are illustrated on Figure 3.19 and were computed using the following equation

$$\begin{aligned} \phi &= \arctan\left(\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)}\right), \\ \theta &= \arcsin\left(2(q_0q_2-q_1q_3)\right), \\ \psi &= \arctan\left(\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\right). \end{aligned} \tag{3.100}$$

Note this conversion uncovers the discontinuities that are present when completing full rotations (±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.

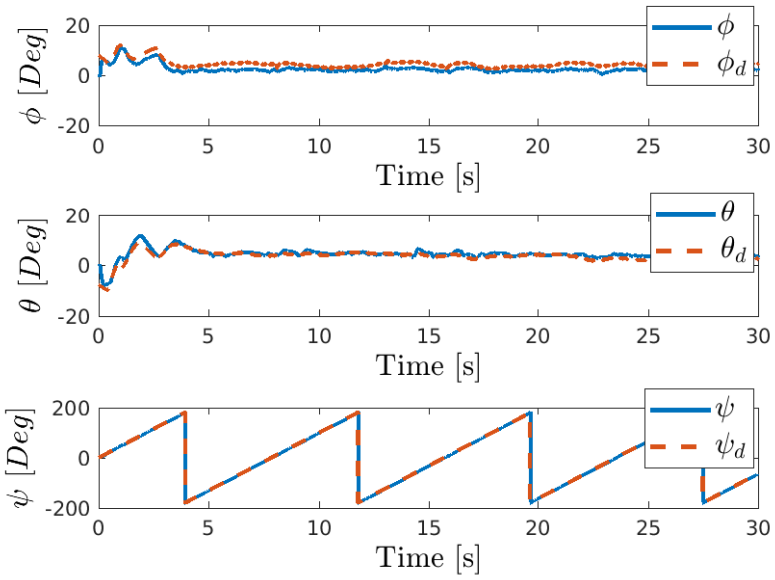


Figure 3.19 – Quadcopter attitude axis-angle representation

3.3 Finite-time convergence using Sliding Mode Control

The next method developed in this thesis concerns the convergence in finite time of an aerial drone to a mobile target using the properties of the sliding mode control. In order to that, the target must to remain in the field of view (FoV) of the quadrotor vehicle. Therefore, the goal is to compute desired trajectory with finite time convergence $\bar{t}$, such that, the control objective $\vec{r}(\bar{t}) = \vec{\xi}_T - \vec{r}_s$ can be guaranteed in finite-time convergence. Note $\vec{\xi}_T = [x_T \ y_T \ z_T]^\top$ represents the position of the target and $\vec{r}_s = \vec{\xi} - \vec{\xi}_d$ means the error between the quadrotor and, a desired reference to keep a distance with the mobile target. This distance is defined by the constant $\vec{\xi}_d = [x_d, \ y_d, \ z_d]^\top$. The goal of this section is to design a desired trajectory such that the quadcopter can follow and achieve the position $\vec{r}(\bar{t})$ in finite time $\bar{t}$. This trajectory will be composed of three components $x_c(t)$, $y_c(t)$ and $z_c(t)$, and represented by the position vector $\vec{\xi}_c(t) = \left[x_c(t), y_c(t), z_c(t)\right]^\top$ where each component $x_c(t) \to x_T$, $y_c(t) \to y_T$ in finite time $\bar{t}$, and with a constant altitude $z_c(t) = z_c(0)$ for all $t \geq 0$.

3.3.1 $x_c(t)$ and $y_c(t)$ components

For designing the first component $x_c(t)$ such that $x_c(t) \to x_T$ in finite time $\bar{t}$ let us define the position error as $\varrho_{x_c}(t) = x_c(t) - x_T$. Notice that if $\varrho_{x_c}(t) = 0$ for all $t \geq \bar{t}$ then, this will imply that $x_c(t) = x_T \forall\, t \geq \bar{t}$. Computing the derivative of ϱ_{x_c} with respect to time, it follows that $\dot{\varrho}_{x_c}(t) = \dot{x}_c(t)$, $\ddot{\varrho}_{x_c}(t) = \ddot{x}_c(t)$ where its initial conditions are given by $\varrho_{x_c}(0) = x_c(0) - \bar{x}_1$, $\dot{\varrho}_{x_c}(0) = \dot{x}_c(0) = 0$, $\ddot{\varrho}_{x_c}(0) = \ddot{x}_c(0) = 0$.

The objective is that $\varrho_{x_c}(t) = \dot{\varrho}_{x_c}(t) = \ddot{\varrho}_{x_c}(t) = 0$ in a finite time $\bar{t}$, thus, the following variables and its derivatives can be stated: $\sum_{x_c}(t) = \varrho_{x_c}(t) - \mu_x(t)$, $\dot{\sum}_{x_c}(t) = \dot{\varrho}_{x_c}(t) - \dot{\mu}_x(t)$ and $\ddot{\sum}_{x_c}(t) = \ddot{\varrho}_{x_c}(t) - \ddot{\mu}_x(t)$ with

$$\begin{aligned}
\mu_x(t) &= \left(t-\bar{t}\right)^3\left(c_0 + c_1 t + c_2 t^2\right) \\
\dot{\mu}_x(t) &= \left(t-\bar{t}\right)^3\left(c_1 + 2c_2 t\right) + 3\left(t-\bar{t}\right)^2\left(c_0 + c_1 t + c_2 t^2\right) \\
\ddot{\mu}_x(t) &= 2c_2\left(t-\bar{t}\right)^3 + 6\left(t-\bar{t}\right)^2\left(c_1 + 2c_2 t\right) + 6\left(t-\bar{t}\right)\left(c_0 + c_1 t + c_2 t^2\right)
\end{aligned}$$

and for derivatives $i : 0, 1, 2$

$$\overset{i}{\mu}_x(t) = \begin{cases} \overset{i}{\mu}_x(t) & \text{for } 0 \le t < \bar{t} \\ 0 & \text{for } t \ge \bar{t} \end{cases} \tag{3.101}$$

Then

$$\overset{i}{\sum}_{x_c}(t) = \begin{cases} \overset{i}{\varrho}_{x_c}(t) - \overset{i}{\mu}_x(t) & \text{for } 0 \le t < \bar{t} \\ \overset{i}{\varrho}_{x_c}(t) & \text{for } t \ge \bar{t} \end{cases} \tag{3.102}$$

Observe that $\sum_{x_c}(0) = 0$ implies that $\varrho_{x_c}(0) = \mu_x(0)$, $c_0 = (\bar{x}_1 - x_c(0))/\bar{t}^3$. $\dot{\sum}_{x_c}(0) = 0$ means that $\dot{\varrho}_{x_c}(0) = \dot{\mu}_x(0)$, hence $c_1 = 3c_0/\bar{t}$ and $\ddot{\sum}_{x_c}(0) = 0$ means that $\ddot{\varrho}_{x_c}(0) = \ddot{\mu}_x(0)$, then $c_2 = (3c_1\bar{t} - 3c_0)/\bar{t}^2$.

Taking the third derivative of $\sum_{x_c}$, it follows that $\dddot{\sum}_{x_c}(t) = \dddot{\rho}_{x_c}(t) - \dddot{\mu}_x(t) = \dddot{x}_c(t) - \dddot{\mu}_x(t)$. If

$$\dddot{x}_c(t) = -\alpha_x \text{sign}\left(\ddot{\sum}_{x_c} + 2\left(\left|\dot{\sum}_{x_c}\right|^3 + \left|\sum_{x_c}\right|^2\right)^{1/6} \times \text{sign}\left(\dot{\sum}_{x_c} + \left|\sum_{x_c}\right|^{2/3}\text{sign}\left(\sum_{x_c}\right)\right)\right) \tag{3.103}$$

with a sufficiently large α_x. The finite-time stability is ensured from Theorem 6.2 in [161]. The above implies that $\varrho_{x_c}(t) = \mu_x(t)$, $\dot{\varrho}_{x_c}(t) = \dot{\mu}_x(t)$ and $\ddot{\varrho}_{x_c}(t) = \ddot{\mu}_x(t)\ \forall\, t \ge t_r$. Nevertheless, if $t_r \le \bar{t}$ when $t \to \bar{t}$ and from (3.101) it follows that $\varrho_{x_c}(t) = \mu_x(t) = 0$, $\dot{\varrho}_{x_c}(t) = \dot{\mu}_x(t) = 0$, and $\ddot{\varrho}_{x_c}(t) = \ddot{\mu}_x(t) = 0$, $\forall\, t \ge \bar{t}$. This means that $x_c(t) = x_T$ for all $t \ge \bar{t}$ therefore the finite-time convergence to x_T is achieved.

Propose $x_c = x_{1c}$, $\dot{x}_{1c} = x_{2c}$, $\dot{x}_{2c} = x_{3c}$ then it follows that

$$\begin{aligned}
\dot{x}_{1c} &= x_{2c} \\
\dot{x}_{2c} &= x_{3c} \\
\dot{x}_{3c} &= -\alpha_x \text{sign}\left(\ddot{\sum}_{x_c} + 2\left(\left|\dot{\sum}_{x_c}\right|^3 + \left|\sum_{x_c}\right|^2\right)^{1/6} \times \text{sign}\left(\dot{\sum}_{x_c} + \left|\sum_{x_c}\right|^{2/3}\text{sign}\left(\sum_{x_c}\right)\right)\right)
\end{aligned} \tag{3.104}$$

with

$$\alpha_x = \begin{cases} 0, & \text{for } \ t = 0 \\ \alpha_x > 0, & \text{for } \ t > 0 \end{cases} \tag{3.105}$$

α_x is defined in this form to achieve $\dot{x}_{3c}(0) = 0$.

Following the previous procedure the $y_c(t)$ component is obtained as

$$\begin{aligned}\dot{y}_{1c} &= y_{2c}\\ \dot{y}_{2c} &= y_{3c}\\ \dot{y}_{3c} &= -\alpha_y \text{sign}\left(\ddot{\sum}_{y_c} + 2\left(\left|\dot{\sum}_{y_c}\right|^3 + \left|\sum_{y_c}\right|^2\right)^{1/6} \times \text{sign}\left(\dot{\sum}_{y_c} + \left|\sum_{y_c}\right|^{2/3}\text{sign}\left(\sum_{y_c}\right)\right)\right)\end{aligned} \tag{3.106}$$

with

$$\alpha_y = \begin{cases} 0, & \text{for} \quad t = 0 \\ \alpha_y > 0, & \text{for} \quad t > 0 \end{cases} \tag{3.107}$$

α_y is defined in this form to achieve $\dot{y}_{3c}(0) = 0$.

3.3.2 Sliding mode control design

The goal in this subsection is to design a controller, based on the sliding mode properties, capable to make the quadcopter tracks the proposed trajectory $\vec{\xi}_c(t)$ with asymptotic convergence. Thus, the first step is to define the tracking errors as follows

$$\begin{aligned}\dot{e}_{x_1} &= e_{x_2}; & \dot{e}_{x_2} &= -\sin(\theta_1)\frac{1}{m}u_1 - \delta_x\\ \dot{e}_{y_1} &= e_{y_2}; & \dot{e}_{y_2} &= \cos(\theta_1)\sin(\phi_1)\frac{1}{m}u_1 - \delta_y\\ \dot{e}_{z_1} &= e_{z_2}; & \dot{e}_{z_2} &= \cos(\theta_1)\cos(\phi_1)\frac{1}{m}u_1 - \delta_z\end{aligned} \tag{3.108}$$

where $e_{x_1} = x_1 - x_{1c}$; $e_{y_1} = y_1 - y_{1c}$; $e_{z_1} = z_1 - z_c(0)$; $\dot{e}_{x_1} = e_{x_2} = x_2 - x_{2c}$; $\dot{e}_{y_1} = e_{y_2} = y_2 - y_{2c}$ and $\dot{e}_{z_1} = e_{z_2} = z_2$. Also, $\delta_x = x_{3c}$, $\delta_y = y_{3c}$ and $\delta_z = g$.

Proposing u_1 as

$$u_1 = \frac{m}{\cos(\theta_{1c})\cos(\phi_{1c})}\left[\delta_z + r_3\right] \tag{3.109}$$

where ϕ_{1c} and θ_{1c} are the desired angles and r_3 will be defined later to assure convergence.

Introducing u_1 into $\dot{e}_{z_2}$ it follows that

$$\dot{e}_{z_2} = \cos(\theta_1)\cos(\phi_1)\frac{1}{\cancel{m}}\left\{\frac{\cancel{m}}{\cos(\theta_{1c})\cos(\phi_{1c})}\left[\delta_z + r_3\right]\right\} - \delta_z$$

If $\phi_1 \to \phi_{1c}$, $\theta_1 \to \theta_{1c}$, $\psi_1 \to \psi_{1c}$, this will imply that

$$\dot{e}_{z_1} = e_{z_2}; \quad \dot{e}_{z_2} = r_3$$

Defining $r_3 = -K_{p_z}e_{z_1} - K_{d_z}e_{z_2}$ with $K_{p_z} > 0$ and $K_{d_z} > 0$, then $e_{z_1}, e_{z_2} \to 0$.

Then, the new goal is to find ϕ_{1c} and θ_{1c}. Introducing u_1 into $\dot{e}_{y_2}$ in (3.108), and considering that τ_ϕ, τ_θ, τ_ψ are designed such that $\phi_1 \to \phi_{1c}$, $\theta_1 \to \theta_{1c}$, $\psi_1 \to \psi_{1c}$, then

$$\dot{e}_{y_2} = \tan(\phi_{1c})\left[\delta_z + r_3\right] - \delta_y \tag{3.110}$$

and ϕ_{1c} can be computed as

$$\phi_{1c} = \tan^{-1}\left(\frac{\delta_y + r_2}{\delta_z + r_3}\right) \tag{3.111}$$

Therefore, the next subsystem is obtained

$$\dot{e}_{y_1} = e_{y_2}; \quad \dot{e}_{y_2} = r_2$$

Proposing $r_2 = -K_{p_y} e_{y_1} - K_{d_y} e_{y_2}$ with $K_{p_y} > 0$ and $K_{d_y} > 0$ this implies that $e_{y_1}, e_{y_2} \to 0$.

Introducing u_1 into $\dot{e}_{x_2}$ in (3.108) and considering that τ_ϕ, τ_θ, τ_ψ are designed such that $\phi_1 \to \phi_{1c}, \theta_1 \to \theta_{1c}, \psi_1 \to \psi_{1c}$, then

$$\dot{e}_{x_2} = -\tan(\theta_{1c})\left\{\frac{1}{\cos(\phi_{1c})}[\delta_z + r_3]\right\} - \delta_x$$

and θ_{1c} can be proposed as

$$\theta_{1c} = \tan^{-1}\left\{(-1)\cos(\phi_{1c})\left(\frac{\delta_x + r_1}{\delta_z + r_3}\right)\right\} \tag{3.112}$$

Hence, the following subsystem is obtained

$$\dot{e}_{x_1} = e_{x_2}; \quad \dot{e}_{x_2} = r_1$$

and proposing $r_1 = -K_{p_x} e_{x_1} - K_{d_x} e_{x_2}$ with $K_{p_x} > 0$ and $K_{d_x} > 0$, we will able to accomplish the asymptotic convergence of e_{x_1} and e_{x_2} around $(0,0)$.

To simplify notation, define $T_x = \delta_x + r_1$, $T_y = \delta_y + r_2$ and $T_z = \delta_z + r_3$. Therefore the desired angular velocities $\dot{\phi}_{1c} = \phi_{2c}$ and $\dot{\theta}_{1c} = \theta_{2c}$ are

$$\begin{cases} \phi_{2c} = \dfrac{\cos^2(\phi_{1c})}{T_z}\left[\dot{T}_y - \dot{T}_z \tan(\phi_{1c})\right] \\ \theta_{2c} = \dfrac{\cos^2(\theta_{1c})}{T_z}\left[T_x \sin(\phi_{1c})\phi_{2c} - \dot{T}_z \tan(\theta_{1c}) - \dot{T}_x \cos(\phi_{1c})\right] \end{cases} \tag{3.113}$$

The initial desired angular positions and velocities are given by $\phi_{1c}(0) = 0$, $\theta_{1c}(0) = 0$ and $\phi_{2c}(0) = 0$, $\theta_{2c}(0) = 0$.

From (2.18), (3.111) and (3.112) it is possible to represent the attitude of a quadcopter in errors terms as follows

$$\begin{array}{ll} \dot{e}_{\phi_1} = e_{\phi_2} & \dot{e}_{\phi_2} = \bar{f}_1 + b_1\tau_\phi + \zeta_1 \\ \dot{e}_{\theta_1} = e_{\theta_2} & \dot{e}_{\theta_2} = \bar{f}_2 + b_2\tau_\theta + \zeta_2 \\ \dot{e}_{\psi_1} = e_{\psi_2} & \dot{e}_{\psi_2} = \bar{f}_3 + b_3\tau_\psi + \zeta_3 \end{array} \tag{3.114}$$

where $e_{\phi_1} = \phi_1 - \phi_{1c}$, $e_{\theta_1} = \theta_1 - \theta_{1c}$, $e_{\psi_1} = \psi_1 - \psi_{1c}$, $\dot{e}_{\phi_1} = e_{\phi_2} = \phi_2 - \dot{\phi}_{1c}$, $\dot{e}_{\theta_1} = e_{\theta_2} = \theta_2 - \dot{\theta}_{1c}$, $\dot{e}_{\psi_1} = e_{\psi_2} - \dot{\psi}_{1c}$. Moreover $\bar{f}_1 = (e_{\theta_2} + \dot{\theta}_{1c})(e_{\psi_2}\gamma_1 - \beta_1) - \ddot{\phi}_{1c}$, $\bar{f}_2 = (e_{\phi_2} + \dot{\phi}_{1c})(e_{\psi_2}\gamma_2 - \beta_2) - \ddot{\theta}_{1c}$, $\bar{f}_3 = (e_{\theta_2} + \dot{\theta}_{1c})(e_{\phi_2} + \dot{\phi}_{1c})\gamma_3 - \cancelto{0}{\ddot{\psi}_{1c}}$. $\zeta_i \leq L_i$ for $i \in [1,\ 2,\ 3]$ are bounded perturbations. The desired angle yaw ψ_{1c} is considered constant, then $\dot{\psi}_{1c} = \ddot{\psi}_{1c} = 0$.

Once the rotational errors are given in terms of the desired angles ϕ_c, θ_c, then the goal is to design $\tau_\phi, \tau_\theta, \tau_\psi$ such that $e_{\phi_1} = \phi_1 - \phi_{1c} \to 0$, $e_{\theta_1} = \theta_1 - \theta_{1c} \to 0$ and $e_{\psi_1} = \psi_1 - \psi_{1c} \to 0$ in finite time t^*, where $t^* < \bar{t}$. In this way e_{ϕ_1}, e_{θ_1} and e_{ψ_1} converge to zero faster, to guarantee that the quadcopter position goes to the desired trajectory in finite time $\bar{t}$. In addition, it is also necessary that $e_{\phi_1} \approx 0$, $e_{\theta_1} \approx 0$ and $e_{\psi_1} \approx 0 \quad \forall\, 0 \le t < t^*$, the previous will imply that $\phi_1 \approx \phi_{1c}$, $\theta_1 \approx \theta_{1c}$, $\psi_1 \approx \psi_{1c}$ for $0 \le t < t^*$.

To guarantee the above stated, a new tracking trajectory and its derivatives are defined as

$$ {}^i\chi(t) = \begin{cases} {}^i\mu(t) & \text{for} \quad 0 < t < t^* \\ 0 & \text{for} \quad t = 0 \text{ and } t \ge t^* \end{cases} \tag{3.115} $$

with $i : 0, 1, 2$ and $\mu(t)$, $\dot{\mu}(t)$ and $\ddot{\mu}(t)$ into (3.115) are given by

$$ \mu(t) = c\left(\exp\left(-\sin^3\left(\frac{\pi t}{t^*}\right)\right) - 1\right) \tag{3.116} $$

$$ \dot{\mu}(t) = -\frac{3\pi c}{2t^*}\sin\left(\frac{\pi t}{t^*}\right)\sin\left(\frac{2\pi t}{t^*}\right)\exp\left(-\sin^3\left(\frac{\pi t}{t^*}\right)\right) \tag{3.117} $$

$$ \begin{aligned} \ddot{\mu}(t) = {} & \frac{9\pi^2 c}{4t^{*2}}\sin^2\left(\frac{\pi t}{t^*}\right)\sin^2\left(\frac{2\pi t}{t^*}\right)\exp\left(-\sin^3\left(\frac{\pi t}{t^*}\right)\right) \\ & -\frac{3\pi^2 c}{2t^{*^2}}\left\{\cos\left(\frac{\pi t}{t^*}\right)\sin\left(\frac{2\pi t}{t^*}\right) + 2\cos\left(\frac{2\pi t}{t^*}\right)\sin\left(\frac{\pi t}{t^*}\right)\right\} \\ & \exp\left(-\sin^3\left(\frac{\pi t}{t^*}\right)\right) \end{aligned} \tag{3.118} $$

Notice from previous equations that $\mu(0) = \mu(t^*) = 0$, $\dot{\mu}(0) = \dot{\mu}(t^*) = 0$ and $\ddot{\mu}(0) = \ddot{\mu}(t^*) = 0$.

Propose the following sliding surfaces

$$ S_j = e_j - \chi(t); \qquad j : \phi_1, \theta_1, \psi_1 \tag{3.119} $$

Thus, using (3.119) for $j : \phi_1$ and (3.114), it follows that $\dot{S}_{\phi_1} = e_{\phi_2} - \dot{\chi}(t)$ and $\ddot{S}_{\phi_1} = \bar{f}_1 + b_1\tau_\phi + \zeta_1 - \ddot{\chi}(t)$. Notice that if $S_{\phi_1}(0) = 0$ it yields, $\dot{S}_{\phi_1}(0) = 0$. Proposing τ_ϕ as

$$ \tau_\phi = b_1^{-1}\left(\alpha_\phi \text{sign}\left(\dot{S}_{\phi_1} + \left|S_{\phi_1}\right|^{1/2}\text{sign}\left(S_{\phi_1}\right)\right) + \ddot{\chi}(t) - \bar{f}_1\right), \tag{3.120} $$

Using a same procedure that those propose for τ_θ, it follows that derivating (3.119) for $j : \theta_1$, it yields $\dot{S}_{\theta_1} = e_{\theta_2} - \dot{\chi}(t)$ and $\ddot{S}_{\theta_1} = \bar{f}_2 + b_2\tau_\theta + \zeta_2 - \ddot{\chi}(t)$. Thus, τ_θ can be proposed as

$$ \tau_\theta = b_2^{-1}\left(-\alpha_\theta \text{sign}\left(\dot{S}_{\theta_1} + \left|S_{\theta_1}\right|^{1/2}\text{sign}\left(S_{\theta_1}\right)\right) + \ddot{\chi}(t) - \bar{f}_2\right), \tag{3.121} $$

Finally, for designing τ_ψ we derivate (3.119) for $j : \psi_1$ and then $\dot{S}_{\psi_1} = e_{\psi_2} - \dot{\chi}(t)$ and $\ddot{S}_{\psi_1} = \bar{f}_3 + b_3\tau_\psi + \zeta_3 - \ddot{\chi}(t)$. Propose

$$ \tau_\psi = b_3^{-1}\left(-\alpha_\psi \text{sign}\left(\dot{S}_{\psi_1} + \left|S_{\psi_1}\right|^{1/2}\text{sign}\left(S_{\psi_1}\right)\right) + \ddot{\chi}(t) - \bar{f}_3\right). \tag{3.122} $$

Notice in (3.120), (3.121) and (3.122), $\alpha_k \geq L_i$ with $k \in [\phi,\ \theta,\ \psi]$, $i \in [1,\ 2,\ 3]$ in order to compensate the bounded perturbations. Therefore it follows that $S_{\phi_1} = \dot{S}_{\phi_1} = 0$, $S_{\theta_1} = \dot{S}_{\theta_1} = 0$ and $S_{\psi_1} = \dot{S}_{\psi_1} = 0$ in a finite time $t^\star$. The finite-time stability of the system is guaranteed from control theory in [161] for the "nested" 2-sliding control.

3.3.3 Numerical and experimental results

This section presents the numerical and experimental results in order to validate the control strategies developed in (3.120), (3.121) and (3.122). As in the case of the last results, the vehicle used to implement the control algorithms was an Ar.Drone Parrot 2.0 under the software Fl-AIR which is open source and runs a Linux-based operating system, capable of implementing a wide range of control schemes, see [160]. For obtaining the vehicle's position, an OptiTrack motion capture system was used while its internal Inertial Measurement Unit (IMU) measures its orientation and angular rates.

For validating the proposed control law, let us take the following scenario: consider a flying target and quadcotper vehicle evolving in $\mathbb{R}^2$ at random position. The goal is that the quadcopter converge near to the flying target, $\vec{\xi}_d$, in finite time. For simulation tests, a desired finite time convergence of $\bar{t} = 5$s is proposed as an example for verifying the effectiveness of the control algorithm. Moreover, for the experimental validation, $\bar{t} = 5$s and $\bar{t} = 3$s are proposed as desired finite times convergence. Therefore, the vehicle must to track the desired nonlinear dynamics trajectories $x_c(t)$ and $y_c(t)$ described by (3.103) and (3.106). Parameters for numerical and practical validation are shown in Table 3.6.

The vehicle should arrives to the desired position with a fixed altitude $z(0) = 1$m in a finite time $\bar{t}$ that denotes the arriving time. Therefore, the following assumptions are established:

1. The quadcotper is located in a random hover position.

2. The flying target is static but its position and velocity are known.

3. The vehicle movements are delimited to the horizontal plane x, y.

Table 3.6 – Control parameters used in experimental and numerical tests for $\bar{t} = 5$s

Parameter	θ	ϕ	ψ	x, y	z
kp	50	100	100	0.8	1.7
kd	2.5	4	4	0.2	0.7
α	3	3	3	$c = 0.0002$	$m = 0.432\ kg$

Numerical results

Take the initial conditions of the flying target as $x_T(0) = 1m$, $y_T(0) = 1m$, $z_T(0) = 1m$ and for the quadcotper at $x(0) = -3m$, $y(0) = -3m$, $z(0) = 1m$. Figures 3.20 – 3.23 present the performance of the quadrotor when converging in finite time to a desired position $\vec{\xi}_d$. Notice from Figure 3.20 and 3.21 the well performance of the control algorithm in the $z-$axis. From these figures, its is possible to observe that the desired trajectories $\vec{\xi}_c$ arrives at the desired finite time $\bar{t} = 5s$, however as the dynamic of the trajectories were tuned faster than the quadcotper position dynamics, an error in the x and y coordinates appears until it converge to zero ($t \geq \bar{t}$).

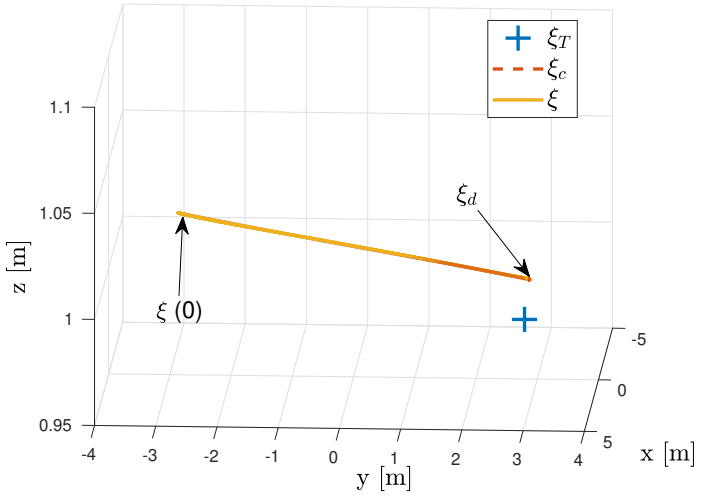


Figure 3.20 – Finite time convergence of the quadcopter to the flying target at $\bar{t} = 5s$

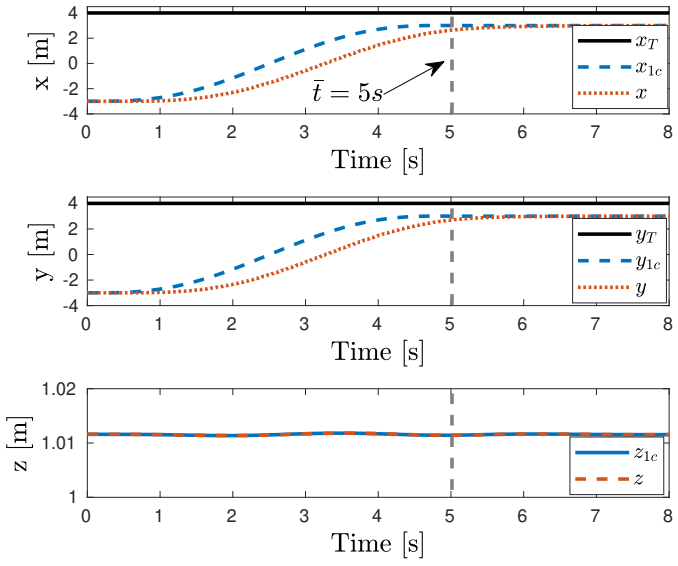


Figure 3.21 – Performance of the x, y, z axes of the quadrotor when converging in finite time.

The performance of desired attitude and the orientation of the vehicle are illustrated in Figure 3.22. Notice that from this figure the desired attitude performance has slight oscillations. This is mainly due to the fact that the finite time convergence trajectory is composed by the sgn function. Therefore, the desired angles (3.111) and (3.112) are affected as can be verified in the figure. The performance of the control inputs are depicted in Figure 3.23. These signals present the well-known effect "chattering" because the composition of their equations. It is worth mentioning that, this kind of strategies has a large demand of energy, but at the same time gives robustness.

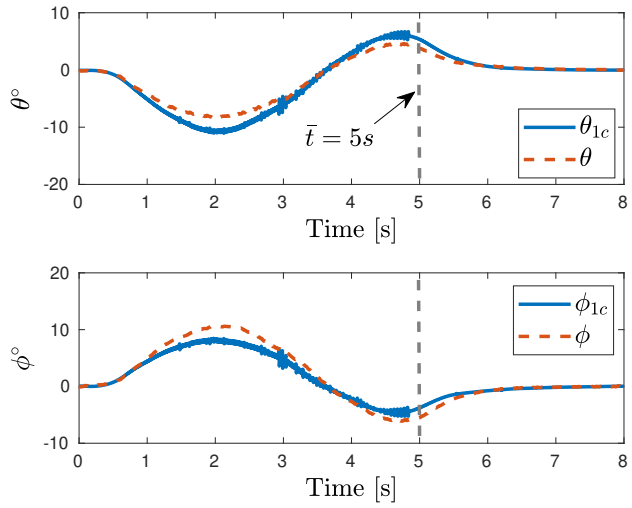


Figure 3.22 – Tracking performance of the attitude system of the quadrotor.

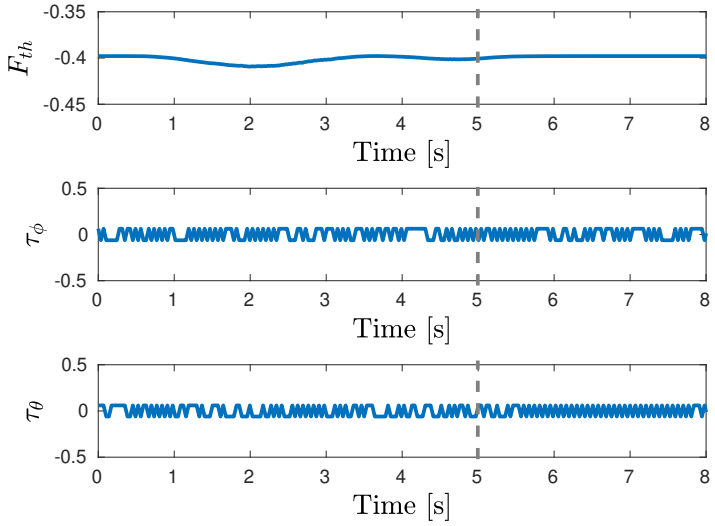


Figure 3.23 – Control inputs of the quadrotor.

Experimental results

The initial conditions for the practical validation were set as follows: the flying target is located hovering in $\vec{\xi}_T = [3,\ 2, 1.2]^T m$ while the quadcopter starts from $\vec{\xi} = [-2,\ -2\ 1.2]^T m$ and the desired safe distance $\vec{\xi}_d = [1,\ 0]^T m$. For this experiment, some control parameters were modified such that $kp_\theta = kp_\phi = 140$, $kd_\theta = kd_\phi = 4$, $kp_x = kp_y = 0.9$ and $kd_x = kd_y = 0.35$.

In Figures 3.24 – 3.27, the performance of the vehicle and its control inputs when performing the finite time convergence for $\bar{t} = 5s$ and $\bar{t} = 3s$ are depicted. Observe from Figures 3.24 and 3.25 when the quadcotper is performing for $\bar{t} = 5s$ the desired trajectories $\vec{\xi}_c$ and the vehicle's position converge with a good performance, with not overshoot and small altitude error. However, for $\bar{t} = 3s$ a slight overshoot appeared in the desired trajectories due the α parameter of (3.103) and (3.106). Nevertheless, this effect does not affect the performance of the control algorithm, as can be confirmed in Figure 3.24 (right column images). In addition, notice from Figure 3.25 the altitude and position errors were increased.

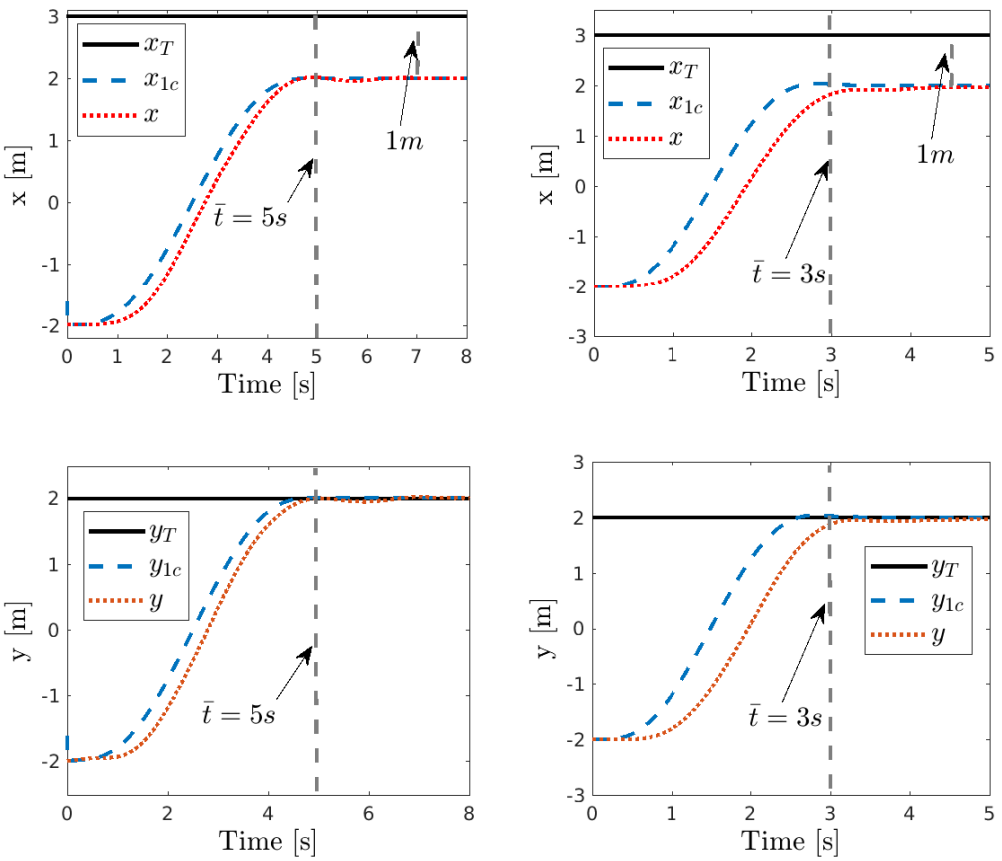


Figure 3.24 – Performance of the x and y axes of the quadcopter. Left column of the picture represents the experiment when $\bar{t} = 5$s and the right column for $\bar{t} = 3$s.

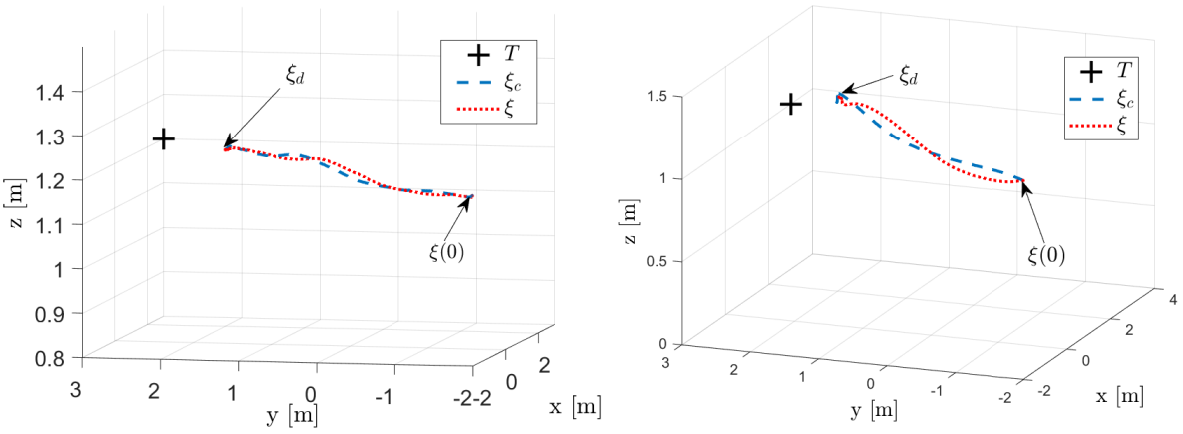


Figure 3.25 – 3D performance of the quadcopter. Left image corresponds when $\bar{t} = 5s$ and right one when $\bar{t} = 3s$.

Figure 3.25 shows the performance of the rotational dynamics of the quadcopter when tracking a desired trajectory. Notice that the left column images are the experimental test when $\bar{t} = 5s$ and the right ones considering $\bar{t} = 3s$. Observe that the dynamics is faster when $\bar{t} = 3s$. This due to the control parameters (for the rotational part) were re-tuned in order to converge at the desired finite-time.

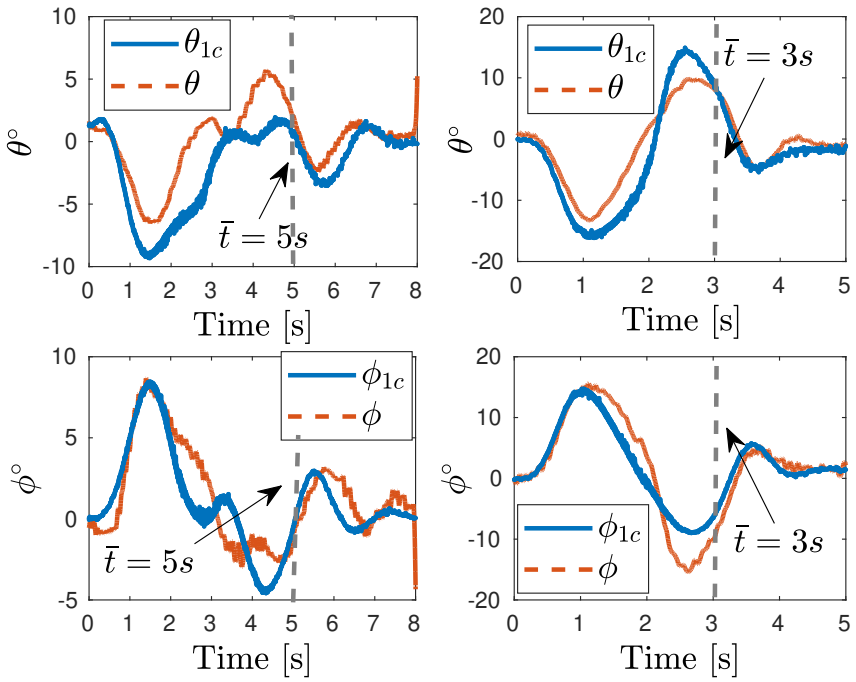


Figure 3.26 – Attitude behavior of the quadcopter when tracking a desired trajectory with $\bar{t} = 5s$ and $\bar{t} = 3s$ as desired finite-time convergence.

Figure 3.27 depicts the behavior of the control inputs. Notice that the "chattering" effect does not appears. For these experimental tests, we approximate the sgn function as an hyperbolic tanh function in order to avoid "chattering" effect. Therefore, the performance is softer with respect to that one of the simulation results.

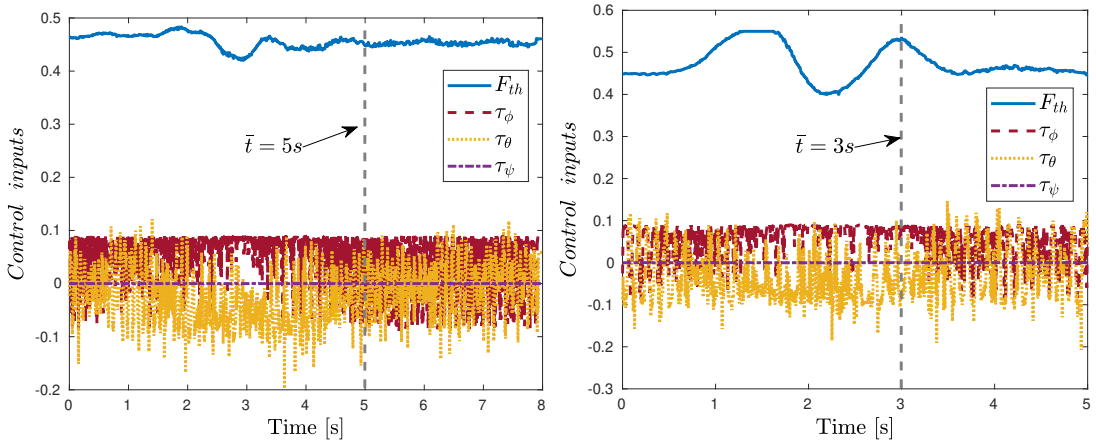


Figure 3.27 – Performance of the control inputs when considering $\bar{t} = 5s$ and $\bar{t} = 3s$ as desired finite-time convergence

3.4 Conclusions

In this chapter, some control techniques based on Euler angles and quaternion formalism were addressed: a brief overview evaluation of popular control algorithms (linear and nonlinear backstepping, nonlinear saturation control and the virtual fully actuated technique), a quaternion-based nonlinear backstepping controller and a Sliding Mode Control for finite-time convergence.

The goal of this chapter was to provide a unified and accessible analysis, placing popular control algorithms, quaternion-based controllers and with finite time convergence, into a proper context. Simulation and experimental results reveals that using the virtual fully actuated technique, over classical approaches, results in a smoother convergence to the desired reference and a smaller control effort demand. Nevertheless, when agile/aggressive movements or finite time convergence are required, quaternion-based algorithms and sliding mode-based controllers can quickly satisfy these demands, in comparison with classic controllers. After analysis, we consider that the choice between the controllers will depend on the problem requirements and the computational capacity of the embedded system.

Chapter 4 Part IV

4 Autonomous navigation algorithms

Most of the works currently found in literature about micro aerial vehicles, especially quadrotors, have received plenty of attention of the scientific community. Studies focus on control [162, 163, 164], state estimation [165], and planning [166, 28], have been conducted respectively, and many significant progresses have been achieved. Because of the strong needs in real applications, autonomous flight of quadrotors, which requires to address the perception, planning and control problem simultaneously, has been investigated and still continuously studied to improve its performance. Due to the noteworthy properties, particularly the structural simplicity, rapid maneuverability, the stable hovering capability, and low cost, a quadrotor is an ideal platform for navigating in confined and hazardous environments such as exploration [167], inspection [168], and search and rescue [169].

Going beyond these motivations and taking advantage of controllers detailed in the previous sections, this chapter presents three different alternatives for the autonomous control of quadrotor vehicles: a robust trajectory generation and tracking algorithm, a scheme for the aerial surveillance of Autonomous Ground Vehicles (AGVs) using a quadcotper vehicle and, a virtual control architecture for controlling robots remotely using a virtual reality environment.

This chapter is organized as follows: first, a path planning algorithm based on model predictive control with a bounded strategy is addressed in section 4.1. Experimental tests are carried-out in order to validate the proposed scheme. In section 4.2, a vision algorithm for target localization is presented. The goal in this section is to present the surveillance application between a quadrotor and a ground vehicle. Experimental tests are carried out to demonstrate the good performance of the control algorithm. Finally, a semi-autonomous navigation architecture using an immersive virtual environment is presented in section 4.3. Here, the propose is to reduce the stress in novice pilots when controlling a quadrotor vehicle. Two different scenario for the experimental tests are presented to confirm the good performance of the proposed architecture.

4.1 Path planning algorithm using MPC

In this section the path planning algorithm based on model predictive control (MPC) for the quadcopter system is discussed. For path generation and tracking, it is assumed that system (2.11) describing the dynamics of the quadcopter is at hover position with a desired altitude, using a controller with the form (3.26). Thus, let us define the mission set as follows: the aerial vehicle starts from an initial position, $\vec{P}(0)$, to a final one, $\vec{P}(t_f)$, where t_f denotes the final time. Solving the optimal trajectory planning problem (OTPP) involves the determination of the trajectory $\vec{P}_d = [\vec{\xi}_d, \dot{\vec{\xi}}_d]^\top$ and the corresponding control trajectory vector $\Delta\vec{P}_r$ which represents the input to the controlled prediction model. This vector will be the reference trajectory for the controller developed in Chapter 3 in terms of position and velocity.

The general control diagram used in this work is depicted in Figure 4.1. Here the reference trajectory will be generated by the Re-Planner algorithm based on MPC, the control algorithm block includes the control laws with the form (3.26) satisfying the path tracking objective, and the disturbance observer block (described in Chapter 5) providing robustness to the system.

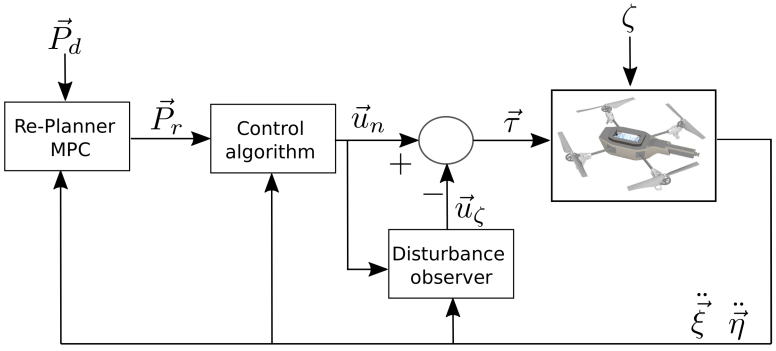


Figure 4.1 – Control-scheme diagram. $\vec{P}_r$ defines the control trajectory obtained from the algorithm MPC, $\vec{P}_d$ denotes the desired final values. The Re-Planner based on MPC generates the optimal feasible trajectory, and $\vec{\eta}$, $\vec{\xi}$ state for the vehicle's states. $\vec{u}_n$ represent the saturation nominal controller, $\vec{u}_\zeta$ expresses a compensate parameter from a disturbance observer and ζ possible external perturbations.

Our idea is to propose simple schemes which allows to compute the algorithms in the embedded system of the vehicle, the MPC approach used in this paper is based on the predictive control for linear systems without constraints. It can be notice from section 3.1.2 and using (3.22) that the controller can impose, using b_i, a linear behavior in the system (2.12). Thus, without lost of generality and for simplify further analysis, system (2.12) can be analyzed as a linear system and it can be represented as a discrete time state space of the form

$$\begin{aligned} \vec{x}(k+1) &= A(K,g)\vec{x}(k) + B(K,I)\Delta\vec{P}_r(k) \\ \vec{y}(k) &= C\vec{x}(k) \end{aligned} \tag{4.1}$$

where $\vec{x} \in \mathbb{R}^{12\times 1}$ is the state vector, $\Delta\vec{P}_r \in \mathbb{R}^{6\times 1}$ represents the control trajectory vector, $\vec{y} \in \mathbb{R}^{12\times 1}$ signifies the vector of measured outputs. $A(K,g) \in \mathbb{R}^{12\times 12}, B(K,I) \in \mathbb{R}^{12\times 6}$ are matrices with elements that depend on the gains defined in (3.22) and g, I were declared in (2.16).

Trajectory generation

The model predictive control approach solves a finite-time unconstrained optimal control problem in a receding horizon manner. Therefore, let H_p be the prediction horizon, H_u the control horizon and H_w the window horizon such that the path planning problem can be formulated as a linear optimization problem with a quadratic cost function with the form:

$$J(\vec{x}, \Delta\vec{P}_r) = \sum_{i=1}^{H_p} \|P(k+i) - P_d(k+i)\|^2_{Q(i)} + \sum_{i=0}^{Hu-1} \|\Delta P_r(k+i)\|^2_{R(i)}. \tag{4.2}$$

$$\vec{P}(k) = \begin{bmatrix} P(k+Hw|k) \\ \vdots \\ P(k+Hp|k) \end{bmatrix}, \quad \vec{P}_d(k) = \begin{bmatrix} P_d(k+Hw|k) \\ \vdots \\ P_d(k+Hp|k) \end{bmatrix}, \quad \Delta\vec{P}(k) = \begin{bmatrix} \Delta P_r(k|k) \\ \vdots \\ \Delta P_r(k+Hu-1|k) \end{bmatrix},$$

and the weighting matrices Q and R are given by

$$Q = \begin{bmatrix} Q(Hw) & 0 & \cdots & 0 \\ 0 & Q(Hw+1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q(Hp) \end{bmatrix}, \qquad R = \begin{bmatrix} R(Hw) & 0 & \cdots & 0 \\ 0 & R(Hw+1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(Hu-1) \end{bmatrix}.$$

The tracking error $\vec{\delta}_k = \vec{P} - \vec{P}_d$, is the difference between the state vector and a time varying reference and is penalized along with the control trajectory effort. The weighting matrices Q, R, penalizing the tracking error, are chosen to reflect the desired balance of their respective terms. The time reference is generated at each step by drawing a line in an Euclidean space from P_0 to the final target. That line is discretized into $N-1$ points where those points are used in sequence as $\Delta\vec{P}_r$.

The state's prediction can be formulated as follows [170]

$$\vec{P}(k) = \Psi\vec{x}(k) + \Upsilon\Delta\vec{P}_r(k-1) + \Theta\Delta\vec{P}_r(k) \tag{4.3}$$

for suitable matrices Ψ, Υ and Θ. Define

$$\vec{\varepsilon} = \vec{P}_d(k) - \Psi\vec{x}(k) - \Upsilon\Delta\vec{P}_r(k-1) \tag{4.4}$$

This can be thought of as a tracking error, in the sens that it is the difference between the future target trajectory and the free response of the system. If $\varepsilon = 0$, then it would indeed be correct to set $\Delta\vec{P}_r(k) = 0$. Rewriting (4.2) using (4.3) and (4.4)

$$J(\vec{x}, \Delta\vec{P}_r) = \left\|\Theta\Delta\vec{P}_r(k) - \vec{\varepsilon}(k)\right\|^2_Q + \left\|\Delta\vec{P}_r(k)\right\|^2_R \tag{4.5}$$

$$= [\Theta\Delta\vec{P}_r(k)^T - \vec{\varepsilon}(k)]Q[\Theta\Delta\vec{P}_r(k) - \vec{\varepsilon}(k)] + \Delta\vec{P}_r(k)^T R\Delta\vec{P}_r(k) \tag{4.6}$$

$$= \vec{\varepsilon}(k)^T Q\vec{\varepsilon}(k) - 2\Delta\vec{P}_r(k)^T\Theta^T Q\vec{\varepsilon}(k) + \Delta\vec{P}_r(k)^T[\Theta^T Q\Theta + R]\Delta\vec{P}_r(k). \tag{4.7}$$

Remark that (4.7) has the form

$$J(\vec{x}, \Delta\vec{P}_r) = cte - \Delta\vec{P}_r(k)^T \vec{G} + \Delta\vec{P}_r(k)^T H \Delta\vec{P}_r(k). \tag{4.8}$$

where

$$\vec{G} = 2\Theta^T Q\vec{\varepsilon}(k), \qquad H = \Theta^T Q\Theta + R.$$

and we can see that, neither $\vec{G}$ nor H depend on $\Delta U(k)$. To find the optimal $\Delta\vec{P}(k)$ we can now find the gradient of $J(k)$ and set it equal to zero. From (4.8) we find

$$\nabla_{\Delta\vec{P}_r(k)} J = -\vec{G} + 2H\Delta\vec{P}_r(k). \tag{4.9}$$

then the optimal set of future input moves is

$$\Delta\vec{P}_r(k)_{opt} = \frac{1}{2} H^{-1}\vec{G}. \tag{4.10}$$

Equation (4.10) guarantees a stationary point, but is not enough to guarantee a minimum. In order to guarantee the minimum, the derivative of the gradient $\nabla\Delta\vec{P}_r(k)J$ with respect to $\Delta\vec{P}_r(k)$ must be obtained, i.e., the Hessian of $J(\vec{x}, \Delta\vec{P}_r)$:

$$\frac{\partial^2 J}{\partial\Delta\vec{P}_r(k)^2} = 2H = 2(\Theta^\top Q\Theta + R) \tag{4.11}$$

Assuming that $Q(i) \geq 0$, this ensures that $\Theta^\top Q\Theta \geq 0$. Thus, if $R > 0$ then the Hessian is certainly positive-definite, which is enough to guarantee that we have a minimum.

Formulation as a Least-Squares problem

The optimal solution, as expressed in (4.10), should never be obtained by computing the inverse of H. The matrix Θ is often ill-conditioned, which can result in H being ill-conditioned. The best way of computing the solution is by solving it as a least-square problem. It is also a way which gives some additional insight.

Since $Q \geq 0$ and $R > 0$, we can find matrices S_Q and S_R which are their square-roots:

$$S_Q^T S_Q = Q, \qquad S_R^T S_R = R$$

Now consider the vector

$$\begin{bmatrix} S_Q(\Theta\Delta\vec{P}_r(k) - \vec{\varepsilon}(k)) \\ S_R\Delta\vec{P}_r(k) \end{bmatrix} \tag{4.12}$$

We shall show that the squared length of this vector, or equivalently, the sum of squares of its elements, is the same as the cost function $J(\vec{x}, \Delta\vec{P}_r)$, so that $\Delta\vec{P}_r(k)_{opt}$ is the value of $\Delta\vec{P}_r(k)$ which minimizes this length.

Therefore, it yields

$$\left\| \begin{bmatrix} S_Q\left(\Theta\Delta\vec{P}_r(k) - \vec{\varepsilon}(k)\right) \\ S_R\Delta\vec{P}_r(k) \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} S_Q(\vec{P}(k) - \vec{P}_d(k)) \\ S_R\Delta\vec{P}_r(k) \end{bmatrix} \right\|^2 \tag{4.13}$$

$$= [\vec{P} - \vec{P}_d(k)^T] S_Q^T S_Q [\vec{P}(k) - \vec{P}_d(k)] + \Delta\vec{P}_r(k)^T S_R^T S_R \Delta\vec{P}_r(k) \tag{4.14}$$

$$= \left\|\vec{P}(k) - \vec{P}_d(k)\right\|_Q^2 + \left\|\Delta\vec{P}_r(k)\right\|_R^2 \tag{4.15}$$

$$= J(\vec{x}, \Delta\vec{P}_r) \tag{4.16}$$

Therefore $\Delta\vec{P}_r(k)_{opt}$ is the least-squares solution of the equation

$$\begin{bmatrix} S_Q(\Theta\Delta\vec{P}_r(k) - \varepsilon(\vec{k})) \\ S_R\Delta\vec{P}_r(k) \end{bmatrix} = 0 \tag{4.17}$$

or, equivalently, of:

$$\begin{bmatrix} S_Q\Theta \\ S_R \end{bmatrix} \Delta\vec{P}_r(k) = \begin{bmatrix} S_Q\vec{\varepsilon}(k) \\ 0 \end{bmatrix} \tag{4.18}$$

We can see equation (4.18) has the form $A\Theta = b$ and can be solved in a least-square sense using the QR algorithm. Although formally this solution is the same as $\theta_{opt} = (A^T A)^{-1} A^T b$, which gives the equation (4.10).

Furthermore, in accordance with the receding horizon strategy, we only use the part of the solution corresponding to the first step. Thus, from (4.10) it is possible to observe that the only part of this solution which changes from step to step is the tracking error $\varepsilon(k)$. Consequently, the predictive controller for the unconstrained problem and with full state measurement is defined by

$$P_r = [I_l, 0_l, \cdots, 0_l]\, H^{-1}\Theta^\top Q \tag{4.19}$$

We point out that the correct way of computing (4.19) is

$$P_r = [I_l, 0_l, \cdots, 0_l] \begin{bmatrix} S_Q\Theta \\ S_R \end{bmatrix} \setminus \begin{bmatrix} S_Q\vec{\varepsilon}(k) \\ 0 \end{bmatrix} \tag{4.20}$$

This essentially works for the following reason: if we had $\vec{\varepsilon}(k) = [1, 0, \cdots, 0]^\top$ then it would effectively have only the first column of S_Q on the right side of (4.10). $\vec{\varepsilon}(k) = [0, 1, 0, \cdots, 0]$ would effectively give the second column, etc. Since $\vec{\varepsilon}$ enters the solution linearly, it is only necessary to solve (4.10) with these columns on the right hand side. This is very efficient, since most of the work involved is computing the QR decomposition of the left hand side. In this work, the householder QR decomposition algorithm is used to solve the unconstrained problem (4.20).

The steps to solve the cost function (4.2) are presented in algorithm 1. In this thesis, the explicit solution for generating the optimal trajectory using the x, y, z position as the control trajectory input is addressed.

Algorithm 1 General statement of the MPC algorithm

initialize MPC-LS with nominal a control trajectory $\Delta\vec{P}_r$
$\vec{P}(0) \leftarrow$ starting position
$\vec{P}_d(t_f) \leftarrow$ desired position
Generate the linear controlled model (4.1)
While ($\vec{P} < \vec{P}_d$)
-Create the augmented MPC model of (4.1)
-Minimize the cost function (4.2) by solving the LS sense using the 'QR' algorithm
-Apply the control trajectories
-Update $\vec{P}$
-Update $\Delta\vec{P}_r$
end

4.1.1 Experimental results

In this section, a helical trajectory with wind and manual disturbances was selected to validate the performance of the presented path planning algorithm. The experimental test was developed using the quadtotor vehicle AR Drone 2 using the open source software Fl-AIR and an OptiTrack motion capture system.

The set up of the experiment is as follows: the quadrotor takes off and follows a hold position $\vec{\xi}0 = (1.5, 0, 0.5)\ m$. Then the desired final trajectory is generated by means of the algorithm1. It is mandatory when the vehicle is navigating through the reference trajectory, to compute the optimal trajectory at each step by means of the MPC. Moreover, a first fan was turned on once the reference trajectory started. After that, manual disturbances were applied at different times during the mission. The parameters used for this test can be found in Table 4.1. A video of the experimental results can be watched at: https://youtu.be/3G7UVYWsyBs.

Table 4.1 – Experimental parameters

$\tau_{i:\phi,\theta}$	***MPC***	
$k_{pi:x,y} = 0.23$	$x_i = 0$	$x_f = 0$
$k_{di:x,y} = 0.13$	$y_i = -3$	$y_f = -2$
$\sigma_{i:x,y} = 0.1$	$Q = 0.1$	$R = 0.0002$
$\sigma_{i:\dot{x},\dot{y}} = 0.2$	$H_p = 10$	$H_u = 3$
$k_{pi:\phi,\theta} = 0.5$		
$k_{di:\phi,\theta} = 0.1$		
$\sigma_{i:\phi,\theta} = 0.1$		
$\sigma_{i:\dot{\phi},\dot{\theta}} = 0.3$		

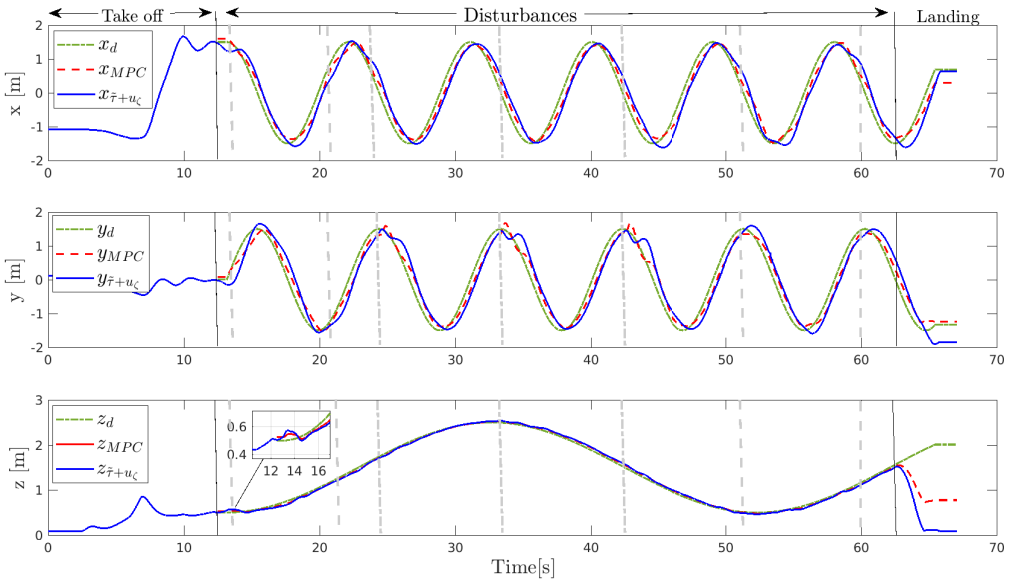


Figure 4.2 – Evolution of the path tracking in x, y, $z-$axis with wind and manual disturbances.

Figures 4.2 − 4.4 depict the trajectory generation and the quadrotor performance when following the desired trajectory subject to wind and manual disturbances. The wind effect is applied at $t = 19s$, $t = 48s$ and $t = 57s$ but not necessarily constant due the vehicle is following a $3D$ reference trajectory. This can be observed properly in Figure 4.2. Remark that at $t = 23s$ and $t = 34s$ approximately, a manual disturbance was applied while the vehicle is ascending. This performance can be seen in Figure 4.3. In this graph, it is roughly depicted when the wind (two times) and manual (two times) perturbation were applied. These effects were compensated in a robust manner by the compensation of the disturbance observer developed in section 5.

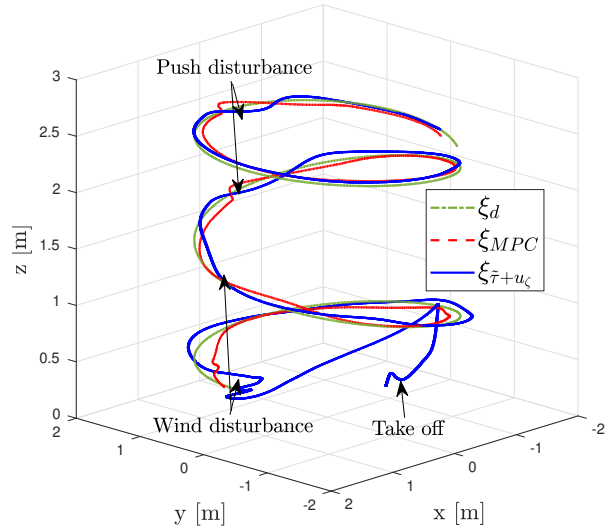


Figure 4.3 – Flight test when the quadcopter follows the helical trajectory in ascending phase in presence of wind and manual disturbances.

Figures 4.4 illustrates the descending behavior of the vehicle. This performance can be sees in the $z-$axis around $t = 36s$. Although the gravity force actuates in the drone, and manual perturbation is applied, the MPC solves the shortest path forcing the control law to avoid oscillations in the altitude position when the vehicle is descending.

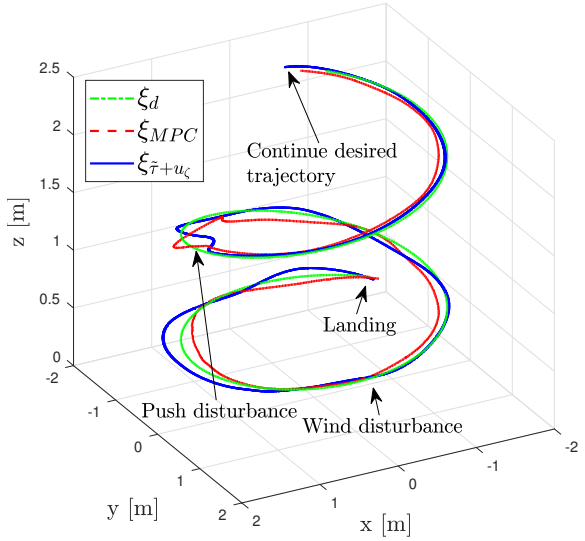


Figure 4.4 – Flight test when the quadcopter follows the helical trajectory in descending phase in presence of wind and manual disturbances.

4.2 Vision algorithm for target localization: a case study for autonomous vehicles surveillance

Aerial visual tracking is a major hurdle to tackle for feasible prototyping of an autonomous ground vehicle (AGV). These vehicles are subject to non-holonomic constraints since rear wheels are fixed preventing to produce lateral velocity in any direction. However, when we drive such a car, we pick visual landmarks ahead (for instance, as showed in Fig. 4.5, a corner at the end of the block) in the visual field to plan how to command the steering wheel to intersect ahead such landmark. The result is smooth car trajectories pointing ahead with small admissible turning ratio of the steering wheel angle. These trajectories can be modeled as vector fields pointing to a landmark (known as the contour), for instance, when we need to turn in the next corner of the block, we solve what the steering wheel angle is, based on the corner ahead of us, as if we were drawing smooth vector fields.

Then, if we neglect that, such driving motion corresponds to an autonomous car that mimics human driving, drawing smooth vector fields. The problem of aerial surveillance becomes the aerial tracking of the car center of mass (x, y) and yaw ψ angle that matches the heading direction of the car, at a given altitude z. However, such aerial tracking of (x, y, z, ψ) requires solving the vector field and underactuated visual tracking. Therefore, we aim exploiting structural properties of the car, such as non holonomy, and the vector field control to mimic driving, and consider underactuation of the quadrotor, to establish a tractable problem statement.

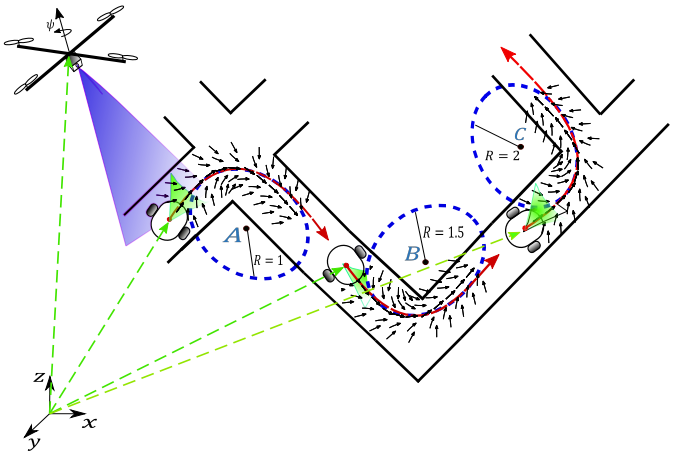


Figure 4.5 – Proof of concept of the problem: The driver commands the vehicle in urban set, picking up the next contour A, then B, then C to plan smoothly ahead, as if navigating in smooth vector fields; then, facilitating tracking with airborne monocular camera, since height can be regulated efficiently, and assuming target remains in the FoV.

To this end, we assume that the car evolves in the (x, y) plane subject to non-holonomic constraints while tracking smooth vector fields designed using contour ahead of the road in ψ direction, thus such vector fields provide the desired admissible trajectories to the car, and an underactuated quadrotor equipped with airborne monocular camera is considered to track the car at a given desired altitude z_d, see Figure 4.6. Thus, let us consider that the visual tracking of the mobile robot smoothly draws a vector field converging to a contour. This contour relies in the field of view of the airborne camera mounted on an underactuated drone. Then, the following assumptions are required:

1. The AGV mimics motion of a car driven by a human, implying that its CoM draws a smooth vector field toward the contour ahead;

2. The AGV's path curvature is finite, implying bounded yaw angle that complies with the non-holonomic constraint and a bounded steer wheel angle.

3. Quadcopter has underactuation in (x, y) coordinates, however the controlled Degrees of Freedom (DoF) are yaw ψ angle, altitude z, as well as (x, y) motions, where the latter are controlled indirectly using roll and pitch angles.

4. Monocular camera is along the optical axis collinear to z in body coordinates, where the mobile robot lies in its FoV. Since the quadcotper altitude can be controlled quickly and independently, 2D visual servoing suffices to capture AGV's CoM.

5. For experiments, we consider the 6 DoF of Parrot AR.Drone with monocular camera and airborne processing only, [171], to track in the image plane a differential driven mobile robot.

6. Moreover, the hardware, software and firmware of airborne sensors have limitations, including monocular camera, noisy, delayed and quantized measurements yielding approximation of the required virtual camera transformations; then, we assume that the fast quadcopter actuation and smoothness of trajectories produce small angles in roll and pitch, then the diffeomorphism of the virtual camera projection is almost an identity. This assumption is substantiated since the proposed quadrotor controls ensure asymptotic convergence to the vicinity of stability along very smooth trajectories.

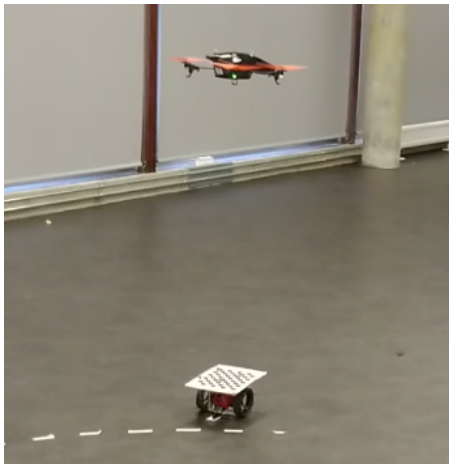


Figure 4.6 – Experimental test-bed showing the AR Drone 2.0 as the quadrotor, and the Jumping Sumo as the mobile robot (with a chess board on top for image processing).

The trajectories of the non-holonomic differential wheeled mobile robot, under Velocity Field Control in inertial frame, are captured by the monocular camera as an image-based field in aerial body frame. Such image is processed on board, without relying data to ground station, to produce the desired image-based trajectories for the controller of the underactuated quadrotor dynamics. The mobile robot differential kinematics is processed under kinematic controller in a ground station, then velocity control command are sent wireless. Trajectories are processed with a pinhole camera to produce an image-vector field. Then, considering the full non-linear dynamics, the quaternion-based backstepping control (3.97) is implemented to closed-loop, under underactuation for well-posed underactuated coordinates.

Vision algorithm

A chess-pattern target was placed onto the mobile robot to process its orthogonal axis in its frame by high density of corner, assuming a background floor without cluttered lines or corners. The camera together with the processing unit of the AR Drone 2.0 quadrotor detects corners with the following:

1. Finds the 64 most confident corners of chess board with a pinhole camera image using the Shi-Tomasi algorithm [172].

2. Determine correlation between images using the Lukas Kanade algorithm, [173].

3. Determine centroid based on matched chess corners, as follows

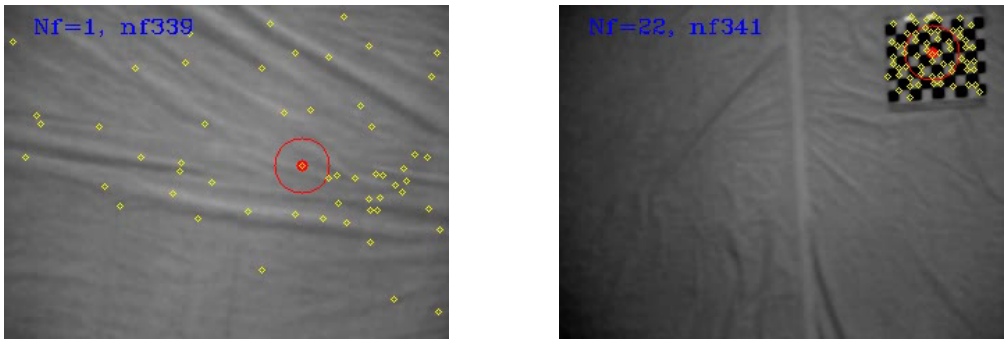
$$p_t = \sum_{i=0}^{N} p_{f_i} / N \tag{4.21}$$

where p_t is the estimated centroid of the target, p_{f_i} is the position of each matching corner feature, and N is the total number of matching corners.

4. Update matching corners, then determine the centroid (4.21).

5. Target is detected if area, centered in the centroid with radius of 20 pixels, has at least 15 corners.

Notice from Figure 4.7a, there is no target detected, even if a centroid is calculated with only one corner, $Nf = 1$. Nevertheless in Figure 4.7b, algorithm determines that the target has been detected for $Nf = 22$ corners.



(a) Target is not the FoV.

(b) Target is in the FoV.

Figure 4.7 – Frames when searching for the target

The estimation of the optical flow is performed on a dedicated DSP to reduce the computational cost to the main control board, whose processing time takes between 15 and 20 ms.

After computation, a Kalman filter was implemented for improving the robustness of processing. To determine the direction of centroid, see Fig. 4.8, let the vector be

$$\vec{p}_t = \begin{bmatrix} u_c \\ v_c \end{bmatrix} \tag{4.22}$$

which can be used, together with its image approximation of velocity, $[\dot{u}_c,\ \dot{v}_c]^T$, for control purposes, by considering that image axes are parallel to x and y axes of body fixed frame.

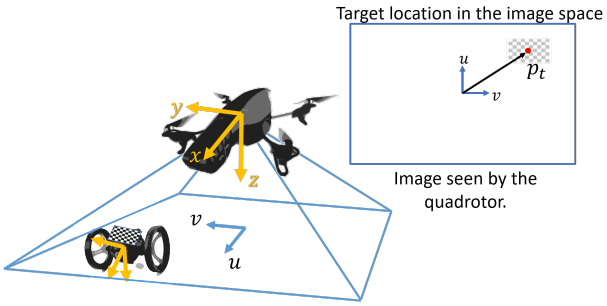


Figure 4.8 – Image position of the mobile robot frame and the camera frame.

The autonomous vehicle as a mobile differential robot

Consider a $2D$ terrestrial differential wheeled robot steered by the difference of angular wheel velocity connected along a perpendicular axis to longitudinal motion, see Figure 4.9. The state equation is as follows

$$\dot{\vec{X}}_s = A\vec{\omega}, \tag{4.23}$$

where $\vec{X}_s = [x_s, y_s, \theta_s]^T \in \mathbb{R}^3$ is the robot pose, for (x_s, y_s) representing position its CoM, and θ_s its orientation, both with respect to the inertial reference frame, and $\vec{\omega} = [\omega_1,\ \omega_2]^T$, the angular speed of the robot's wheels.

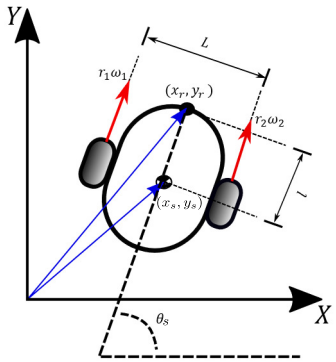


Figure 4.9 – Kinematic of a differential robot as an abstract representation of an AGV.

The flow matrix is given by

$$A = \begin{bmatrix} \frac{r_1}{2}\cos\theta_s & \frac{r_2}{2}\cos\theta_s \\ \frac{r_1}{2}\sin\theta_s & \frac{r_2}{2}\sin\theta_s \\ -\frac{r_1}{L} & \frac{r_2}{L} \end{bmatrix}, \tag{4.24}$$

which maps generalized angular velocities $\vec{\omega}$ to operational velocity $\dot{\vec{X}}_s$, with $r_1 = r_2$ the radius of the left and right wheels, respectively, and L the length to the center of each wheel.

Vector fields as desired smooth trajectories

Velocity Fields stand for Cartesian vector distributions, or fields, and describe kinematic motion of a moving particle, i.e. the instantaneous changes in direction toward the objective, or contour. It is useful in our case, if we consider that AGV driver commands the vehicle to go to a target, or contour, and such driver steers the vehicle within prescribe standards, thus producing a smooth maneuverings, that can be modeled as a vector field that converge to the contour. Then, we assume that the driver mental map commands toward the final destination by picking locally targets as soon as they appear in our FoV. Once the driver picks a target ahead, for instance to turn in the next corner, we somehow compute how to turn the AGV avoiding obstacles to pass through until converge tangentially to the contour. Once we determine this objective is achieved, we pick next target (contour) in the next block. Thus, this abstraction stands for how we plan next trajectory based on local contour in our FoV. Similar to this phenomena, we assume that the AGV produces such vector field, then the surveillance problem is to produce a visual servoing to maintain the AGV within its FoV.

To this end, consider a parametric curve, $G(s)$, where s stands for arc length, or contour ahead in the FoV. The problem is to determine the $\mathbb{R}^2$ directional components $(\vec{N}_n(s), \vec{T}(s))$ of the field as a radial attraction to the curve s, or contour, where $\vec{N}_n(s)$ and $\vec{T}(s)$ stand for the Normal and Tangential components, respectively, of the field in $\mathbb{R}^2$, whose resultant points at the target correspond to $G(s) = 0$. There are several methods in the literature to produce such $(\vec{N}_n(s), \vec{T}(s))$ that smoothly converge to $G(s) = 0$, the arc to turn in the corner asymptotically with uniform speed, as we usually compute when we drive our cars through urban settings. The method developed considers to pick the objective path, or contour, as a parametric curve to compute from position data of such contour, the velocity vector that intersects ahead such contour $G(s) = 0$, see Figure 4.10.

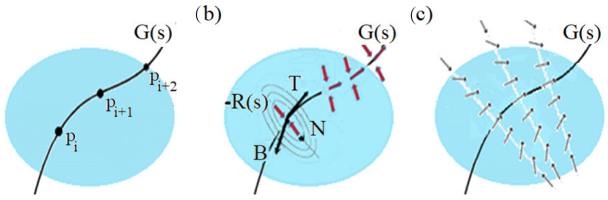


Figure 4.10 – Three steps Velocity Fields design method based on differential geometry properties of the desired trajectory:(a) parametric curve of motion objective, (b) main directional components of the field, (c) calculation of the velocity vectors.

Thus, consider $\vec{v} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ the velocity vector that depends on the position of a given point $\vec{r}$, with $\vec{N}_n(s) = -\vec{R}_n(s)$ (where $\vec{R}_n(s)$ is a radial unit vector) and $\vec{T}(s)$, components gradually transitioning from a given position CoM to an attractive curve $G(s) = 0$ standing for the turning in the corner. Such transition uses a variable coefficients as a function of the distance to the curve. Then, consider

$$\vec{v}(r) = \mu_{close}(d_r)\vec{T}(s) + \mu_{far}(d_r)(-\vec{R}_n(s)) \tag{4.25}$$

where $\mu_{close}(d_r)$ and $\mu_{far}(d_r)$ are the variable coefficients depending on the distance d_r to the curve $G(s) = 0$. To keep a uniform speed motion along the attractive trajectories, $||\vec{v}(r)|| = 1$, consider that the coefficient of the weighted sum are defined using sinusoidal functions,

$$\mu_{close}(d_r) = \cos(c_1 d_r); \mu_{far}(d_r) = \sin(c_1 d_r) \tag{4.26}$$

where c_1 is a normalizing constant such that $c_1 d_r \in [0, \frac{\pi}{2}]$. Recalling the arc length s along $G(s)$ is determined by $\vec{r}$ at its closest point, $\vec{v}$ becomes

$$\vec{v}(r) = \cos(c_1 d_r)\vec{T}(r) + \sin(c_1 d_r)(-\vec{R}_n(r)) \tag{4.27}$$

thus $\vec{v}(r)$ is a gradual uniform speed transition from the attractive to the guiding direction as the position gets closer to the curve, and represents the reference of the desired velocities, $\begin{bmatrix} \dot{x}_d & \dot{y}_d \end{bmatrix}^T$, for the differential robot.

Without loss of generality, aiming at illustrating our proposal, let the desired trajectory of the mobile robot, be a 2D circumference. This simplifies the velocity field design because the radial and the tangential vectors to the curve can be calculated with respect to the center of the circumference, $r_c = (x_c, y_c)$ and its radius, ρ, then

$$\vec{T}(x,y) = -\operatorname{sgn}(\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} - \rho) \cdot \frac{(y_s - y_c, -(x_s - x_c))}{\sqrt{(x_s - x_c)^2 + (y_G - y_c)^2}},$$
$$\vec{R}_n(x,y) = -\operatorname{sgn}(\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} - \rho) \cdot \frac{(x_s - x_c, y_s - y_c)}{\sqrt{(x_s - x_c)^2 + (y_G - y_c)^2}},$$

such that the velocity field becomes

$$\vec{v}(r) = \cos(c_1 d_r)\vec{T}(r) + \sin(c_1 d_r)(-\vec{R}(r)) \tag{4.28}$$

with $c_1 = \frac{\pi}{2r}$.

Integration of the overall scheme

The image-based control law is based on the position control (3.80)

$$\vec{U} = \begin{bmatrix} -(\dot{u_c} + u_c) - \dot{u_c} - \frac{\alpha_{1_x}}{\alpha_{2_x}} u_c \\ -(\dot{v_c} + v_c) - \dot{v_c} - \frac{\alpha_{1_y}}{\alpha_{2_y}} v_c \\ -e_{z_2} - z_2 - \frac{\alpha_{1_z}}{\alpha_{2_z}} e_{z_1} + g \end{bmatrix} \tag{4.29}$$

where $\begin{bmatrix} u_c & v_c \end{bmatrix}^T$, and $\begin{bmatrix} \dot{u_c} & \dot{v_c} \end{bmatrix}^T$, are respectively image coordinates of the target's centroid, and their rate of change. α_{1_i}, α_{2_i} are positive constants and e_{z_2} and e_{z_1} are the position and velocity errors of the altitude system, defined in section 3.2 and it is controlled independently, thus it can be considered that visual projection occurs in 2D, at a given altitude $z = z_d$.

Then, the well-posed desired quaternion, $\boldsymbol{q_d}$, is computed from (3.97) using the normalized vector of (4.29) and AGV's yaw angle, which represents such mapping.

To design the kinematic control of the AGV, forward kinematics yields the following map

$$\vec{\Xi} = \begin{bmatrix} x_s + l\cos\theta_s \\ y_s + l\sin\theta_s \end{bmatrix} \Longrightarrow \dot{\vec{\Xi}} = \begin{bmatrix} \dot{x}_s - l\dot{\theta}_s\sin\theta_s \\ \dot{y}_s + l\dot{\theta}_s\cos\theta_s \end{bmatrix} \tag{4.30}$$

By using the non-holonomic constraint, the change of coordinates $\dot{x}_s = \tilde{u}_1\cos\theta_s$, $\dot{y}_s = \tilde{u}_1\sin\theta_s$, $\dot{\theta}_s = \tilde{u}_2$, allows the differential kinematic map to be written as follows,

$$\begin{aligned} \dot{\vec{\Xi}} &= \begin{bmatrix} \cos\theta_s & -l\sin\theta_s \\ \sin\theta_s & l\cos\theta_s \end{bmatrix} \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} \\ &= B\vec{u}_{GV} \end{aligned} \tag{4.31}$$

Since B is a positive definite matrix, consider the kinematic control as follows

$$\vec{u}_{GV} = \gamma B^{-1} \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} \tag{4.32}$$

where $\gamma > 0$ is used to scale the desired velocity $\begin{bmatrix} \dot{x}_d & \dot{y}_d \end{bmatrix}^T$ of the field. Then, substituting (4.32) into (4.31) yields $\dot{\vec{\Xi}}(t) = \gamma \begin{bmatrix} \dot{x}_d & \dot{y}_d \end{bmatrix}^T$, which ensures that (4.32) computes the desired angular wheel velocity based on the given velocity field components. Notice that γ becomes an extra degree of freedom needed in practice to adjust engineering units and for calibration purposes, otherwise it can be set $\gamma = 1$.

4.2.1 Experimental tests

To demonstrate the effectiveness of the proposed method two experimental tests are discussed in this section. The ground vehicle used is a Jumping Sumo, and the quadrotor is the AR Drone 2.0 with monocular camera pointing downwards, both from the Parrot company. A custom-made programming platform is configured to command both robots based on personal computers, giving rise to a platform called Fl-AIR, which decodes communication protocols of peripheral devices by using a custom made software, including the closed-loop control [160].

The AR Drone 2.0 uses a board with an ARM Cortex A8 1GHz processor with 1GB of RAM, running the controller programmed in C++, for a IMU update at 5 ms, to handle all system processes, including control outputs, interruptions, data fusion and communication to ground station. The mobile sumo robot's Wi-Fi node enables control command in velocity mode using a Broadcom BCM43526 transceiver, but for higher range, a pair of Skyworks SKY85803 Wi-Fi front-ends is used. It has an MPU-6050 invenSense 6-Axis gyroscope and accelerometer.

Two small DC gear motors drive this mobile robot up to 4.3 MPH. A MoCap system, from Optitrack, composed of 24 cameras, is used at a rate of $100Hz$ to provide inertial measurement of sumo robot only. The parameters of the mobile robot tuning of feedback gains are shown in Table 4.2.

Table 4.2 – Experimental parameters for the control law (4.29) and the ground vehicle control (4.32).

$\alpha_1 = 0.8$	$L = 0.092\ m$	$g = 9.81$	$\gamma_1 = 0.8$
$\alpha_2 = 0.1$	$l = 0.08\ m$	$z_d = 1.5m$	$\gamma_2 = 0.3$
$m = 0.420$	$r_1 = r_2 = 0.055\ m$	m = 0.18 Kg	–

Scenario 1: target detection test

In this test, arbitrary initial conditions in the longitudinal and lateral positions were considered for both robots and a desired altitude of $z_d = 1.5m$ for the quadcopter. The goal of this test is to demonstrate the well-performance of the vision algorithm when the ground vehicle is turning in the $z-$axis and the algorithm uses the estimated velocities instead that ones coming from the classical optical flow. Hereafter, variables with subscript "f" and superscript "$\wedge$" represent the measures of the traditional optical flow data and the estimation of the states based on the Kalman filter, respectively.

Figures 4.11 − 4.13 show the performance of the quadcotper and the vision algorithm when tracking the ground vehicle. Remark from Figure 4.11 the x and y centroid positions are obtained in the image plane. In addition, it can be observed that these positions are slightly noisy when using only the traditional Lukas Kanade optical flow.

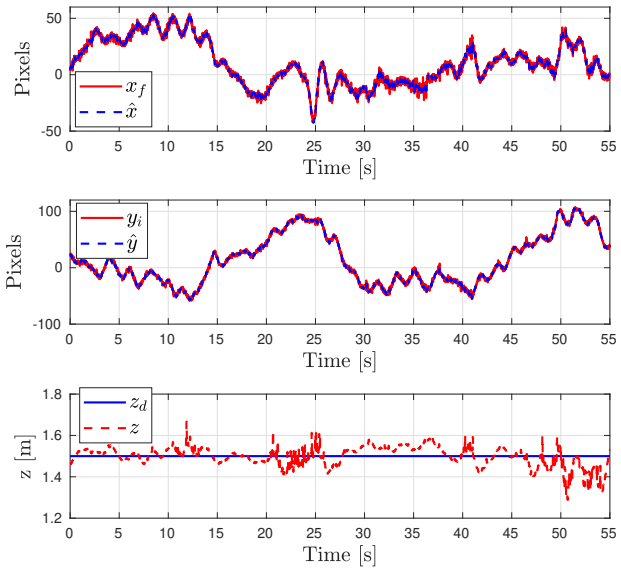


Figure 4.11 – Positions x and y of the centroid using the optical flow data and the kalman filter. The z position is measured using the Optitrack camera system.

Observe that, when using the Kalman filter the velocity measure is less noisy. This can be confirmed in Figure 4.12. Observe in this figure that, the velocities computed from the optical flow data are notably noisy due the fact the ground vehicle is turning on its own axis ($z-$axis). However, computing the velocity using the Kalman filter gives smoothness in the measure allowing the vehicle keeping the desired hover position, even if the ground robot is turning. This can be verified in Figure 4.13 where the control laws are showed. From this figure, it is possible to see that the vehicle keeps the desired hover position during the test with a slightly movement in the roll control as a result that the ground vehicle moved while rotating.

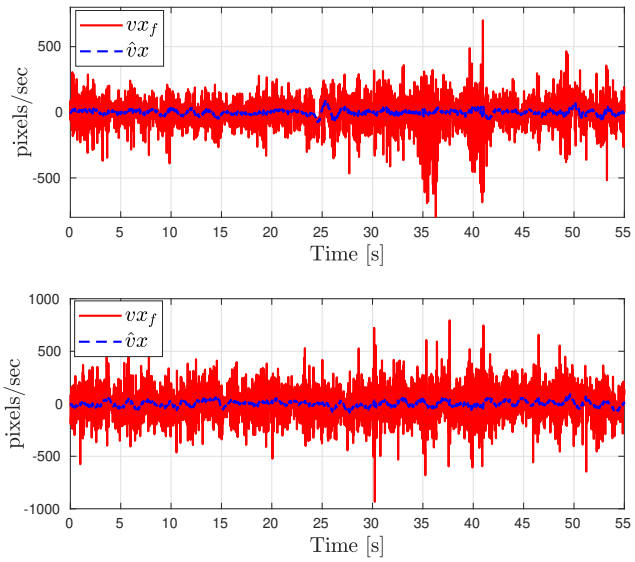


Figure 4.12 – Velocities of the target in the image plane.

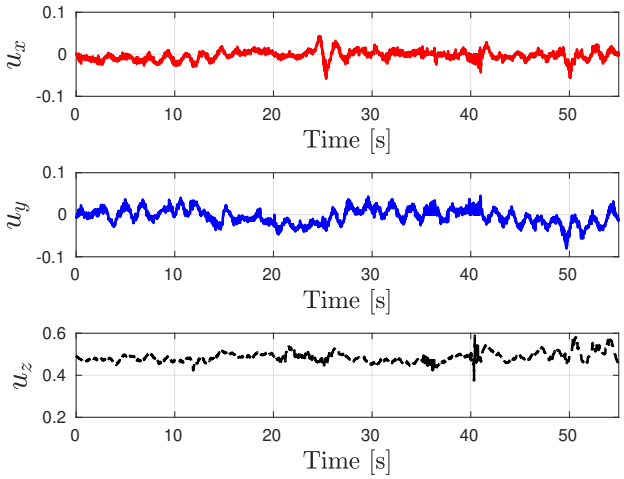


Figure 4.13 – Control inputs for the quadcopter vehicle.

Scenario 2: tracking an autonomous ground vehicle

The goal of this test is to prove the effectiveness of the proposed control algorithm when the ground robot is moving. For that, in this scenario initial conditions of $\vec{\xi}_u(0) = \vec{\xi}_s(0) = [-0.2, -1]^T m$ were considered for both robots, and a desired altitude of $z_d = 1.5m$ for the quadrotor. Then, three circumferences are defined as the smooth contour of radius $r_{c_1} = 1m$, $r_{c_2} = 1.5m$ and $r_{c_3} = 2m$, starting from an arbitrary initial condition in the plane. While the sumo starts its motion, the drone vision algorithm detects the target (the centroid of the visual landmarks placed over the mobile robot). Then, the velocity field algorithm computes online the kinematic controller to provide its smooth velocity field components depending of the instantaneous position with respect to the contour, see Fig. 4.14. Figure 4.15 illustrates the performance of the vision algorithm when following the smooth velocities imposed by the ground vehicle. Remark from these figures when the radius of circumference was changed. The video of the experimental test can be seen in https://www.youtube.com/watch?v=zB6dsH3zVg0 .

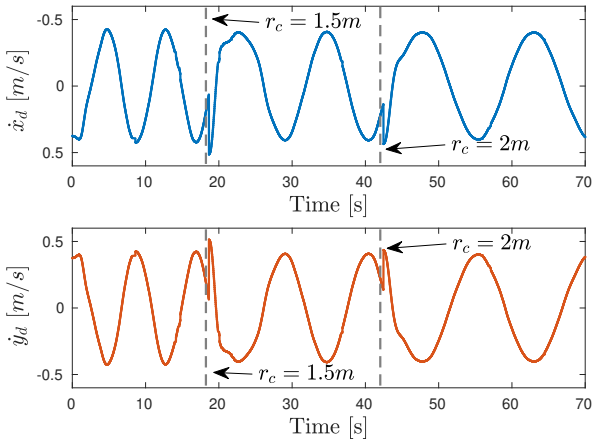


Figure 4.14 – Desired velocity field components of the ground robot.

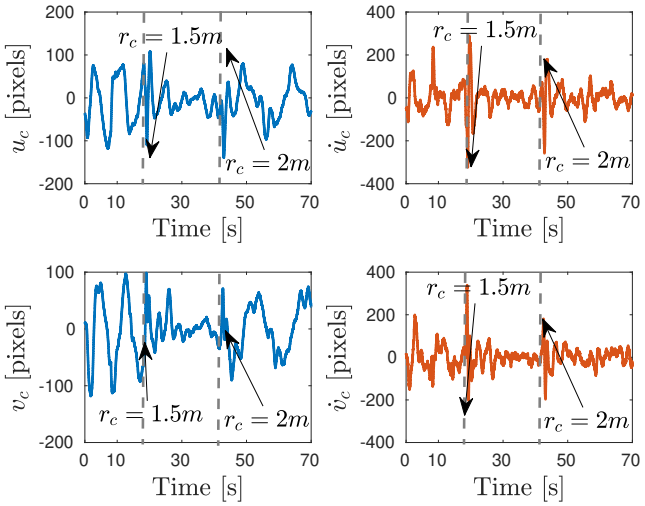


Figure 4.15 – Position of the centroid $\vec{p}_t$ and its estimated velocity.

Figures 4.16 and 4.17 show the resultant path of the quadcotper immersed into a visual velocity field that converges to the ideal 2D mobile robot's velocity field. The positions of the quadcopter and the ground autonomous vehicle are represented by x_u, y_u and x_s, y_s, respectively. These figures also show the relative position of quadcotper and the mobile robot. Remark from these figures that the error in both axes is relatively small allowing the quadcotper keeps the ground robot on the FoV. It is worth to mention that in order to get the well performance of the vision algorithm showed in Figure 4.15, the desired altitude was set constant at $z_d = 1.2m$. Furthermore, the shortest distance where the quadcopter can continue detecting the AGV was $z = 0.8m$. Nevertheless at this distance the ground vehicle can come out quickly from the FoV. Contrary, the highest altitude was $z = 1.5m$. This helps to maintain the AGV in the FoV. However, more features are needed for computing the centroid and therefore more computational burden is required. In addition, remember the optical flow camera of the Ar.Drone 2.0 and its embedded system have computational limits.

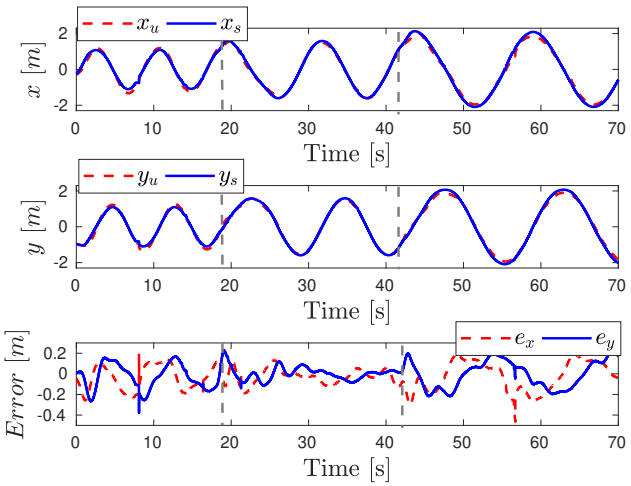


Figure 4.16 – Cartesian position trajectories and errors of the mobile robots.

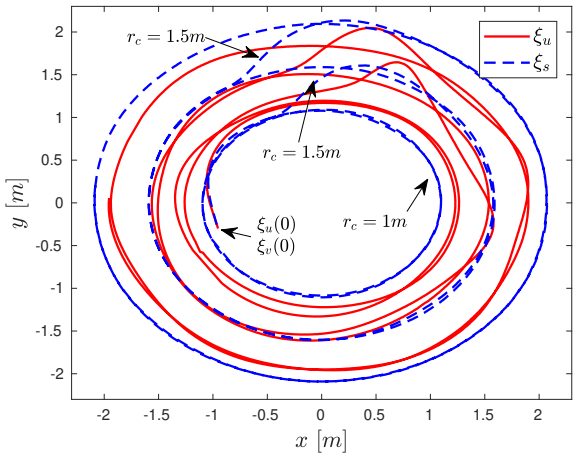
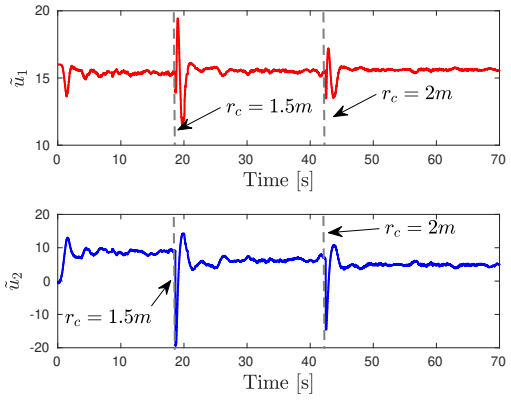
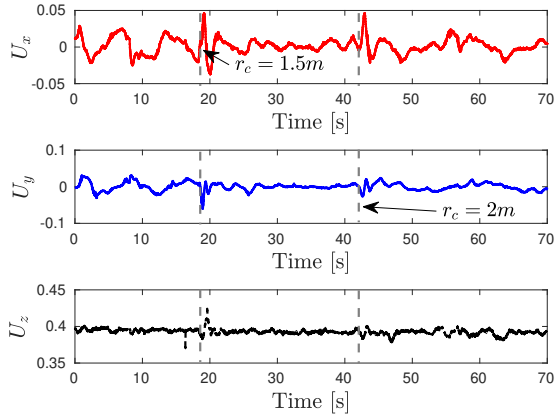


Figure 4.17 – Performance of both robots in the xy plane. These signals are provided by the Optitrack system.

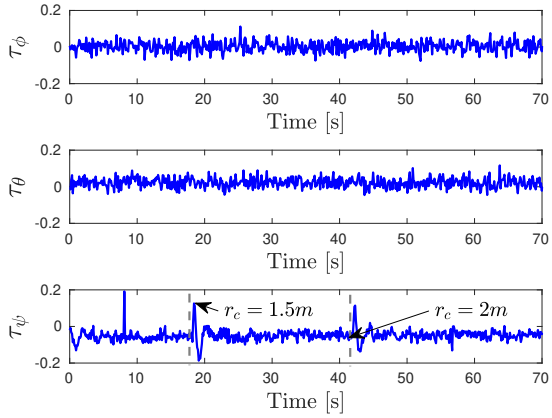
The mobile ground robot control signals and that of the quadcopter are shown in Figure 4.18. Notice the virtual control input of the ground robot vehicle are depicted in Figure 4.18a, according to the change of coordinates developed in (4.31). Figure 4.18b illustrates the virtual control input $\vec{U}$ that makes $\boldsymbol{q} \rightarrow \boldsymbol{q}_d$ and Figure 4.18c presents the control torques that take $\boldsymbol{q}_e \rightarrow 0$.



(a) Virtual control signals $\vec{u}_{GV}$ of the ground vehicle.



(b) Virtual control signals U_x, U_y and U_z of the quadcotper.



(c) Control torques $\vec{\tau} = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ of the quadcotper.

Figure 4.18 – Control inputs of the flying and ground vehicles.

Assuming the AGV mimics the driving style of a human driver, who tracks in practice trajectories modeled by vector fields to obtain a smooth drive ride, the quadcotper tracks the CoM and heading angle of the AGV, thus fulfilling the mission of camera surveillance. Clearly, we are assuming the CoM of mobile robot remains in the FoV of the airborne camera, then desired contour, resultant velocity field and quadcotper tuning require to comply with it. Therefore, the smoothness of the fields facilitates to match with this assumption. Furthermore, despite that theoretically it is proved that the quadrotor tracks asymptotically the vector field of the AGV in the image plane, care must be exercised when solving the underactuation algorithm since this scheme may introduce large acceleration.

4.3 Semi-autonomous navigation using an immersive virtual reality environment

Piloting a drone can sometimes be a difficult task because its unstable and fast dynamics allow it to get out of balance quickly with any external disturbance. Unlike a car that moves in a plane, aerial robots move in a 3-dimensional space. Besides, controlling a drone is not an intuitive task for novice pilots when they lose their spatial orientation. The clearest example is when the heading of the drone is directed towards the pilot while the joystick command is in the opposite direction. This task of flying the quadrotor in direct view demands several cognitive overload in general for beginners, and in most cases the quadrotor ends up crashing during early flight experiments. Even if technological advances have improved them significantly by implementing inner controls for flying easily these types of vehicles, when the vehicle leaves the operator's line-of-sight, this mental overload increases until the pilot loses the control of the robot, see Fig 4.19. The goal of this work presented in this section is to

Figure 4.19 – Left: stress when piloting a drone. Right: frustration after crashing the vehicle [1].

reduce the cognitive overload when a beginner pilot drives a quadcopter vehicle by means of a virtual control scheme. The proposed architecture is located in two arenas; one for the robot's testing room, where the real robots are evolving and the second one named CAVE - Cave Automated Virtual Environment, where the virtual environment is placed. In this area, a World In Miniature (WIM) is modeled representing the real environment where the robot is acting. A new interaction metaphor DrEAM (Drone Exocentric Advanced Metaphor) is conceived for the virtual control scheme which represents in the virtual environment (VE) the real drone. Therefore, the goal of the architecture is to reduce the stress and help the user when piloting remotely a real robot from a virtual environment, in our case an aerial vehicle. This aim is conceived by an adaptive visual/sound feedback in the virtual environment, that can be used for improving the task.

4.3.1 Virtual control scheme

The proposed virtual control scheme can be seen as a structure composed by two blocks as depicted in Figure 4.20; the first one corresponds to the virtual environment that emulates the real scenario. The real world where the aerial robots are evolving and the CAVE platform communicate together via User Datagram Protocol (UDP) with their respective routers connected in the same network.

Both platforms have their own VRPN (Virtual-Reality Peripheral Network) server for recovering the attitude and position of the real and virtual robot respectively. Each block is represented by their corresponding frames that in most cases are different. The following subsections explain these blocks and its components.

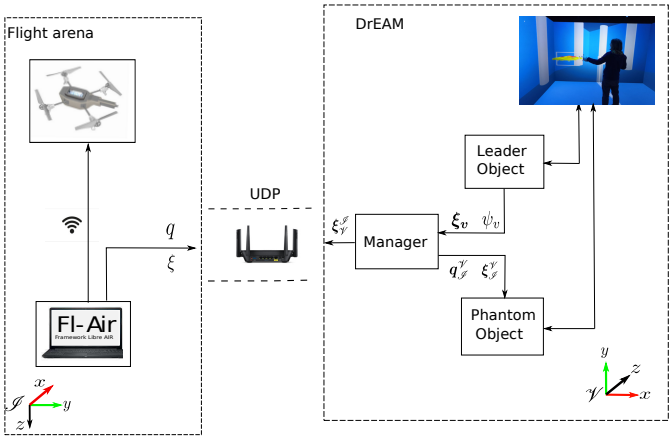


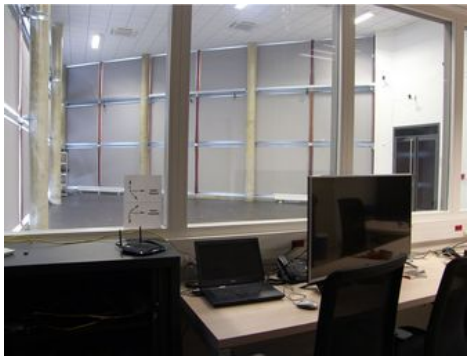
Figure 4.20 – Virtual control scheme

Real world environment

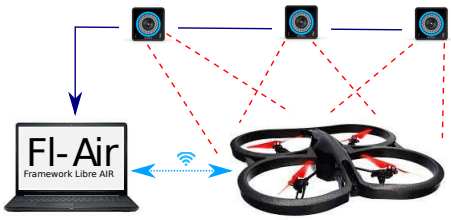
The real world environment (RWE) is where the robot is evolving. It can be seen as the scenario of the mission, for example; building inspection or rescue missions. This scenario will be represented as a WIM where the virtual drone will be piloted. On this document, the robot used for experimental purposes is the quadcopter vehicle and the RWE is the flight arena.

Flight arena
The flight arena is composed by an OptiTrack motion capture system (24 cameras, $1mm$ of precision) used to estimate the vehicle's position at 100Hz, and a monitoring room, separated from the test area for security reasons, where the ground station is placed. The size of the flight arena is determined by $10m \times 12m \times 6m$, see Figure 4.21a.

Quadcopter vehicle
The quadcopter prototype used in the flight tests, is a Parrot AR Drone 2. Its firmware was modified to work under our software Fl-AIR which is open source and runs a Linux-based operating system, capable of implementing a wide range of control schemes, see [160]. The prototype has an internal Inertial Measurement Unit (IMU) for measuring its orientation and angular rates. All the control algorithms are computed into the embedded system of the aerial vehicle, and each sampling period, the drone communicates with a ground station containing Fl-AIR framework, where their states are graphically shown for analysis purposes. This ground station communicates with the OptiTrack's software for collecting the position of the drone in order to send it via wifi as well among other values (desired references, gains, setup configuration, etc) as depicted in Figure 4.21b.



(a) Flight arena and monitoring room.



(b) Fl-AIR - Framework libre AIR.

Figure 4.21 – Real world environments.

Quadrotor dynamic model and quaternion control scheme

In this section, the dynamic model of the quadcopter corresponds with that one previously seen in Chapter 2 section 2.3 based on the quaternion formalism. Therefore, without lost of generality, the mathematical representation of the vehicle can be described as a virtual complete actuated system by imposing a desired orientation with a unit quaternion, $\mathbf{q}_d$, that will be related with the desired main thrust imposed by a controller, $\vec{U}$. Therefore, if $\mathbf{q} \to \mathbf{q}_d$ implies that $\vec{F}_\xi \to \vec{U}$.

Let the desired quaternion be with the form

$$\boldsymbol{q}_d = \boldsymbol{q_t} \otimes \boldsymbol{q_z}, \tag{4.33}$$

with

$$\boldsymbol{q_t} = e^{\frac{\ln\left(\vec{U} \otimes F^*_{th}\right)}{2}}; \quad \boldsymbol{q_z} = e^{\frac{\psi_d}{2}}; \quad \forall \psi_d \in \mathbb{R} \tag{4.34}$$

where ψ_d denotes the desired heading. Notice that (4.33) is the desired attitude imposed by means of the virtual environment.

Let us define the quaternion error as $\mathbf{q_e} = \mathbf{q}^*_d \otimes \mathbf{q}$, therefore, differentiating it with respect to time, it follows that the dynamic error between the real arena and the virtual environment can be defined as

$$\begin{aligned} \dot{\boldsymbol{q}}_e = & \ \frac{d}{dt}\left(\boldsymbol{q}^*_d \otimes \boldsymbol{q}\right) \\ \frac{1}{2}\boldsymbol{q}_e \otimes \vec{\Omega}_e = & \ \frac{1}{2}\boldsymbol{q}^*_d \otimes \vec{\Omega}^{\mathscr{I}} \otimes \boldsymbol{q} - \frac{1}{2}\boldsymbol{q}^*_d \otimes \vec{\Omega}^{\mathscr{I}}_d \otimes \boldsymbol{q} \\ \vec{\Omega}_e = & \ \boldsymbol{q}^* \otimes \left(\vec{\Omega}^{\mathscr{I}} - \vec{\Omega}^{\mathscr{I}}_d\right) \otimes \boldsymbol{q} = \vec{\Omega} - \vec{\Omega}_d \end{aligned} \tag{4.35}$$

where $\vec{\Omega}_e$ represents the angular rate error in the body frame, $\vec{\Omega}^{\mathscr{I}}$ and $\vec{\Omega}^{\mathscr{I}}_d$ are the angular velocity and the desired angular velocity in the inertial frame and $\vec{\Omega}_d$ denotes the desired angular velocity in the body frame. For validating the virtual control scheme in real-time experiments, the attitude control algorithm proposed in (3.97) was implemented in order to achieve the control objective $\mathbf{q} \to \mathbf{q}_d$.

Note from (4.34) that the only desired value imposed in the controller is ψ_d, this value will be related with the yaw angle from the virtual drone and must be mapped from the virtual environment before being used in the controller by the real robot. Observe that when introducing (3.97) into the rotational dynamics in (2.51) implies that these dynamics are stabilized. This means that $\mathbf{q}_e \to 0$ and then $\vec{F}_{th} \to \vec{U}$.

Therefore, let us propose $\vec{U} \in \mathbb{R}^3$ in the following form

$$\vec{U} = m\bar{g} - K_{p_t}(\vec{\xi} - \mathbf{q}_{r\leftarrow v} \otimes \vec{\xi}_v \otimes \mathbf{q}^*_{r\leftarrow v}) - K_{v_t}(\dot{\vec{\xi}} - \mathbf{q}_{r\leftarrow v} \otimes \dot{\vec{\xi}}_v \otimes \mathbf{q}^*_{r\leftarrow v}), \tag{4.36}$$

where $K_{p_t} = \text{diag}\{[k_{p_x}, k_{p_y}, k_{p_z}]\} > 0$ and $K_{v_t} = \text{diag}\{[k_{v_x}, k_{v_y}, k_{v_z}]\} > 0$ are control gains. $\vec{\xi}_v$ describes the position of the virtual drone expressed in the virtual frame, the term $\mathbf{q}_{r\leftarrow v} \otimes \vec{\xi}_v \otimes \mathbf{q}^*_{r\leftarrow v}$ represents the mapping of the data from the virtual environment to the inertial frame where the real drone is evolving. Notice from (3.97) and (4.36) that when $\mathbf{q}_e \to 0$, implies $\mathbf{q} \to \mathbf{q}_d$, and this means that $u_1 \to \vec{U}$ and therefore, $\vec{\xi} \to \vec{\xi}_v$.

DrEAM in CAVE

CAVE - Cave Automated Virtual Environment

A CAVE-like platform was used for validating DrEAM. This virtual room has a dimension of $7 \times 3.4\ m^2$ and is composed of four 3D projectors *Christie Mirage* 1920×1200, a workstation *HP* Z840 with two graphic cards *Nvidia* M5000, RF Active 3D Glasses, an OptiTrack motion capture system with 10 cameras, and a *PS Move* motion joystick. A Unity-plugin called *TransOne*, encapsulates data from VRPN into Unity Objects to simplify the data protocol between the motion capture system and Unity, and with the framework called Translife which creates the virtual environment.

DrEAM-Drone Exocentric Advanced Metaphor

The DrEAM is a new interaction metaphor conceived from the characteristics of the real robot and the scenario to create virtual robots and a WIM in the virtual environment where they can evolve. The goals of using DrEAM are, on one hand, to train inexperienced users providing the control of the movements of the robot that he will do during the real mission. On the other hand, to control remotely the real drone in the real scenario from a virtual environment, reducing the cognitive overload in neophytes or novice pilots. With DrEAM, user is a spectator of the virtual environment and can see a 3D reproduction of the world, for moving, resizing and rotating the environment using basic commands.

DrEAM naturally offers multiple points of view and multiple scales where the user can operate, all without requiring explicit modes or commands.User observation indicates that the operator quickly adapts to the Worlds in Miniature, and that physical props are helpful when manipulating the WIM and other objects in the environment, see Figure 4.22.

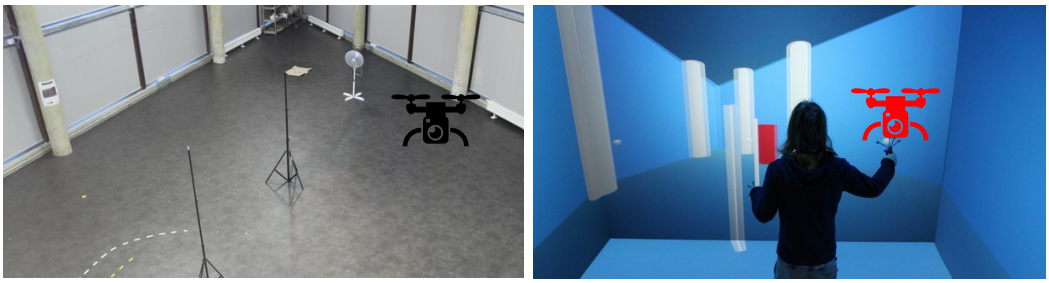


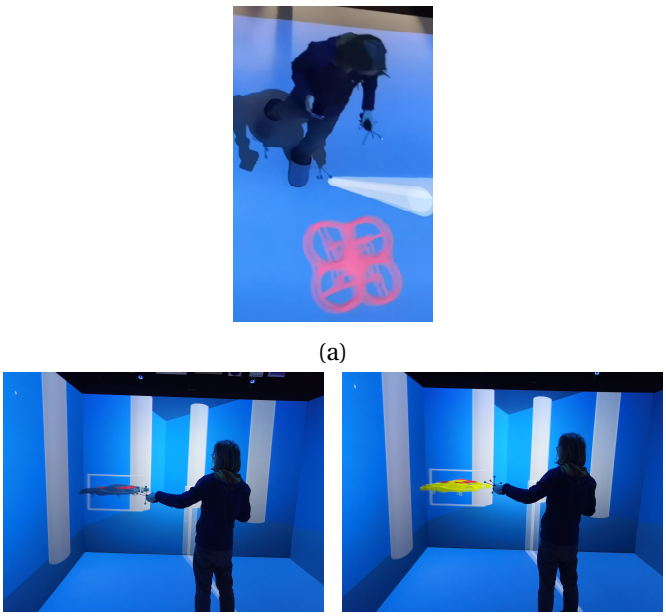
Figure 4.22 – DREAM's virtual environment and flight arena

In our case, the flight area was modeled statically using blender homemade model and unity basic shapes. Virtual robots are simple rigid bodies not affected by the law of physics that can be selected by the user using the *spherical ray-casting* function provided by Translife. This infers that their position will be the *PS-Move*'s position in the 3D-scene. The virtual frame of the 3D scene has the same origin as the inertial frame in the flight area, nevertheless, they were conceived with different rotations. Once the real scenario is represented in WIM with a direct relationship between life-size objects, a virtual drone (VD) is placed at the same position where the real drone is. This VD is an object of the virtual world that the user can take or leave it by pressing/releasing a specific button on the *PS-Move*. While the user takes the VD, he can rotate and translate it with simple natural gestures (moving his hand), therefore the user has the impression that the drone is really in his hand.

In addition, user can resize and reorient the environment without changing the scene (object positions) in any way since the program only change user position (in case of translation and rotation) and the field of view (in case of resizing). As explained before, in DrEAM, the VD is manipulated by the user by only pressing the "Take" button on the index of the wand. The first step for manipulating the drone is that, the PS Move must to be placed in the hitbox of the VD (which has the same size as the real robot). With this measure, a precise control of the VD is guaranteed. In the second step, the user can describe a trajectory with the wand, while the VD is already in his hand. Besides, the color of the VD changes to help the user known its state; a).- red when it is not possible to be taken, b).- yellow when it is ready to be taken and c).- black when has been taken already, as is depicted in Figure 4.23. In DrEAM, two different feedbacks to help the user are conceived. The first one is a visual feedback of the position and orientation of the real drone. For this, a 'phantom' drone is designed and placed with the information of the real drone coming from the flight arena. The second one is a speed indicator of the real drone emulated by a sound feedback in the CAVE.

Virtual representation

When the virtual drone (VD) is manipulated via the *PS Move*, it describes a trajectory that is a function of its position and velocity, i.e. $\vec{x}_v(t) = f(\vec{\xi}_v(t), \dot{\vec{\xi}}_v(t), \mathbf{q}_v(t), \vec{\Omega}_v(t))$ where the subindex v refers the virtual drone. All the data are in the virtual frame $\mathscr{V}$. These data information $\vec{x}_v$ is periodically sent to the RWE, as desired references for the autonomous navigation of the real drone, using the Windows asynchronous socket API in DrEAM's platform and the Linux socket API employing a string-based protocol in the ground station.



(a)

(b) (c)

Figure 4.23 – Different states of the virtual drone; (a) - cannot be taken , (b) - can be taken and (c) - has been taken.

The VD is referred in the frame $\mathscr{V}$ and the real drone in several cases in the inertial frame (or body frame) in the real world. Both frames are different and a mapping between them (Rotation matrix) is necessary in order to use the data of each one in their respective frame. For example, for a given point $(\vec{\xi}_v, \mathbf{q}_v)$ of the virtual drone in the virtual frame, with $\vec{\xi}_v = \left[x_v, y_v, z_v\right]^T$ and $\mathbf{q}_v = \left[q_{0_v}, q_{1_v}, q_{2_v}, q_{3_v}\right]^T$, its representation in the inertial frame (real scenario) is for the translation $\left[z, x, y\right]^T$ and for the orientation $\left[\cos(q_{2_v}/2), 0, 0, \cos(q_{2_v}/2)\right]^T$.

Observe that the pitch and roll information is not yet used in the virtual drone. Therefore, for a good correlation between frames the following mapping is defined

$$\mathbf{q}_{r\leftarrow v} = \frac{1}{2} + \frac{1}{2}\mathbf{i} + \frac{1}{2}\mathbf{j} + \frac{1}{2}\mathbf{k}. \tag{4.37}$$

Hence, the position of the virtual drone can be correctly represented in the inertial frame as a desired value by

$$\vec{\xi}_d = \mathbf{q}_{r\leftarrow v} \otimes \vec{\xi}_v \otimes \mathbf{q}^*_{r\leftarrow v}. \tag{4.38}$$

Similarly, for the visual feedback in the virtual environment, the attitude and position of the real drone are expressed as

$$\begin{aligned} \vec{\xi}_p &= \mathbf{q}_{v\leftarrow r} \otimes \vec{\xi} \otimes \mathbf{q}^*_{v\leftarrow r} \\ \mathbf{q}_p &= \mathbf{q}_{v\leftarrow r} \otimes \mathbf{q} \otimes \mathbf{q}^*_{v\leftarrow r}. \end{aligned} \tag{4.39}$$

where ξ_p and $\mathbf{q}_p$ define the position and attitude of the phantom drone in the virtual environment.

The phantom drone position and orientation are updated in DrEAM every frame ($100Hz$ rate) using the last information received from the real drone and mapping it, to the virtual frame using the above equations.

The virtual environment can be fitted to any space, this signifies that, it can represent a complex or simple structure, such as a building, a cube or even a sphere. Using this property, it is then possible to impose bounds in the area where the real drone is evolving and at the same time, keep safe the prototype. For our room test, the virtual drone will be bounded inside of a virtual cubic scene with the following properties

$$\begin{aligned} x_{b_1}(t) \leq x_v(t) \leq x_{b_2}(t) \\ y_{b_1}(t) \leq y_v(t) \leq y_{b_2}(t) \\ z_{b_1}(t) \leq z_v(t) \leq z_{b_2}(t) \end{aligned} \tag{4.40}$$

where $\xi_v = [x_v, y_v, z_v]^T$ denotes the position of the virtual drone and $x_{b_i}, y_{b_i}, z_{b_i}$ with $i : 1,2$ are the bounds that can be functions of time or constants delimiting the testing room.

4.3.2 DrEAM's experimental fatigue tests results

In order to test the advantages of DrEAM when controlling a real drone over a control in direct view, an experimental study was lead with eight volunteers evaluating their performance among six criteria: Mental demand, Physical Demand, Temporal Demand (this depicts the stress involved by the control task), performance (this pictures the sensation of success), Effort, Frustration (this shows the sensation of UAV's obedience). The test consists to fly manually the vehicle with a conventional joystick and with the DrEAM architecture, when performing a specific task. The participants had four minutes to learn how to use the platforms and the task must be finished in three minutes.

The scenario of the task is settled as follows: the volunteers take the control of the drone when it is hovering around the start point (S) with a fixed altitude z_t. The goal is to move the vehicle from the point (S) to the point (C) and then to the point (A) keeping, all the time, the heading of the vehicle pointing to a desired target as illustrate in Figure 4.24.

During the test each participant must follow the following rules

- The altitude of the drone should be kept constant, i.e. $z(t) \approx z_t = 1$m.
- The real drone must not be put in danger in any case.
- The task must be accomplished with accuracy as much as possible .
- The task must be achieved as faster as possible.
- The roll and pitch angles of both experiments were previously stabilized with an inner controller.

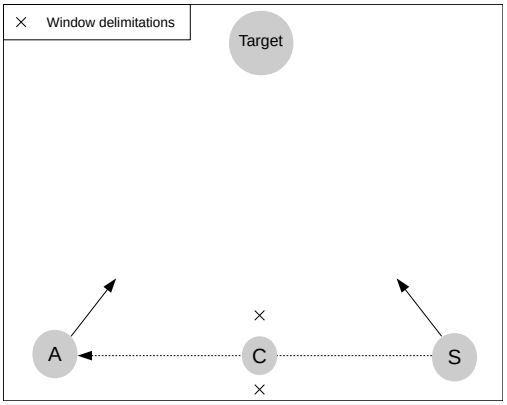


Figure 4.24 – Navigation task: start point (S), checkpoint (C) and the final point (A) keeping the heading of the vehicle pointing to the target.

After each flight test, a NASA-TLX form was filled by each participant. Figure 4.25 depicts the results of this questionnaire. Most of the volunteers were men and without any experience flying a quadrotor vehicle. Note from Figure 4.25 for every index when controlling the aerial robot with a conventional joystick, for beginner pilots can result complicated to have success even for a simple task as proposed in the test.

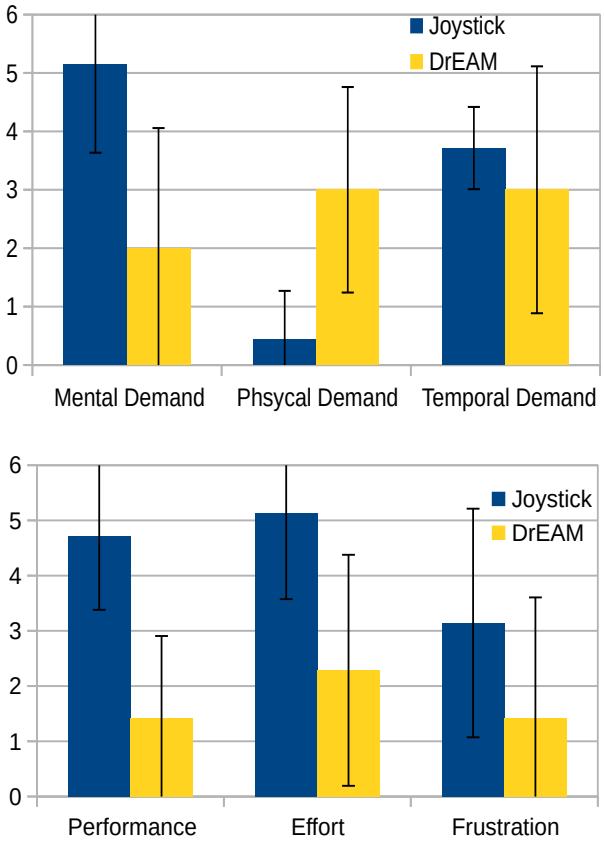


Figure 4.25 – Graphics results from the NASA-TLX questionnaire. From graphs 0 means low and 6 high demand or impact on the presented criteria.

Flight data information as timestamp, attitude, position and velocities are used to compute the mean lateral error (MLE) with respect to the desired path, the mean completion time (MCT) of all laps (a lap is every time user finish the task from the initial point to the final one) and the mean yaw error (MYE) with respect to the target. In Table 4.3 the results of these indexes are presented. During the tests, 162 laps were successfully performed by the users: 81 with a joystick (direct view) and 81 using DrEAM. Latency between CAVE and the flight arena was around $0.06s$ and no jitter has been registered, which could have disturbed the flight tests. DrEAM was sending data at $100Hz$ as well the aerial drone.

Table 4.3 – Data results of MLE, MCT and MYE indexes

Index	Joystick	DrEAM
MLE	0.389 m	0.104 m
MCT	6.810 s	5.130 s
MYE	0.252 rad	0.140 rad

To corroborate the outcomes obtained with the NASA-TLX questionnaire, one sample t-Test was carried out considering a maximum percentage of confidence (5%, $\alpha = 0.05$) in the six performance criteria. Results of this test are shown in Table 4.4, where H_0 means DrEAM has no impact on the performance criteria. Notice from this table that the $p-$value on almost all the performance criteria, excepts for the 'temporal demand' and 'frustration', is less than 5% ($p < 0.05$). Therefore, it is possible to state that DrEAM has not impact on these criteria. Nevertheless, for the case of the 'temporal demand' and 'frustration', with a $p-$value bigger than 0.05, it is not possible, at the moment, to state that there is not demand of this criteria.

Table 4.4 – Results from the one sample t-Test.

Hypothesis	p-value	H_0
Mental Demand	0.013	Rejected
Physical Demand	0.007	Rejected
Temporal Demand	0.269	Not Rejected
Performance	0.003	Rejected
Effort	0.022	Rejected
Frustration	0.140	Not Rejected

The fatigue conclusion in these tests was that DrEAM increases the control ergonomy for inexperienced users when controlling the aerial robot, without loss of precision, in comparison with controlling the robot in direct view, in particular concerning complex movements with more than two DoF.

4.3.3 DrEAM real-time validation with robots

DrEAM performance was validated in real-time experiments when controlling remotely a real quadrotor vehicle. The practical goal of these experiments is to corroborate and compare the performance of the real robot versus the virtual robot in two scenarios. In the first scenario, the inspection in reduced spaces is emulated.

For this, the goal is to navigate with the real robot inside a building crossing small surfaces as windows. In the second scenario a cellar inspection is emulated, therefore, the objective is to send the aerial vehicle inside of the cellar and navigate there to survey the zone. The cellar is composed by pillars that the aerial vehicle needs to avoid. A video with the experimental results can be seen at: https://youtu.be/jMYWIoCsj7I.

The tests are carried-out with the following procedure

1. The take off and landing of the real robot are not still considered controlled with DrEAM. They are done in a safety mode by a user in the flight arena.
2. The initial position of the real drone for the tests is $\vec{\xi}(t_0) = [0,0,0.5]^T$m. At this position, DrEAM can take the control of the vehicle.
3. At any moment, the user in the virtual world can always put the vehicle at hover position, to change/rotate/expand the view of the WIM.
4. For simplifying the test, when the user takes the virtual drone and moves it, the real drone will be aligned to the direction of the trajectory. Nevertheless, the user can rotate it, in any desired yaw angle.
5. No sensors for detecting obstacles are embedded into the drone. Therefore, large errors in the virtual commands should produce the crash of the robot with the obstacles.

Scenario 1: crossing reduced spaces for rescuing

As previously explained, in this scenario, the vehicle must cross through some reduced spaces that can produces several stress when the user drives the aerial drone in direct view. For this scenario, two physical windows are located in the flight arena in different altitudes, positions and headings. These windows are also created in the WIM with the same characteristics. The first one is located at $\vec{\xi}_{w_1} = [2,2,1.4]^T$m and the second one at $\vec{\xi}_{w_2} = [-0.5,-1.7,1.7]^T$m in the flight area. The test consists that the user must move the virtual vehicle in any desired path while crossing the windows and repeating the experiment three times with different translational velocities. The real robot must imitate the same performance of the virtual drone in real time as can be seen in Figure 4.26.

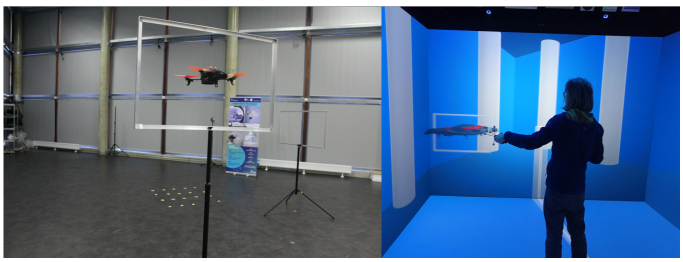


Figure 4.26 – Scenario of the first experiment.

From Figures 4.27 - 4.32, the states performance of the robots (real and virtual) when this scenario is validated are depicted. In Figure 4.27, the translation motion in 3D is illustrated. Moreover, observe from Figure 4.28 that the user changes the heading of the virtual robot in order to cross the windows. This information is also presented in each axes in Figures 4.29 – 4.31. In this experiment, from $0 < t_1 \leq 5s$ the real vehicle takes off autonomously to be placed at 1.4m in z-axis. Once the vehicle switch to DrEAM's mode, it is posed in an altitude of 0.5m. At time $t = 10$s the user in DrEAM takes the virtual drone and manipulates it making a trajectory and repeating it three times at different velocities. The three laps are done at times: $10 < t_1 \leq 40s$, $40 < t_2 \leq 60s$ and $60 < t_3 \leq 80s$, respectively.

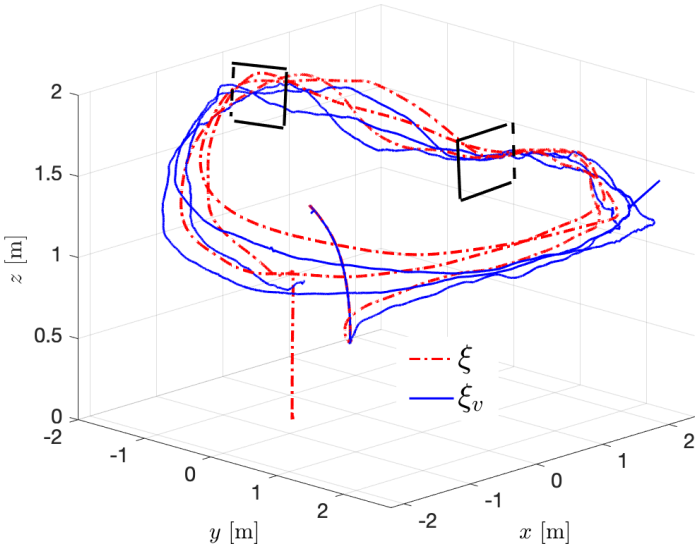


Figure 4.27 – 3D performance of the virtual drone and the real vehicle when scenario 1 is validated in real time.

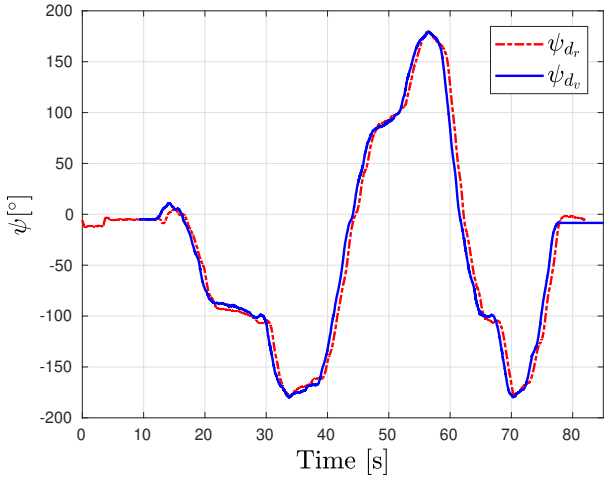


Figure 4.28 – Yaw angle behavior of the drones (real and virtual).

The first one was performed with a slower velocity because the user did not have much confidence and he was tense. The second lap, the user increases the velocity of the trajectory (up to 1m/s) and in the last one, the user tries to do the displacement as fast as possible. The variations in the velocity can be checked in Figure 4.32. In figures, the subscripts d_r and d_v represent the state of the real drone and the virtual drone respectively.

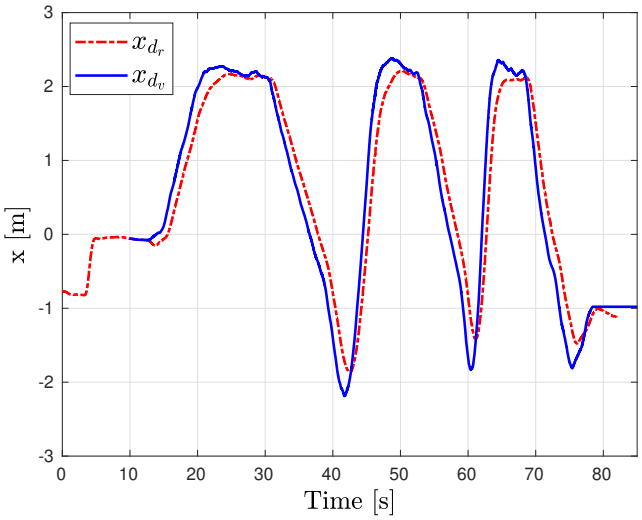


Figure 4.29 – x-position performance of the virtual drone and the real vehicle obtained from the first scenario. Observer the good behavior of the real drone when imitates the virtual drone trajectory.

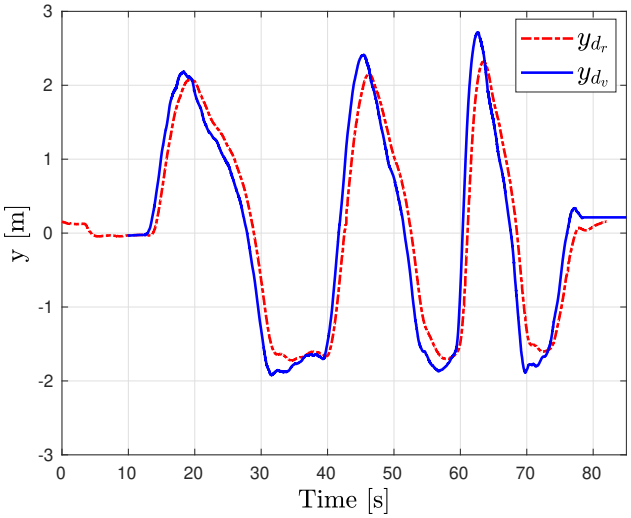


Figure 4.30 – Performance of the robots (virtual and drone) in the y axis.

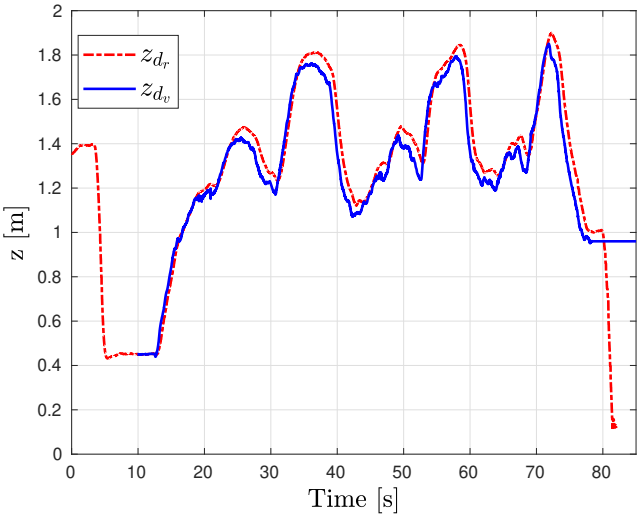


Figure 4.31 – z-state behavior of the first scenario. Observe that the robots (virtual and real) change their altitude for crossing the windows.

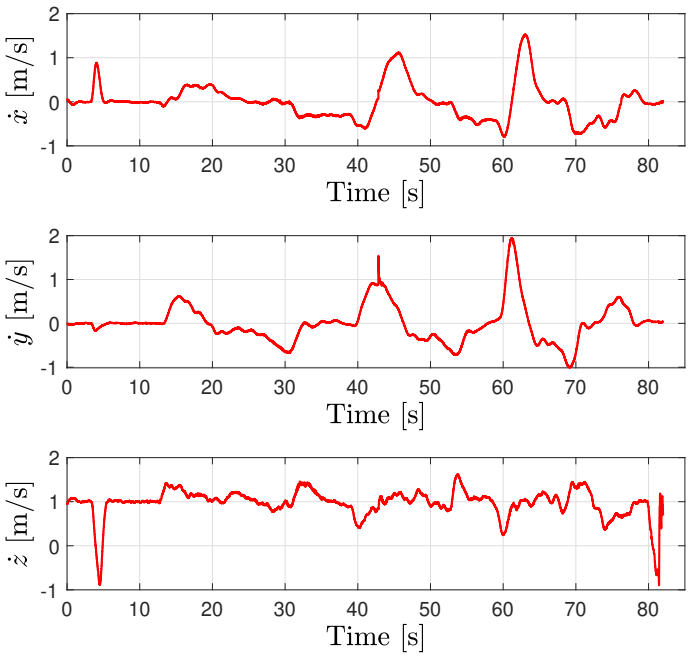


Figure 4.32 – Performance of the velocities of vehicle during the first scenario. Note that it is increased mainly in the x and y axes.

Scenario 2: Building inspection

In this scenario, two tripod are required in the flight arena emulating pillars in the cellar. In DrEAM, two cylinders were drawn to simulate the corresponding 'pillars' as can be seen in Figure 4.33. The practical goal is to illustrate the maneuverability of DrEAM when the user manipulates the virtual drone between the emulated pillars and changing its heading. The experiment is repeated several times with different maneuvers trying to approach the drone as near as possible to the 'pillars'.

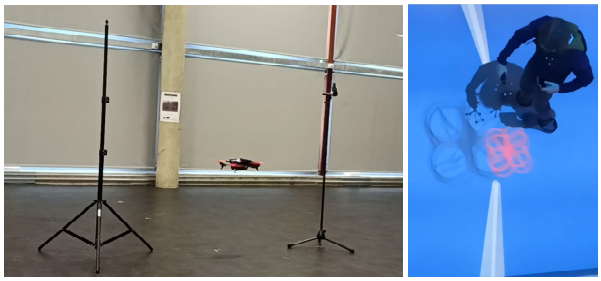


Figure 4.33 – Scenario of the second experiment.

The performance of the states when the real drone follows the position references of the virtual vehicle are shown in Figures 4.34 - 4.38. A 3D representation of the trajectories is presented in Figure 4.34. As in the first scenario, the initial step was to take off the real vehicle autonomously in safety mode. In this mode the real drone is placed at an altitude of 1.4m and a few seconds later its altitude is reduced at 0.5m in hover position, ready to follows the virtual commands. This is done in the interval time $0 < t_0 \leq 20$s.

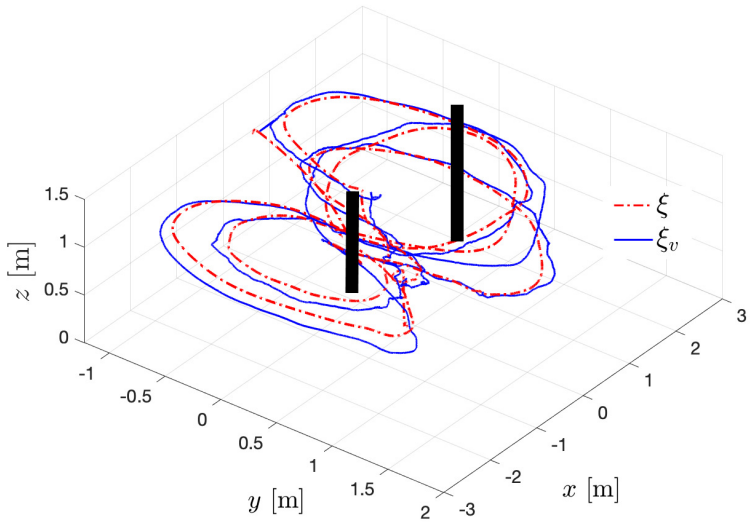


Figure 4.34 – 3D behavior of the virtual and the real aerial vehicles when performing the second scenario.

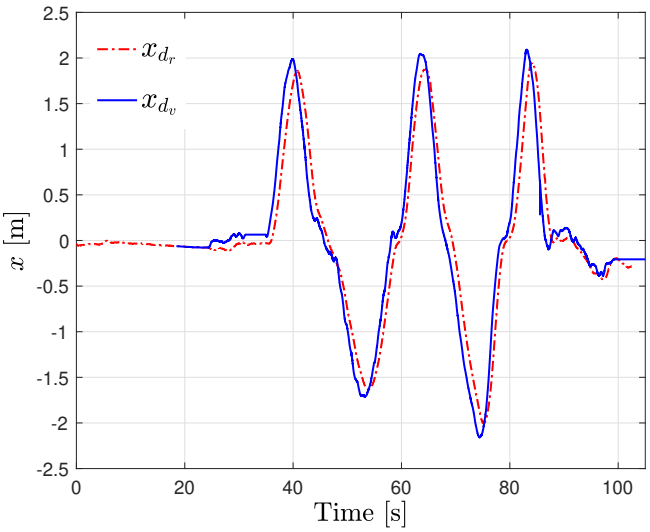


Figure 4.35 – Performance of the aerial robots in the x axis. Observe the good performance when the real robot imitates the virtual robot.

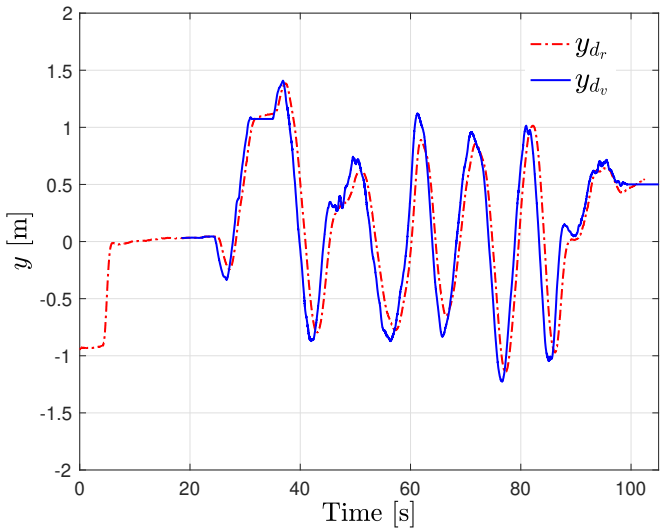


Figure 4.36 – y- state behavior of the aerial robots (virtual and real) when the user turns the pillars three times.

As depicted in Figure 4.34 the goal of the experiment was that the user manipulates the virtual drone between the pillars. Three laps were carried out by the user at different times; $20 < t_1 \leq 55$s for the first one, $55 < t_2 \leq 80$s for the second lap and $80 < t_3 \leq 100$s for the last one. This task was made in an area of $-2 \leq x \leq 2$, $-1.5 \leq y \leq 1.5$ and $0.5 \leq z \leq 1.4$ all in meters. Moreover, the mission demands to change the heading of the aerial robots several times as is illustrated in Figure 4.38. From these figures, observe the good performance of the real robot when it is controlled with the DrEAM architecture. Remark that the task has been developed with good precision, that it should not be possible when the vehicle is controlled in direct view.

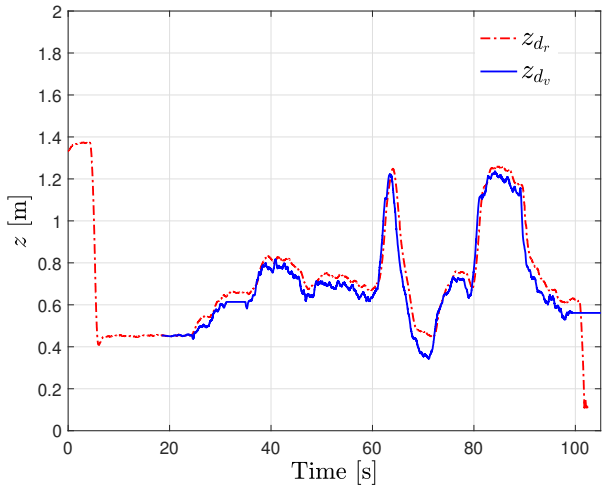


Figure 4.37 – z performance of the vehicles obtained during the second scenario. Observe that the last two laps the user change also the altitude during the trajectory.

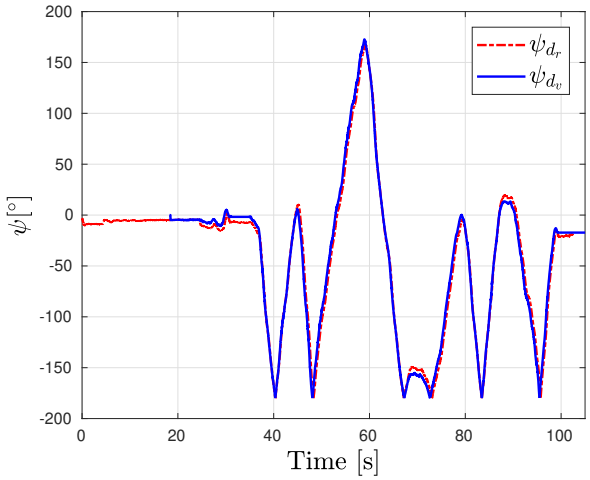


Figure 4.38 – Behavior of the heading of the virtual and real drone respectively.

4.4 Quadcopter autonomous navigation conclusions

The aim of this chapter was to propose and introduce autonomous and semi-autonomous navigation control techniques. They exploit properties from classical and quaternion based algorithms providing advantageous features for navigating aerial vehicles. These navigation techniques have been implemented for tracking optimal trajectories, ground autonomous vehicles and virtual trajectories imposed by means of an immersive virtual reality environment.

The first navigation technique concerns the trajectory generation and tracking problems. It is based on a simple nonlinear controller, the MPC algorithm for path generation and a disturbance observer for giving robustness to the system.

This technique was validated in a real-time flight test for a trajectory generation in a $3D-$space while dealing with external perturbations. On one hand, the optimal problem was solved considering $H_p = 10$ without constraints for reducing computational issues. Nevertheless, augmenting H_p for giving robustness to the prediction of the trajectory will exceed the capacity of memory of the embedded system. This is mainly due to the fact the H_p is related with the prediction model. On the other hand, the experimental results demonstrate that the architecture reduces considerably the error in the path tracking problem even when external perturbations were presented.

Furthermore, a novel scheme for the aerial surveillance of AGVs using quadcopters was proposed. The scheme uses a simple vision algorithm for the target detection and, a vector field approach for describing the movements of a driver. Experimental results illustrate the feasibility and good performance of the proposed scheme. However, from this experimental validation, we can sate the following: the maximum number of features for the vision algorithm correspond to $N_p = 64$ and the movements described by the ground vehicle need to be smooth. In other words, the ground robot must to remain in the FoVs of the airborne camera. Higher number of features can cause mismatching when looking for the centroid. Moreover, fast accelerations of the ground robot produce that it goes out of the FoV.

Finally, a new semi-autonomous virtual control architecture for controlling robots remotely was developed in this chapter. The architecture is composed by a new metaphor interaction called DrEAM that uses a world-in-miniature located in a virtual environment and, a real aerial robot located in the flight arena. Two experimental tests were addressed to highlight the easy implementation and feasibility of the proposed architecture. The testing rooms were connected using the UDP protocol with a insignificant delay in the exchange of packages such that, it was neglected. In addition, it was demonstrated in real-time experiments the easy maneuverability and controllability of the real drone when the user is controlling it by means of DrEAM. Therefore, a reduction of the cognitive overload was observed when piloting with DrEAM than when controlling the real drone in direct view.

Chapter 5 Part V

5 Robust control scheme based on disturbance estimator

Aerial Vehicles have gained an enormous interest for their civil potential applications. Among different UAVs, quadrotors are remarkably popular and have been used extensively in research over the past decade [174, 175]. A high-performance attitude control is a prerequisite for developing any other high-level control tasks [176]. For autonomous navigation, aerial vehicles need robust control systems to compensate the adverse effects produced by parametric and non-parametric uncertainties, unmeasured dynamics and atmospheric disturbances, such as wind and turbulence [177]. Although many solutions have been proposed in the literature, very few of them have been validated in real flight tests and the most popular techniques are still based on classical control strategies. This is mainly due to the constraints imposed by the limited computational resources of the embedded systems, which are typically microcontrollers. Also, and perhaps more importantly, because of the unstable nature of quadrotors, controllers must run typically at very high frequencies [10].

Robust control for quadrotors is still an active field of research, see [178, 179, 180], because the aerodynamic effects are extremely hard to be accurately modeled [181] and, specially in outdoor applications, a UAV is constantly perturbed by wind gusts [182]. Disturbance observers have drawn the attention of many researchers as a tool for facing these problems [183, 184].

Inspired by the works [140, 141, 185], the goal of this chapter is to present a novel robust control architecture for solving the practical stability for aerial autonomous vehicles subject to uncertainties in the model, external perturbations and rotor's failures, while meeting the rotor's physical constraints. This chapter is organized as follows: first, some preliminaries as well the problem formulation and the proposed solution are stated in section 5.1. Furthermore, a robust control architecture based on disturbance estimator is introduced in section 5.2. An experimental study on disturbance rejection is carried out to prove the performance of the proposed architecture. Then, an enhanced robust tolerant control architecture is addressed in section 5.3. Here, simulations and experimental tests with multiple rotors failures are conducted. Finally, a bounded robust control scheme is conceived and proved experimentally in section 5.4. Some conclusion of the developed strategies are discussed in section 5.5.

5.1 Preliminaries

Recall from (2.16), that the nonlinear dynamic model of the quadcopter using the Newton-Euler formalism can be expressed as

$$\begin{aligned} \dot{\vec{\xi}}(t) &= \vec{v}(t), & m\dot{\vec{v}}(t) &= R(t)\vec{F}_{th}(t) - m\vec{g}, \\ \dot{R}(t) &= R(t)\hat{\vec{\Omega}}(t), & J\dot{\vec{\Omega}} &= -\vec{\Omega}(t) \times J\vec{\Omega}(t) + \vec{\tau}(t) \end{aligned} \tag{5.1}$$

In this work it is assumed that the thrust, $f_i \approx k_f w_{M_i}^2$, and torque, $\tau_{M_i} \approx k_\tau w_{M_i}^2$, of each propeller are directly controlled by the angular rate of the motor, w_{M_i}, with k_f and k_τ are aerodynamic thrust and torque factors, respectively. Therefore, from (5.1), $\vec{F}_{th}$ and $\vec{\tau}$ can be written as $\vec{F}_{th} = [0, 0, \sum_{i=1}^{4} f_i]^T$ and $\vec{\tau} = [l(f_1 + f_4) - l(f_2 + f_3), l(f_1 + f_2) - l(f_3 + f_4), \sum_{i=1}^{4} \tau_{M_i}]^T$, with l representing the distance from the center of mass to the point where the force is applied.

Notice from the above, and according to (2.7), the main thrust can be defined as $u_1 = \sum_{i=1}^{4} f_i$, which acts along the direction of $z-$axis in the inertial frame. In addition, notice that the torque generated by each propeller can be represented by using its thrust and a scale aerodynamic factor denoted by c_{tf}. Thus, the following relation can be established

$$\vec{u}_R = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ l & -l & -l & l \\ l & l & -l & -l \\ -c_{tf} & c_{tf} & -c_{tf} & c_{tf} \end{bmatrix} \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{bmatrix} \tag{5.2}$$

Now, let's consider that system (5.1) can be stabilized using a feedback controller with the form

$$\vec{u}_R = -K_L \vec{x} \tag{5.3}$$

where $\vec{x} = [\vec{\xi},\ \vec{\eta}]^T$ denotes the system states, $K_L > 0$ is a constant matrix gain and $\dot{\vec{\eta}}$ represents the attitude vector of the vehicle, with $\vec{\Omega} = W_\eta \dot{\vec{\eta}}$ and W_η defines the standard kinematics relation between $\vec{\Omega}$ and $\vec{\eta}$, see [147]. The linear system of (5.1) for designing $\vec{u}_R$ is given by $\dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}_R(t)$, where A and B are the state matrix and control vector and $\vec{u}_R = [u_1,\ u_2,\ u_3,\ u_4]^T$.

Experimental tests using the (5.3) controller has shown a correct performance, nevertheless, when the vehicle is exposed to external disturbances (as wind gust), the system becomes unstable and some crashes occurs. These non desired effects can be also produced by actuators fault (damage in the propeller or the motor itself) producing a loss in the efficiency of the vehicle. Moreover, voltage variations in battery could be seen also as a motors Loss of Efficiency (LoE), which is reasonable due the fact, the motor thrusts are related with the battery voltage. In addition, when the aerial robot is exposed to these undesired situations, the controller, in the embedded system, computes a large amount of energy for counteracting them.

If this energy is sent and without measuring it, and during large periods of time, the actuators in the aerial vehicle can be overheated and damaged, leading a poor performance or undesirable crashes when performing a desired mission. This situation happens because they have physical constraints that are not often considered when computing the control law, e.g. maximum of revolution per minute (RPM) in a motor. Therefore, the design of robust controllers to counter-actuate aggressive external disturbances or non-desired dynamics without saturate the actuators remains a challenge for the automatic control community.

5.1.1 Proposed solution

Exploring the properties of disturbance observers, we introduce a robust control strategy for solving the disturbance rejection problem. The underlying idea behind this work is that the unknown lumped signal $d(t)$, along the solutions of (5.1) can accurately be estimated and counteracted. However, as aforementioned in the presence of LoE in rotors or their physical constraints, DOBC strategies have limitations and needs to be improved. Therefore, our robust control architecture is composed of the following stages,

1. Enhancing (5.3) by means of a disturbance estimator such that $\vec{u}_R$ becomes robust in presence of model uncertainties and a class of external disturbances.

2. Improving the previous robust control strategy such as a rotors fault estimator $\hat{M}_\zeta$ can be derived by means of $\vec{u}_R$, giving robustness to the architecture even in LoE in rotors.

3. Solving the physical constraints of the motors such that, the bounded control input $\underline{\sigma} \leq \vec{u}_R^* \leq \bar{\sigma}$ of the robust control scheme, guarantees $\vec{\eta} \to \vec{\eta}_r$ and $\vec{\xi} \to \vec{\xi}_r$ with $\vec{\eta}_r$ and $\vec{\xi}_r$ standing for the attitude and position references, respectively.

5.2 Robust control scheme for disturbance rejection

Let consider that system (5.1) can be also expressed as a perturbed nonlinear system with the form

$$\begin{aligned} \dot{\vec{x}}(t) &= (A+\Delta A)\,\vec{x}(t) + (B+\Delta B)\,\vec{u}_R(t) + f(\vec{x},t,\vec{u}_R(t)) + d(t) \\ \vec{y}(t) &= \vec{x}(t) \end{aligned} \tag{5.4}$$

where $\vec{x}(t) = [\vec{\xi},\vec{\eta}]^T \in \mathbb{R}^n$ and $\vec{u}_R(t) \in \mathbb{R}^m$ are the state and control vectors respectively, $f(\vec{x},t) : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}^n$ defines an unknown non-linear function, and $d(t) : \mathbb{R}^+ \to \mathbb{R}^n$ denotes the vector of unknown disturbances. The state and control matrices are split so that A and B are known and ΔA and ΔB are parametric uncertainties. Moreover, it is assumed that the nonlinear function $\frac{\partial[f(\vec{x})+B\vec{u}_R]}{\partial \vec{u}_R} \neq 0$, for all $(\vec{x},\vec{u}_R) \in \mathbb{R}^n \times \mathbb{R}^m$.

The following assumptions are stated

Assumption 2. *The pair* (A,B) *is controllable.*

Assumption 3. *The uncertainty* $f(\vec{x})$ *belongs to the column space of* B*, i.e. there exists a vector* $\vec{d}_f(\vec{x}) \in \mathbb{R}^m$ *such that* $f(\vec{x}) = B\vec{d}_f(x)$.

Assumption 4. *There is a region* $\mathscr{D} = \vec{x} \in \mathbb{R}^n : |\vec{x}| \leq r_x$ *where: i.)* $\vec{d}_f(\vec{x})$ *is locally bounded, ii.)* $\vec{d}_f(0) = 0$ *and iii.) its derivative is locally bounded by* $|\nabla \vec{d}_f(\vec{x})| \leq c_x$.

Assumption 5. *The input disturbance* $d(t)$ *is uniformly bounded and its derivative is bounded by* $\left\|\dot{d}\right\| \leq c_d$, $\forall\, t \geq 0$.

For developing the robust control scheme, let us propose the following ideal reference model

$$\dot{\vec{x}}_q(t) = A_q\vec{x}_q(t) + B_q\vec{u}_q(t) \tag{5.5}$$

where $A_q \in \mathbb{R}^{n\times n}$, is Hurwitz, $B_q \in \mathbb{R}^{n\times m}$, $\vec{x}_q \in \mathbb{R}^n$ and $\vec{u}_q \in \mathbb{R}^m$.

The goal for robust control strategies, is that $\vec{x} \to \vec{x}_q$, therefore, the following error can be proposed

$$\vec{e} = (\vec{\xi} - \vec{\xi}_q, \vec{\eta} - \vec{\eta}_q)^\top = \vec{x} - \vec{x}_q \tag{5.6}$$

Differentiating (5.6) and using (5.4), (5.5) and adding and subtracting $A_q\vec{x}$, it holds that

$$\dot{\vec{e}} = A_q\vec{e} - \Gamma \tag{5.7}$$

with

$$\Gamma \quad = \quad (A+\Delta A)\vec{x} + (B+\Delta B)\vec{u}_R + f(\vec{x},\vec{u}_R,t) + d(t) - A_q\vec{x} - B_q\vec{u}_q$$

Observe that if $\Gamma \to 0$ then (5.7) will be asymptotically stable. Hence, $\vec{u}_R$ can be proposed as

$$\vec{u}_R = \overbrace{B^+\left[(A_q - A)\vec{x} + B_q\vec{u}_q\right]}^{\vec{u}_n} + \underbrace{B^+\left[-\Delta A\vec{x} - \Delta B\vec{u}_R - f - d\right]}_{\vec{u}_\zeta} \tag{5.8}$$

where $B^+ = (B^TB)^{-1}B^T$ corresponds to the pseudo-inverse of B. Then, the challenge will be to compute $\vec{u}_n$ and $\vec{u}_\zeta$.

Notice from (5.8) that, on one hand $\vec{u}_n$ concerns about the reference tracking performance for system (5.5). Thus, proposing $\vec{u}_q = -K\vec{x}$ makes $(A_q - A)\vec{x} + B_q\vec{u}_q$ stable, with $K > 0$. Therefore, this implies that $\vec{u}_n = -B^+K\vec{x} = -\bar{K}\vec{x}$ with $\bar{K} > 0$. On the other hand, notice that, $\vec{u}_\zeta$ will be the control part for compensating the inner uncertainties, unknown dynamics or external perturbations. In addition, observe that $\vec{u}_\zeta$ is a function of unknown variables, i.e., $\vec{u}_\zeta = f(\Delta A\vec{x}, \Delta B, \vec{u}_R, f(\vec{x},\vec{u}_R,t), d(t))$, and therefore it is difficult to estimate this parameter in this form. However, using (5.4), it can be rewritten as

$$\vec{u}_\zeta = B^+[-\Delta A\vec{x} - \Delta B\vec{u}_R - f(\vec{x},\vec{u}_R,t) - d(t)]. \tag{5.9}$$

Observe that all the proposed methodology for estimating the uncertainties and external perturbations are based on computing $\vec{u}_\zeta$ in (5.9). Nevertheless, this parameter is function of unknown variables as stated in (5.9), i.e., $\vec{u}_\zeta = f(\Delta A\vec{x}, \Delta B, \vec{u}_R, f(\vec{x}, \vec{u}_R, t), d(t))$. However, using (5.4), (5.9) can be rewritten as

$$\vec{u}_\zeta = B^+[A\vec{x} + B\vec{u}_R - \dot{\vec{x}}] \tag{5.10}$$

Intuitively, it indicates that the unknown dynamics and disturbances can be estimated from the known dynamics of the systems and control signal. Implementing (5.10), in this form in a microcontroller, is not commonly used, hence for solving it, the signal (5.10) can be accurately represented in the frequency domain as follows [186]

$$\vec{u}_\zeta(s) = G_f(s)B^+[A\vec{x}(s) + B\vec{u}_R(s) - s\vec{x}(s)] \tag{5.11}$$

if $G_f(s)$ is a strictly proper stable low-pass filter with unity gain and zero phase shift over the spectrum of $\vec{u}_\zeta$ and gain elsewhere. Computing the Laplace transform of (5.8) and using (5.11), the disturbance rejection observer control law $\vec{u}_R(s)$, can be obtained as

$$\vec{u}_R(s) = [I - G_f B^+ B]^{-1} B^+ \left[A_q\vec{x} + B_q\vec{u}_q - A\vec{x}(1 - G_f) - sG_f\vec{x}\right] \tag{5.12}$$

Observe that (5.12) is in general only an approximation cause the pseudo-inverse, however, it is generally satisfied for some cases [187]. The asymptotic stability of the closed-loop system is established in [188], when the filter $G_f(s)$ is chosen appropriately as a strictly-proper stable filter with unity gain and zero phase shift over the spectrum of the uncertain term $f(\vec{x}, \vec{u}_R, t) + d(t)$ and zero gain elsewhere. The developed robust control architecture is depicted in Figure 5.1.

To summarize the tuning procedure, three essential parameters are considered: the control gains of the feedback controller $\vec{u}_n$, the filter time constant T and the desired reference model (5.5). On one hand, the value of the law-pass filter corresponds to the dynamic of the perturbation to be estimated. A high value of T causes to filter excessively and, a small one makes that any noise be a perturbation giving large peaks of the control action which means that the system could become unstable. On the other hand, the b value of the reference model represents a double integrator of each axes of the quadrotor and it was obtained experimentally using a Pseudo Random Binary Sequence (PRBS) and the Recursive Least Squares (RLS) algorithm, for identification purposes.

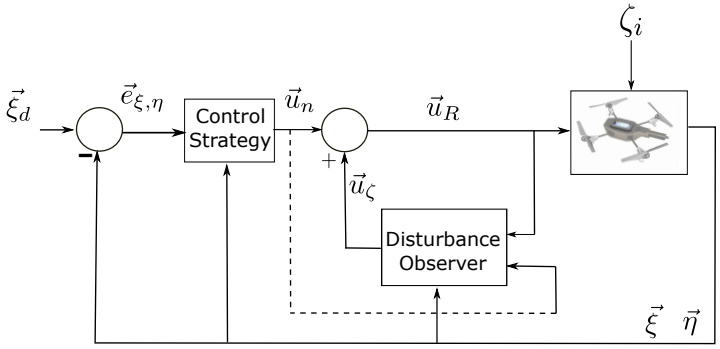


Figure 5.1 – Block diagram of the control structure. $\vec{u}_n$ and $\vec{u}_\zeta$ correspond to the the nominal control action and the control rejection part, respectively, $\vec{\xi}_d$ is the desired reference. $\vec{u}_R$ represents the proposed controller ζ represents external disturbances.

5.2.1 Experimental results for disturbance rejections

The quadrotor prototype used in the flight tests, is a Parrot AR Drone 2 running the open source software Fl-AIR [160]. An OptiTrack motion capture system was used to estimate the vehicle's position and analyzed in a ground station with others control variables coming from the IMU sensor.

Three scenarios are proposed for validating the robust control scheme (5.8) using (5.11). The practical goal is to expose the aerial vehicle to common disturbances as: wind, variable mass and degradation in the performance of the rotors. The tests are carried-out with the following procedure:

- The state feedback controller is implemented and tuned into the aerial vehicle and used in all experiments.
- The goal is to keep the vehicle at hover at the desired position is $(x_r, y_r, z_r) = (0, 0, z_r)$m.
- Disturbances are applied when the vehicle is at hover for degrading its performance.

The experimental tests are developed using parameters on Table 5.1. Here, $K(1)$, $K(2)$, express the gains of the feedback controller for each subsystem. For the design of disturbance estimator, each axis of (5.1) is considered decoupled and modeled by a double integrator of the form $G(s) = b/s^2$ such that b is is obtained experimentally and T represents the bandwidth of the low-pass filter $G_f = 1/(Ts+1)$ defined in (5.11). A video of the experimental results can be seen at: https://youtu.be/lhcRiGlPq54.

Table 5.1 – Gain parameters used in the experimental tests

Controller	ϕ/θ	ψ	x, y	z
$K(1)$	0.8	0.6	0.17	0.3
$K(2)$	0.1	0.2	0.13	0.1
b	140	44	10	5
T	0.1	0.5	0.5	0.3

Wind-gust disturbance

In this experiment, the practical goal consists of keeping the quadcotper at hover position with an altitude $z_d = 1$m while the vehicle is being affected by wind disturbance. For that, three fans (1m from the desired position, see Fig. 5.2) were put to encircle the aerial robot for emulating wind disturbance. Thus, the scenario is as follows: the quadrotor starts at hover position $x(0) = y(0) = 0$m with the observer block activated while is being perturbed by the fans. The observer block is deactivated and re-activated to compare the performances.

Figure 5.2 – Wind-gust scenario.

From Figures 5.3-5.5 observe the good quadcopter performance when applying the proposed controller. In Figure 5.3 the 3D state performance is depicted. Observe that when the robust control scheme is activated, the vehicle remains in a small neighborhood around the desired set-point, that is not the case when using only the state feedback controller.

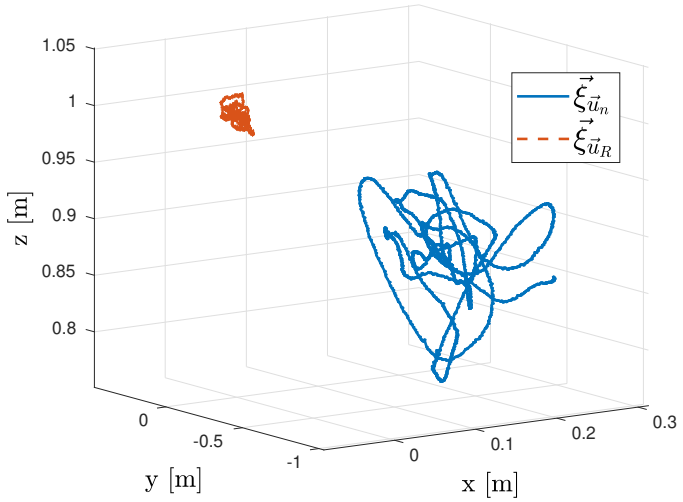


Figure 5.3 – Vehicle's performance in $3D$ against wind disturbance.

Note in Figure 5.4 that from time $t = 0$s to $t < 45$s, the proposed controller is applied and the vehicle stay close into a neighborhood near to the origin. From $t \geq 45$s and $t \leq 75$s, only the state feedback control is used and the quadrotor behavior is degraded due the control action is not enough to compensate the wind-gust disturbances. To recover the good performance of the system, the observer block is activated again at time $t > 75$s.

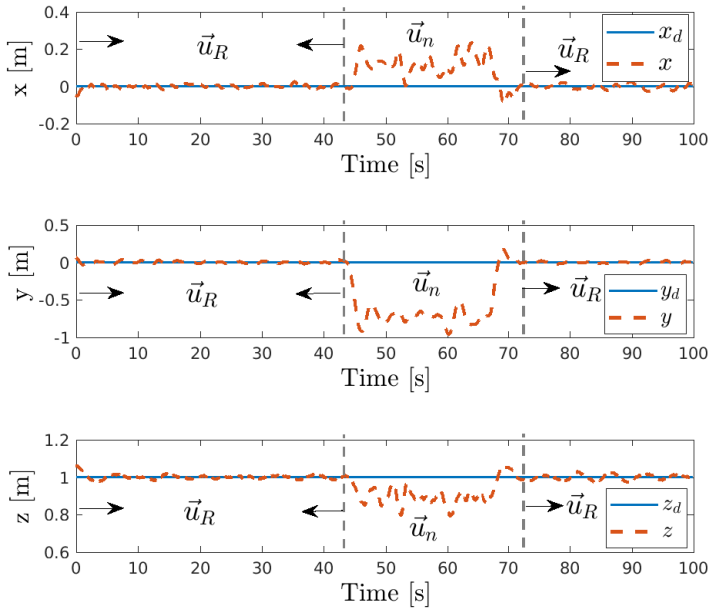


Figure 5.4 – Performance of the vehicle in $x-$axis, $y-$axis and $z-$axis.

In Figure 5.5 the disturbance estimation is depicted related to the system performance. Notice that when the observer is activated the disturbance is compensated and then the system behavior is barely affected. Nevertheless that, from $t \geq 45$s and $t \leq 75$s the observer is not activated and the real values of the disturbances appear as it can be seen in this figure. It is worth mentioning that, the observer is used in open-loop in order to continue estimating the disturbances.

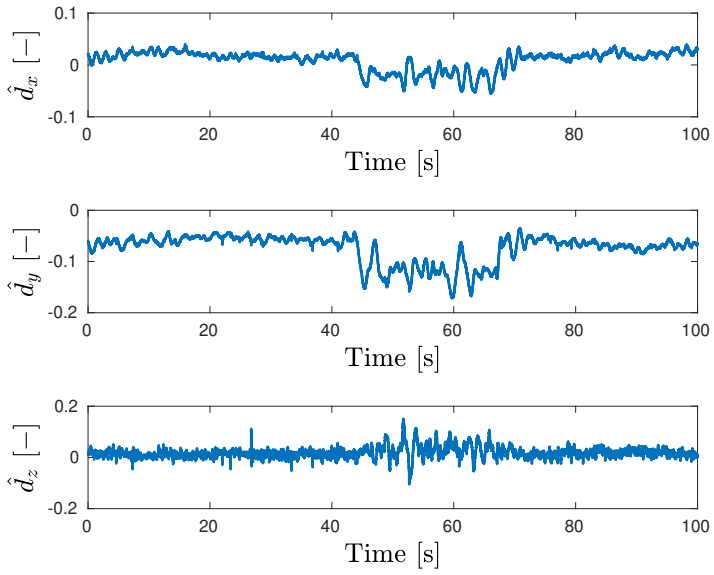


Figure 5.5 – Estimation of the wind disturbance.

Variable mass

This scenario was proposed for applications with variable mass. For example, when the drone is tuned for certain gains and takes a mass for transporting to a desired position and needs to leave it, usually, their performance is degraded. In this experiment, a mass of 100g is applied on the rear right rotor when the quadrotor is at hover, emulating a box that needs to be transported, see Figure 5.6. The desired altitude is $z_d(t) = 1.5$m.

Figure 5.6 – Lateral weight disturbance applied to an aerial system.

In Figure 5.7, the 3D performance of the vehicle is depicted. On one hand, note here that when the mass is added, the state feedback controller is not enough to compensate it and the system performance is severely degraded (blue line). On the other hand, when the aerial drone is flying with the proposed control scheme and the mass is added, its performance is barely degraded (red line).

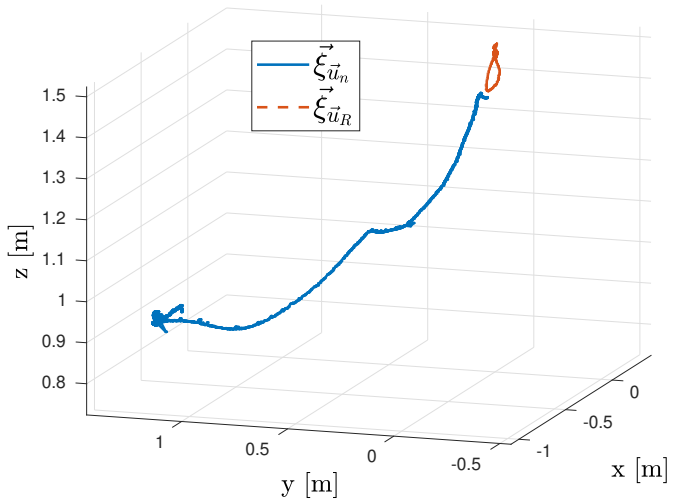


Figure 5.7 – Vehicle's performance in $3D$ against the weight disturbance.

Figure 5.8 depicts the behavior of the vehicle in the three axes when adding a mass in motor M_2. Observe in this figure the poor performance when the mass is added to the vehicle flying only with the linear controller, which is not the case, when the proposed scheme is used during the flight and the mass is added. Notice that for this last case, the disturbance in the system is unperceived, see graphs at time 2s$< t <$ 4.5s.

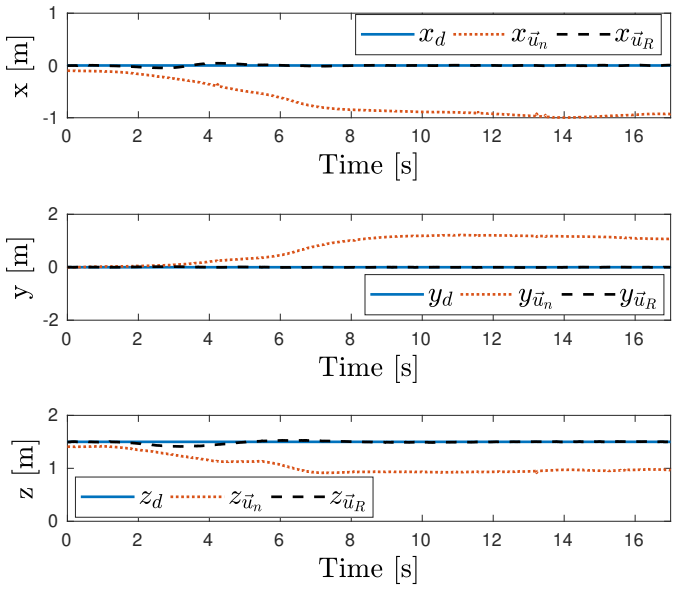


Figure 5.8 – Performance of the quadrotor against the later weight disturbance.

In Figure 5.9 the disturbance estimation is presented. Notice here from $t \geq 2$s when the observer is activated the disturbance is compensated and then the system behavior is barely affected. However at the same time but with the observer deactivated, the real values of the disturbances appear and the system performance is harshly degraded. Notice that $\hat{d}_z$ have similar behaviors. This is due the mass was not dropped immediately, for safety reasons, in the test when the observer was not activated. This fact can be appreciated in the video.

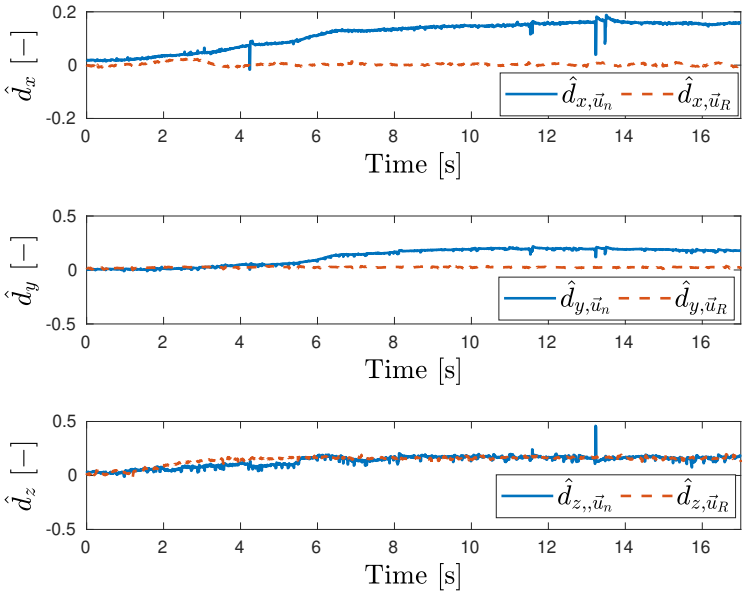


Figure 5.9 – Estimation of the weight disturbance.

Motor failure

The goal of this experiment, is to analyze the aerial vehicle performance when a motor failure appears. Several researchers, for this kind of problems, propose solutions based on FTC algorithms such that, an identification, isolation and compensation of the failure need to be done. In our case, we propose to solve this problem only using the proposed robust control scheme. Therefore, a virtual motor failure is considered when reducing its efficiency of the frontal left motor (M_1 in Fig 5.10) in 40%. The desired altitude for this test is $z_d = 1.5$m and the fault is applied at time $t = 4.5$s. Observe in Figures 5.11 to 5.13 the vehicle performance when a motor failure is applied.

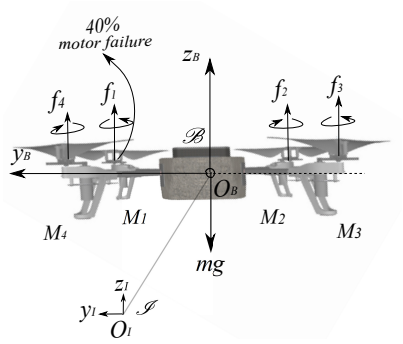


Figure 5.10 – Partial rotor failure in a quadcopter.

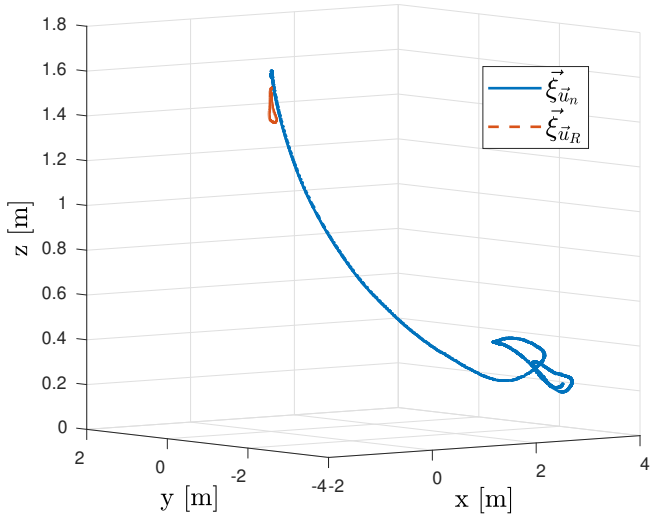


Figure 5.11 – Vehicle's performance in $3D$ against a rotor failure.

In Figure 5.12, the states behavior in the three axes are illustrated. Observe in this figure that the closed-loop system is harshly affected when the linear controller is used. However, note that when applying the robust control strategy, the drone performance remains stable. In Figure 5.13 the disturbance estimation is presented. Notice in this figure in red line, when the proposed controller is applied, it could be appreciated that no disturbances are presented because the proposed scheme estimates them considering a desired system performance and then, compensate this external/internal disturbances. However, a small degradation can be appreciated in the altitude.

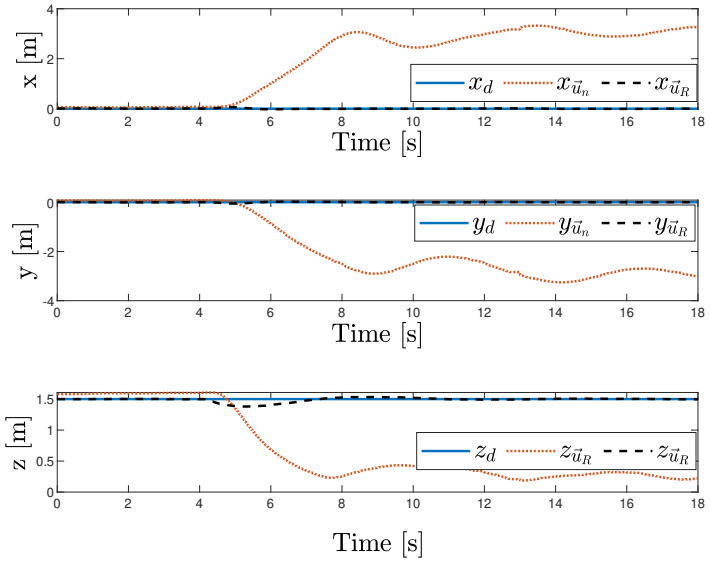


Figure 5.12 – Performance of the quadrotor against a rotor's fault.

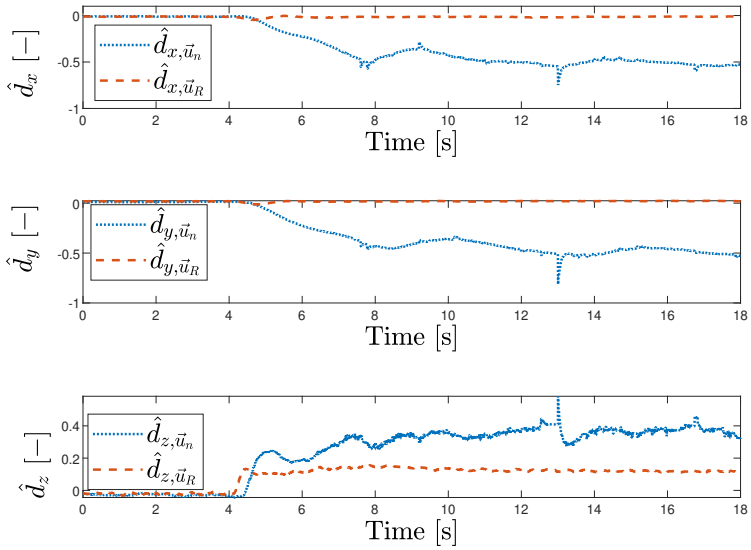


Figure 5.13 – Estimation of the rotor failure

Moreover, when using a simple feedback controller without the observer, the system is sternly affected and the vehicle altitude is reduced to 0.2m even if the controller tries to recover the performance applying more energy to the affected motor, see Figure 5.14. It is worth to mention that in Fl-AIR framework, the operating range of motors is from 0 to 1. Letting the control algorithm operates out of these limits can lead to a crash of the vehicle or damage in the actuators. In addition, 40% was the maximum value of the virtual rotor failure that has been successfully applied to the proposed robust controller. This leads to the possibility to improve the performance of the robust control architecture for dealing with these issues.

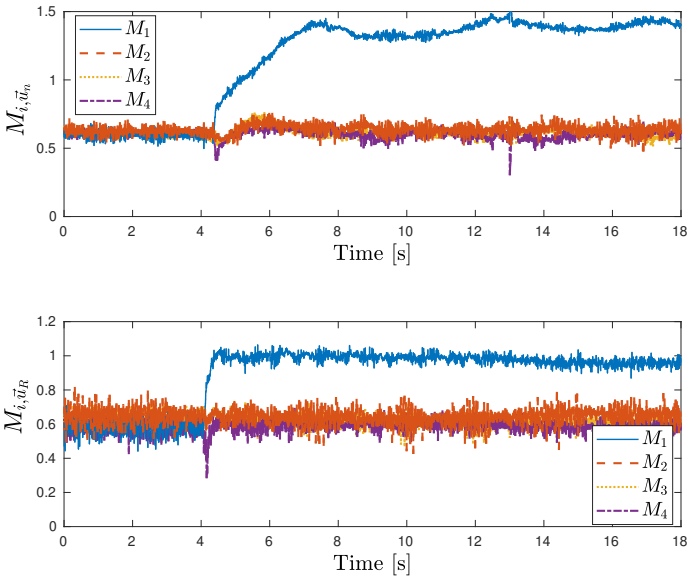


Figure 5.14 – Control actions of the motors.

5.3 Enhancing robust control architecture for LoE in rotors

In this section, the challenge will be to design a rotor fault observer for improving the performance of the closed-loop system when using the robust controller addressed in section 5.2. Note from (5.8), in order to compensate external disturbances affecting the vehicle, the estimation of $\vec{u}_\zeta$ through (5.11) is only necessary, nevertheless, for rotors fault estimation, this variable needs to be reformulated. In addition, from (5.2) observe that u_i is a function of the motors force f_i. We assume, in this work, that the thrust of each propeller is directly controlled and attached in the z axis of the vehicle, i.e., the blade flapping effect is not considered.

Notice that (5.8) has the same structure that (5.2), therefore, $\vec{u}_n$ and $\vec{u}_\zeta$ are also vectors of four components and functions of the forces f_i. As stated before, the force i produced by $i-$th motor can be computed using its angular rate w_{M_i}, such that, $f_i \approx k_f w_{M_i}^2$. Therefore, it is also possible to have a relation between the $i-$th motor and the combination of the control inputs u_i, $i : 1...4$.

This relation is often obtained when the control laws u_i need to be transformed into the real control inputs, i.e., the actuators (motors). For the quadrotor vehicle with parallel motors, we have

$$M = H\vec{u}_R \tag{5.13}$$

with

$$M = \begin{bmatrix} M_1(t) \\ M_2(t) \\ M_3(t) \\ M_4(t) \end{bmatrix}; \qquad \vec{u}_R = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} = \begin{bmatrix} u_{n_1} + u_{\zeta_1} \\ u_{n_2} + u_{\zeta_2} \\ u_{n_3} + u_{\zeta_3} \\ u_{n_4} + u_{\zeta_4} \end{bmatrix};$$

and H denotes the allocation control matrix with the form

$$H = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix}$$

Hence, M can be also separated as $M = M_n + M_\zeta$; the first one corresponding for the values computed with the control law and the second one for the disturbances rejection. Therefore, M_ζ is the vector parameter that should compensate the undesired performances in the vehicle. Analyzing only the disturbance part, it follows that $M_\zeta = H\vec{u}_\zeta$. Rewriting it, in the scalar form, it follows that

$$\begin{aligned} M_{\zeta_1} &= u_{\zeta_1} + u_{\zeta_2} + u_{\zeta_3} - u_{\zeta_4} \\ M_{\zeta_2} &= u_{\zeta_1} - u_{\zeta_2} + u_{\zeta_3} + u_{\zeta_4} \\ M_{\zeta_3} &= u_{\zeta_1} - u_{\zeta_2} - u_{\zeta_3} - u_{\zeta_4} \\ M_{\zeta_4} &= u_{\zeta_1} + u_{\zeta_2} - u_{\zeta_3} + u_{\zeta_4} \end{aligned} \tag{5.14}$$

From (5.14), observe that $\vec{u}_{\zeta_1}$ is common for the four rotors and it can be considered as the altitude disturbance in the vehicle and affects the efficiency of all rotors power at the same time. Therefore, its contribution for detecting if one motor is losing efficiency is less important. Nevertheless, notice also from (5.14) that the combination of the terms $\vec{u}_{\zeta_j}$ for $j : 2,3,4$; has a severe impact in each rotor. Hence from the previous analysis, it is possible to propose the following rotors fault estimator

$$\begin{aligned} \hat{M}_{\zeta_1} &= u_{\zeta_2} + u_{\zeta_3} - u_{\zeta_4} \\ \hat{M}_{\zeta_2} &= -u_{\zeta_2} + u_{\zeta_3} + u_{\zeta_4} \\ \hat{M}_{\zeta_3} &= -u_{\zeta_2} - u_{\zeta_3} - u_{\zeta_4} \\ \hat{M}_{\zeta_4} &= u_{\zeta_2} - u_{\zeta_3} + u_{\zeta_4} \\ \hat{M}_{th} &= u_{\zeta_1}. \end{aligned} \tag{5.15}$$

with $\hat{M}_{\zeta_i}$ is the disturbance parameter estimation for motor i.

The analysis of equation (5.15) can be done in two scenarios:

1. *without rotors fault* (conventional flights), here the values of $\hat{M}_{\zeta_i}$ for $i:1:4$; are relatively small but they should not be necessarily null, $\hat{M}_{th}$ is increased and the external and unknown disturbances can be estimated using only (5.11).

2. *with rotors fault*, in this scenario the values of $\hat{M}_{\zeta_i}$ for $i:1:4$; increase, degrading severally the performance of the vehicle and it can be seen as LoE in motors.

From the vehicle configuration and physical characteristics, observe that if one parameter of $\hat{M}_{\zeta_i}$ reaches a critical value ϵ_m, the control law will not be capable of keeping the vehicle stable at hover position, and thus, an inevitable crash will arise. This critical value is chosen such that a minimal main thrust (u_1) is required to overcome the weight of the vehicle (mg), i.e.,

$$\epsilon_m = \min u_1 = mg + \delta.$$

with δ is a small positive constant. Therefore, two flight modes are proposed for scenario (2)

$$\begin{aligned} S_M: &\quad \hat{M}_{id} \leq \epsilon_m, \\ E_M: &\quad \hat{M}_{id} > \epsilon_m, \end{aligned}$$

where S_M and E_M stand for the safe and emergency modes, respectively.

The enhanced control architecture is depicted in Figure 5.15. $\hat{d}_\xi$ and $\hat{d}_\eta$ correspond to the internal/external disturbances on the position and attitude subsystems and $\vec{\xi}_d$ is the desired position. $\vec{u}_R$ represents the vector control input containing the main thrust and the control torques respectively. ζ_i means the external disturbances and ϵ_m symbolizes the desired threshold.

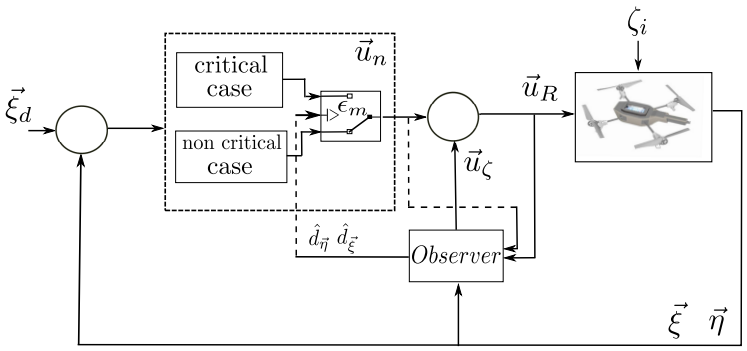


Figure 5.15 – Block diagram of the enhanced control architecture.

5.3.1 Numerical and practical validation

The robust control scheme developed in (5.8) and (5.15) were validated in numerical simulations and real-time experiments. The experimental and numerical tests were performed using Fl-AIR and the quadrotor Ar. Drone 2.0. The control parameters for simulations and experiments are the same as presented in Table 5.1. A video of the experimental results can be seen in https://youtu.be/OHoGpP7s12w.

The proposed architecture is validated using the following priorities

- Assuring the attitude stabilization at hover of the aerial vehicle, i.e., $\vec{\eta} \rightarrow \vec{\eta}_d$, where $\vec{\eta} = [\theta\ \phi\ \psi]^T$, $\vec{\eta}_d$ defines the desired attitude and ψ, θ and ϕ are the yaw, pitch and roll angles, respectively.
- Once the attitude performance is recovered, then $\vec{\xi} \rightarrow \vec{\xi}_d$, where $\vec{\xi}(t) = [x, y, z]^T$, $\vec{\xi}_d$ denotes the desired position.

For simulations and experimental flight tests and for solving the practical stability of the vehicle, two flying modes are considered in rotors failures. The first one is called safe mode - S_M or non critical case. Here, the proposed robust architecture can attenuate internal/external disturbances such as: wind, uncertainties into the model, etc, in such a way that, the practical stability of the quadrotor can be guaranteed. The non critical case is considered when rotors faults are less than a desired threshold ϵ_m.

The second one, the emergency mode - E_M - or critical case, uses the proposed robust scheme to detect important loss of effectiveness in the rotors and assure a safe landing. The critical case is activated when rotors failures are greater than ϵ_m. In this critical case, one degree of freedom is lost into the system and the control architecture is reconfigured for maintaining the aerial drone in flight or ensuring a safe landing. The control reconfiguration is given in following procedure :

1. The controllability of the yaw angle (ψ) is lost when the rotor failure exceeds the threshold.
2. The vehicle turns along its z axis and the yaw angle rate ($\dot{\psi}$) is controlled at high velocities for recovering its practical stability.
3. The practical stability is recovered trying to reach the desired altitude, z_d. If z_d is not reached quickly the safe landing is activated.
4. The drone lands in a safe mode.

For simulations and flight tests, the following assumptions are taken into account:

Assumption 6. *Physical constraints of the rotors, as damage in the propeller, are also considered as Loss of Effectiveness.*

Assumption 7. *Motors are constrained with a maximum value of revolutions per minute (RPM) for physical safety reasons.*

Assumption 8. *For the practical validation and keeping the integrity of the prototypes, it is assumed that only individual partial faults in the actuators are presented in the system, i.e., not simultaneous rotors fault are considered in this work.*

Numerical results

Two rotors fault scenarios are introduced for numerical validation. The first one consists in apply a LoE in motor M_1 into the aerial robot while it is performing a non hover position. For the second one, a critical rotor failure in M_1 is injected when the vehicle is at hover position and the emergency landing is required.

Rotor fault in a non hover position

The goal of this scenario is to show the performance of the quadrotor, when a rotor fault of 30% in motor M_1 is applied while performing a non hover position. The initial and desired conditions, for this scenario, are the following: $\vec{\xi}(t_0 = 0) = [-2,\ 0,\ 0]^T$m and $\vec{\xi}_d = [3,\ 0,\ 2]^T$m, respectively, moreover $\epsilon_m = 0.4$.

In Figure 5.16 the performance of the vehicle in 3D space when performing the desired task is addressed. Notice that when using the proposed architecture and the fault is injected, the whole system overcomes it and converges to the desired final point (solid line). In other case, when using only $\vec{u}_n$, the performance of the vehicle is degraded (dotted line). In other case, the performance of the vehicle is degraded, as can be verified in Figure 5.17. Notice from this figure, when the fault is injected at $t = 6.5s$, the y and z axes are slightly more compromised. Nevertheless, when using the proposed approach, at $t = 7s$ the quadrotor is recovered and thus, it can converge to the desired trajectory.

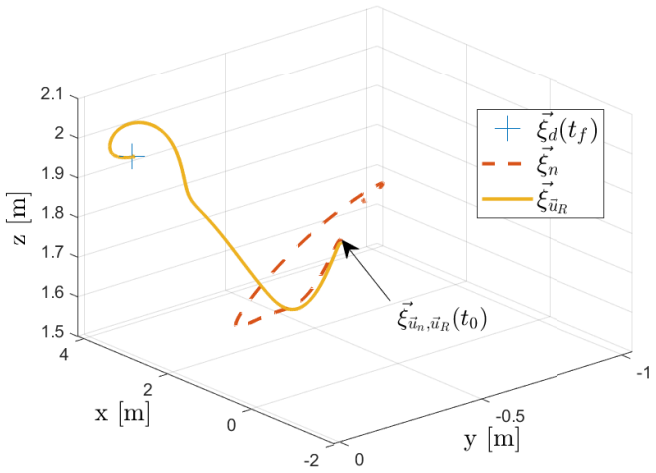


Figure 5.16 – Quadrotor position performance when using a feedback controller and the proposed architecture.

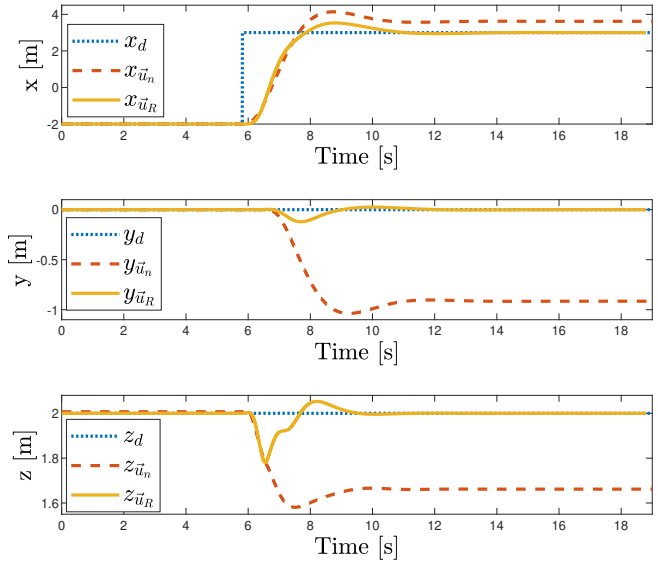


Figure 5.17 – Scenario 1.- Performance of the quadrotor subject to a LoE of 30% in M_1.

In Figure 5.18 the performance of the proposed rotors faults estimator is shown. It can be observed that the fault occurs at $t = 6.5s$. Moreover, the performance of the rotor inputs is displayed in 5.19. Notice the slightly difference when the scheme is activated or deactivated. In order to compensate the LoE in motor M_1 when using the scheme's feedback, motor M_3 reduces its velocity to maintain the drone flying in the desired path.

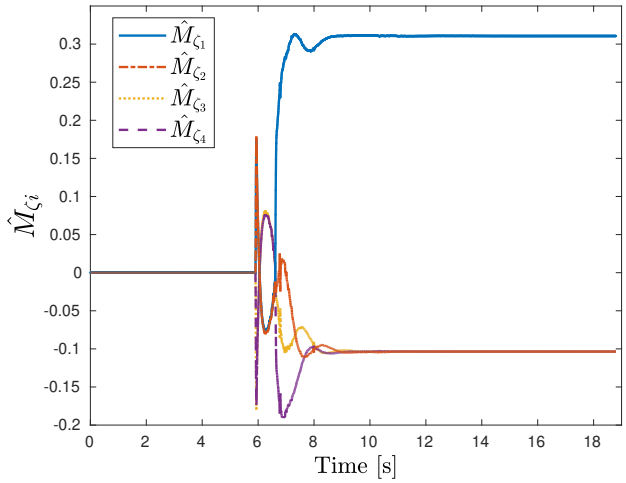


Figure 5.18 – Scenario 1.- Performance of the rotor fault estimator $\hat{M}_{\zeta_i}$, $i = 1, \cdots, 4$.

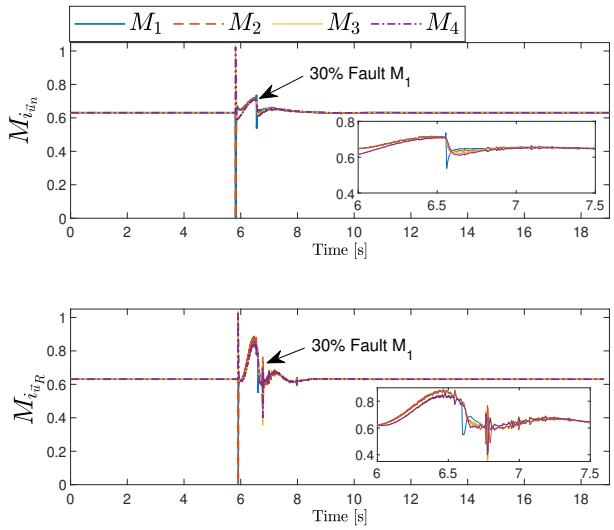


Figure 5.19 – Scenario 1.- Control action's behavior of each of the rotors when using $\vec{u}_n$ and $\vec{u}_R$ controllers.

A critical rotor fault (emergency mode)

In this scenario the aerial vehicle is at hover at the position $\vec{\xi}(t_0 = 0) = [0,\ 0,\ 2]^T$m. A critical rotor failure in motor M_1, emulated by the loss of 60% of its effectiveness, is introduced in this scenario at $t = 4.5s$. Here, the fault exceeds the threshold chosen $\epsilon_m = 0.4$, and then, the control algorithm switches into emergency mode. In the reconfiguration control, the yaw angle is lost and the desired yaw velocity is chosen as $\dot{\psi}_d = 10$rad/s.

Figures 5.20 shows the performance of the quadrotor subject to a critical rotor failure. From this figure, notice that when the fault is injected instantaneously the vehicle crashes when using only the state feedback controller $\vec{u}_n = -K\vec{x}$. However, when using proposed control scheme $\vec{u}_R$ the quadrotor recovers quickly its altitude (at $t = 6s$) for then, landing in a safe way.

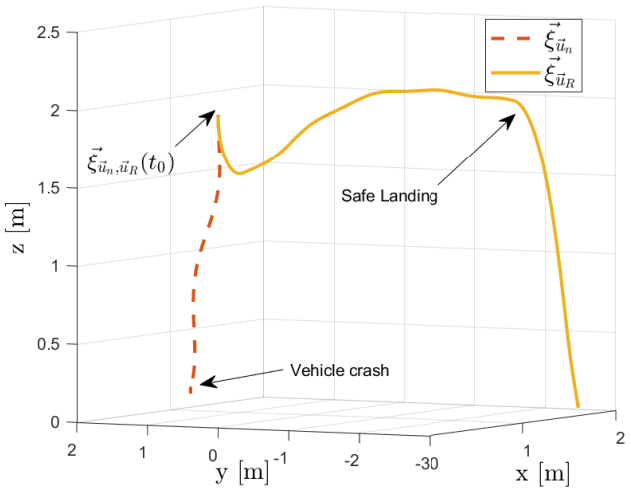


Figure 5.20 – Position performance of the vehicle subject to a critical failure in M_1.

In Figure 5.21 the performance of the fault estimator is presented. Notice that the fault is injected at $t = 4.5s$ and some small oscillations appear are as result of the vehicle is turning on its own axis. Nevertheless, the estimation has almost reached the exact value of the injected fault. This can be confirmed in Figure 5.22. Observe in here, the motors M_2, M_3 and M_4 as a result that the control scheme lost the controllability of the yaw angle. Remark that the magnitude of the motor M_1 is reduced to a maximum of 40% of its effectiveness. Hence, rotors M_2, M_3 and M_4 change its velocity, even reaching and passing the limits of saturation, in order to recover the quadrotor subject to the rotor fault.

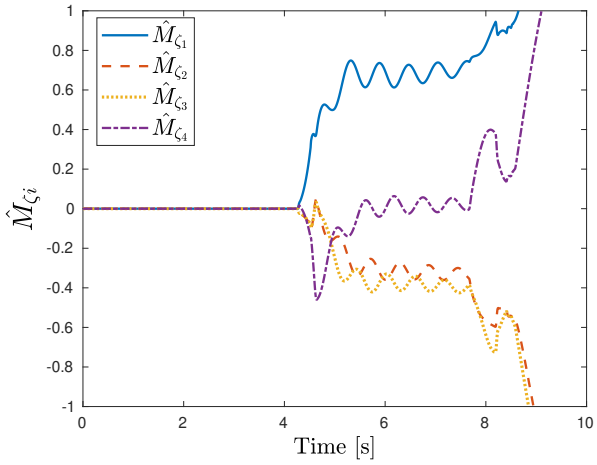


Figure 5.21 – Scenario 2.- Performance of the rotor fault estimator $\hat{M}_{\zeta_i}$, $i = 1, \cdots, 4$.

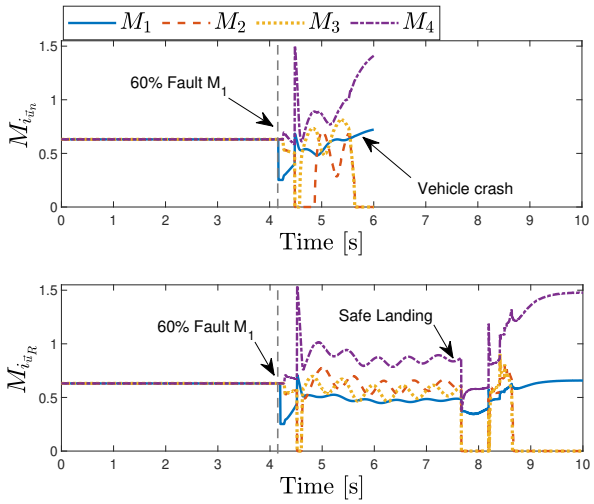


Figure 5.22 – Scenario 2.- Rotors control action's performances when a critical fault in motor M_1 is applied.

Experimental results

In this section, four different motor fault scenarios are introduced to validate in real-time tests the proposed robust control scheme considering $\epsilon_m = 0.4$. Thus, they are defined as follows:

1. A motor fault is applied in different times in each motor (academic example).
2. A motor fault is injected in M_1 while the drone is flying to reach a desired position.
3. A critical fault (50% LoE) is induced in a motor when the vehicle is at hover position.
4. A critical fault (60% LoE) is applied in a motor and the emergency landing is applied.

Scenario 1: The academic example

In this scenario, non critical rotors faults are injected into the system, i.e., each $10s$ a virtual fault of 30% LoE is applied on each rotor. The goal of this scenario is to show the good accuracy of the proposed approach to estimate and compensate the virtual fault. The initial conditions of the system are: $x(0) = y(0) = 0$ and $z = 2$ in meters and the goal is to keep the robot at hover position.

Observe in Figure 5.23 at 3s$\leq t \leq$ 8s the virtual fault in the rotor M_1 was injected. Similarly, for the motor M_2 the fault was induced at 13s$\leq t \leq$ 18s, then at 23s$\leq t \leq$ 28s for the rotor M_3 and finally at 33s$\leq t \leq$ 42s in rotor M_4. Notice that the proposed architecture is capable to continue estimating almost the total value of the injected fault, as we proposed in (5.15), even if the control law is not using the feedback of the proposed scheme.

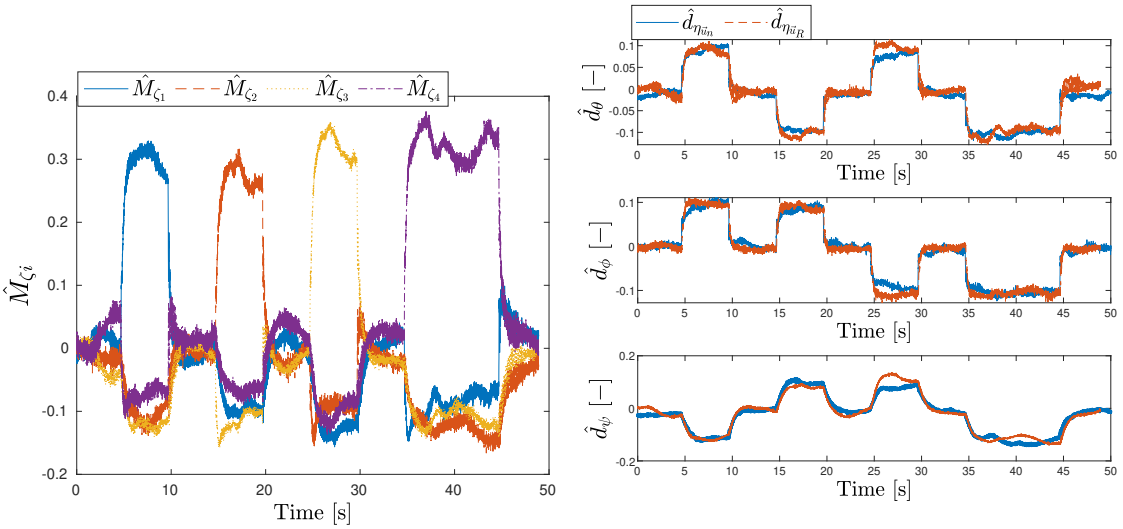


Figure 5.23 – Performance of the rotor faults estimator $\hat{M}_{\zeta_i}$ and the disturbance estimations (5.11) of the attitude system of the quadrotor .

Furthermore, in Figure 5.24, when the fault is induced, each motor suffers a degradation on its performance. Here, it is worth to mention that the operating range of the rotors in Fl-AIR's framework is between 0 to 1. Thus, when the rotor fault is injected, its effectiveness is changes to 0 - 0.7, in this particular scenario. Notice in this figure, that when the fault is injected and the scheme is active, the rotor reacts faster to compensate this fault and the vehicle converges to the desired reference. This can be checked in Figure 5.25 in each axis, where the z and x axes are slightly more penalized when the rotor fault is injected. Notice that according to number of the rotors, this will affect the performance of the vehicle, moving forward or backward respectively (positive or negative sign in the graph).

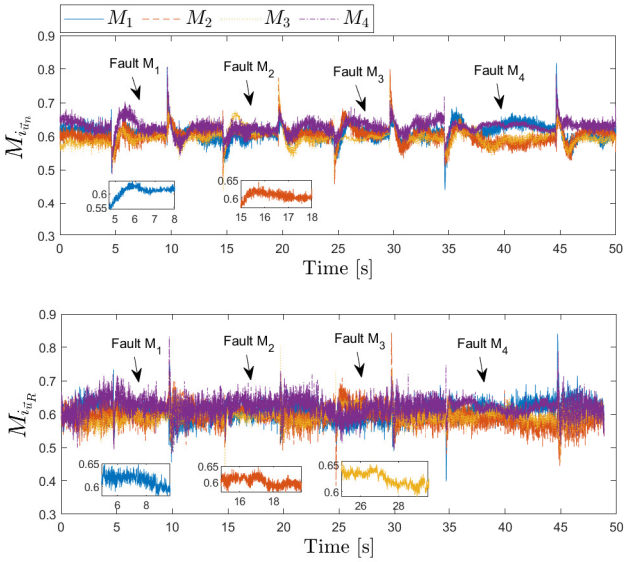


Figure 5.24 – Scenario 1.- Control rotors performance when using the linear controller (top graph) and the proposed scheme.

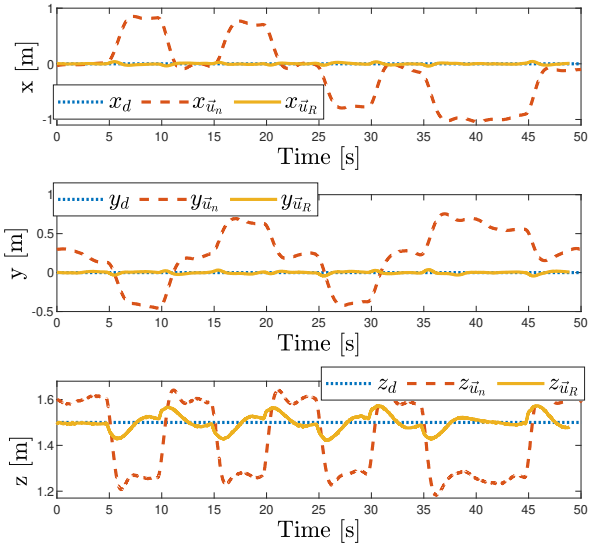


Figure 5.25 – Scenario 1.- Performance of the quadrotor position subject to a sequence of rotors failures.

Scenario 2: Rotor fault in a non hover position

Most of the aerial robot applications imply long displacements and when a rotor fault occurs the drone crashes. The goal of the second scenario is to show the performance of the quadrotor when moving from a point A to a point B and then, a degradation of 30% of the effectiveness in motor M_1 on time $t = 6s$ is virtually induced. For this end, the initial and desired conditions are as follows $\xi(t_0 = 0) = [-2,\ 0,\ 2]^T$m and $\xi_d = [3,\ 0,\ 2]^T$m, respectively.

In Figure 5.26 the behaviors of the vehicle in 3D space when performing the desired task, using two controllers, are introduced. Observe that when the fault is injected and the proposed architecture is applied, the quadrotor overcomes it and converges to the desired final point. Otherwise, the performance of the vehicle is degraded, as can be verified in Figure 5.27.

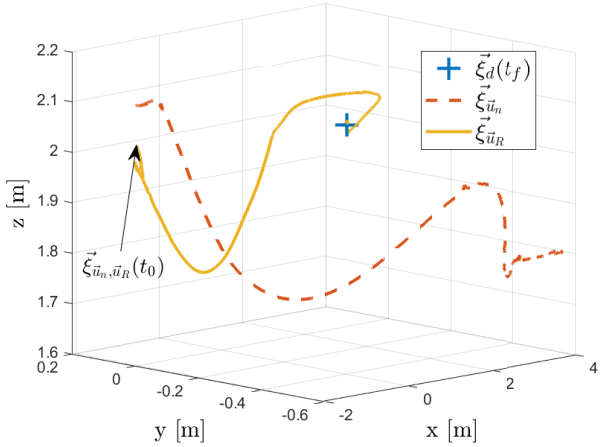


Figure 5.26 – Scenario 2.- Position performance of the vehicle when using the linear controller and the proposed architecture, and a rotor fault is induced in a non hover position.

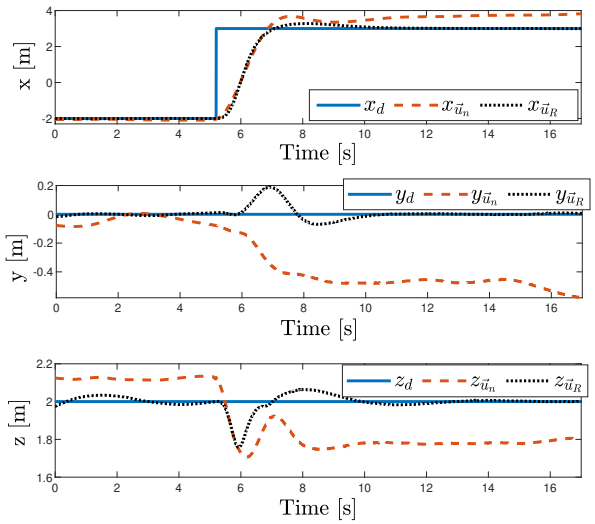


Figure 5.27 – Scenario 2.- Performance of the quadrotor position subject to a rotor fault while moving to a desired position $\vec{\xi}_d(t_f)$.

In Figure 5.28 the estimation of the motors fault is presented. Notice that the observer keeps estimating the fault even if it was compensated. Similarly in Figure 5.29 the motor control input performances are depicted. A slightly difference between these graphs can be seen. In order to compensate the magnitude of the rotor fault, when using the proposed scheme, motor M_2 immediately switch to zero its velocity to maintain the drone flying in the desired trajectory.

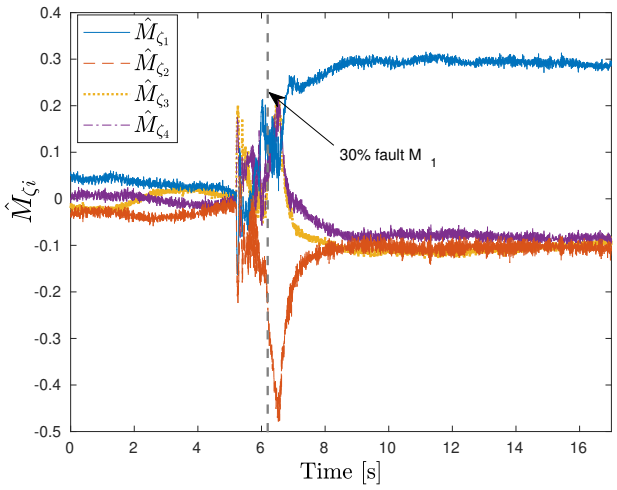


Figure 5.28 – Scenario 2.- Rotors faults estimations $\hat{M}_{\zeta_i}$, $i = 1, \cdots, 4$.

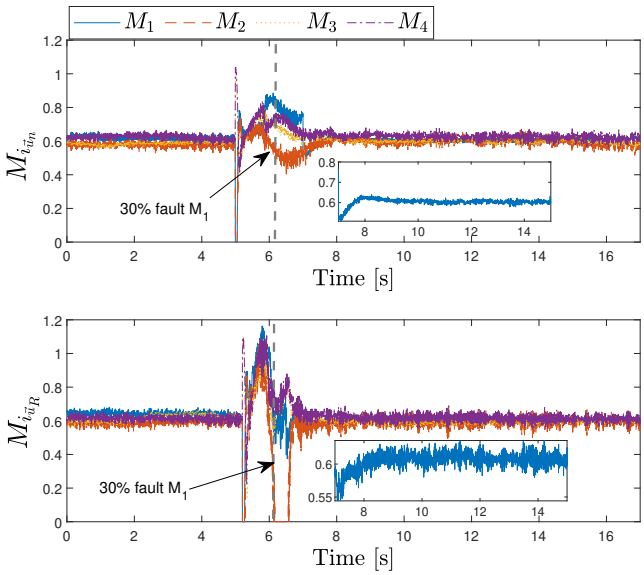


Figure 5.29 – Scenario 2.- Rotors control action's performances, when a LoE of 30% in M_1 is applied during a non hover position.

Scenario 3: A critical rotor failure: searching for a safe zone for landing

When a quadrotor is performing a task and a critical rotor fault occurs is not always possible to land safely. The goal of this test is to present the performance of the vehicle when a critical fault appears while it is flying in zones where, a safety landing immediately is not possible. Thus, searching for a safe zone must be done. For that, suppose the quadrotor is located at $\vec{\xi}(t_0 = 0) = [0,0,2]^T$m, and a virtual fault of 50% of LoE in the motor M_1 is introduced. Then, according to the control reconfiguration procedure, the control law switches to the emergency mode. Here, the position yaw control is lost and only the yaw rate can be controlled. Once the drone attitude is recovered, it reaches quickly its desired altitude z_d and then a safe set-point; $\vec{\xi}_d = [2,0,2]^T$m, is sent to the quadrotor simulating a coordinate where the vehicle can be landed in safety mode.

In Figure 5.30 the 3D performance of the vehicle when a critical fault injected in motor M_1 is depicted. Notice here that, on the one hand, when the fault is injected, the state feedback controller is not enough to compensate it and the system performance is harshly degraded, making the drone crashes (dotted line). On the other hand, when the aerial drone is flying with the proposed control scheme, its performance is barely degraded such that, safe displacements and safe landings are possibles.

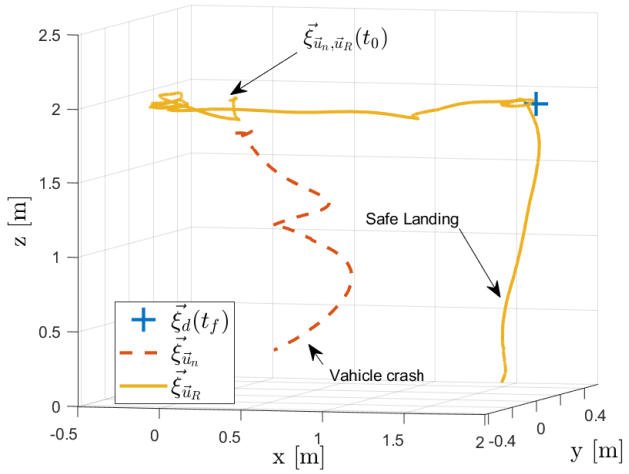


Figure 5.30 – Scenario 3.- Performance of the vehicle when applying a motor fault of 50% and, two different controllers are used.

Figure 5.31 presents the behavior of the vehicle in the three axes when the fault in motor M_1 is injected. Observe that, when the virtual rotor fault is injected, the performance of the vehicle when using the state feedback controller is degraded. However, in the same figure in the $z-$ axis at 4s$\leq t \leq$ 6s, note that when using the proposed scheme the performance of the drone is barely degraded.

The variations in the x and y axis are due the quadrotor is turning on its own axis and simultaneously, controlling the error position. The safe reference is sent at $t = 12$s and once the vehicle arrives, the safe landing can be achieved. The performance of the proposed rotor fault estimator is shown in Figure 5.32. From this figure we can corroborate that the fault was injected at $t = 4.5$s and the well performance of the rotor fault observer when estimating the failure.

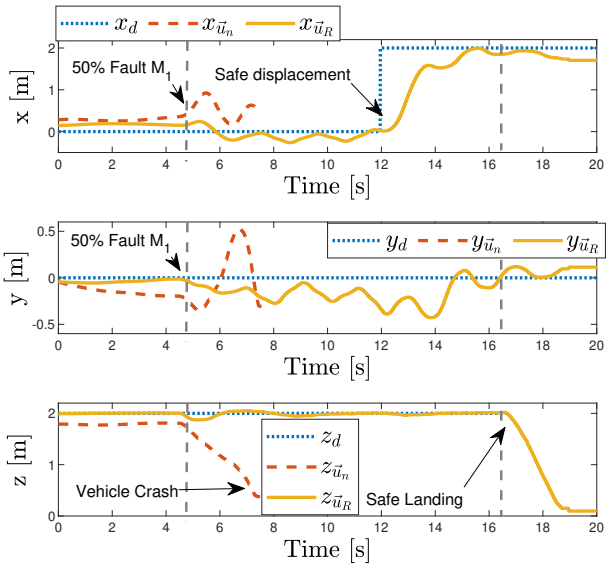


Figure 5.31 – Scenario 3.- Performance of the quadcopter position subject to a critical rotor fault of 50% in M_1.

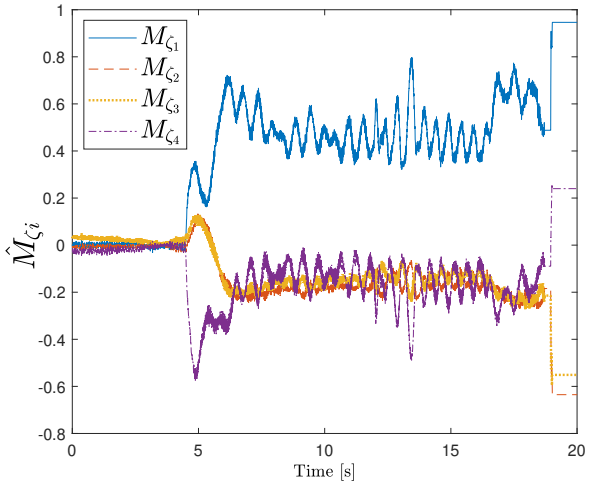


Figure 5.32 – Scenario 3.- Rotors faults estimations $\hat{M}_{\zeta_i}$, $i = 1, \cdots, 4$.

In Figure 5.33 the performances of the motors M_i subject to a rotor fault of 50% of LoE in M_1 are depicted. Notice that, once the fault is injected, the effectiveness range in motor M_1 is reduced 50%, in addition, remark that the motor M_4 turns off for some milliseconds (for compensating M_1) and, motors M_2 and M_3 increases its velocity to make possible the yaw velocity control.

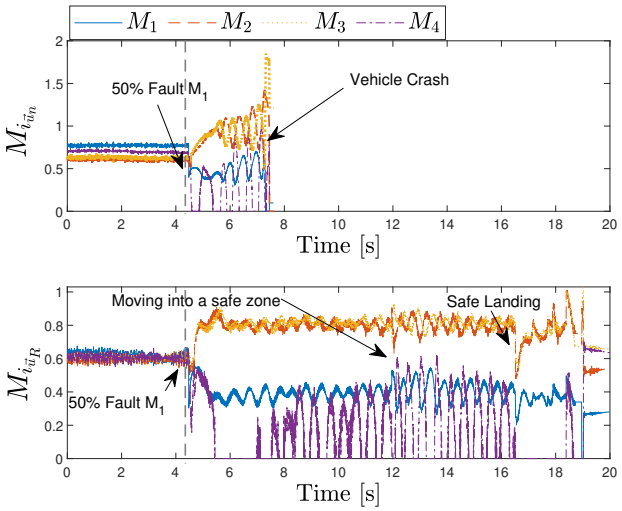


Figure 5.33 – Scenario 3.- Rotors control action's performances, when a critical fault of 50% in M_1 is applied.

Scenario 4: A critical rotor fault with emergency landing

An emergency landing is a prioritized flight mode made by an aircraft in response to an imminent or ongoing threat, e.g. rotors failures, in order to keep the safety operation of the vehicle. This scenario shows the performance of the drone when a LoE of 60% is induced in a motor. Thus, the scenario is set as follows: the drone is flying at hover $\xi = [0,0,2]^T$m, this position is taken as its initial position $\xi_{t_0=0}$, at time $t = 3.9$s the virtual failure in M_1 is applied and the vehicle performance degraded.

Notice from Figure 5.34 that when using only the linear controller the robot performance is instantly degraded and it crashes (dotted line). Nevertheless, when using the proposed architecture, the rotor fault observer estimates the motor fault that is compared with the threshold. As the threshold is overpassed, the control scheme switches into the emergency mode. This allows to recover the quadrotor attitude performance and the desired altitude. Nevertheless the aerial robot begins to lose altitude and then the condition for an emergency landing is activated. And thus at the end, the quadrotor lands safely (solid line).

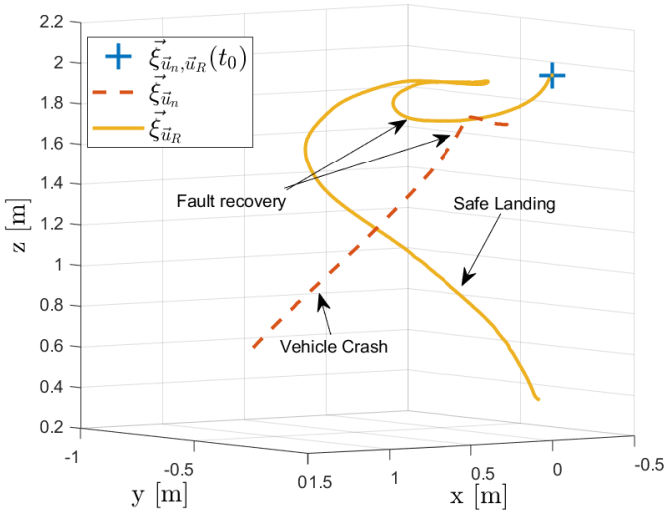


Figure 5.34 – Scenario 4.- Position performance of the vehicle when using the linear controller and the proposed architecture, and a critical rotor fault is induced.

In Figure 5.35 the estimations of $\hat{M}_{\zeta_i}$ are depicted. Notice here that their values oscillate after time $t = 4.2$s and they are produced when the aerial robot is turning in its vertical axis. Moreover, in Figure 5.36 the motors performances are illustrated. Notice that M_1 has 40% of tis effectiveness when the failure is applied, and then M_4 stops working for short intermittent milliseconds for compensating effectiveness in M_1. Observe also that M_2 and M_3 increase their effectiveness to recover the main thrust and recover the desired altitude, nevertheless it produces that the vehicle turns in its vertical axis.

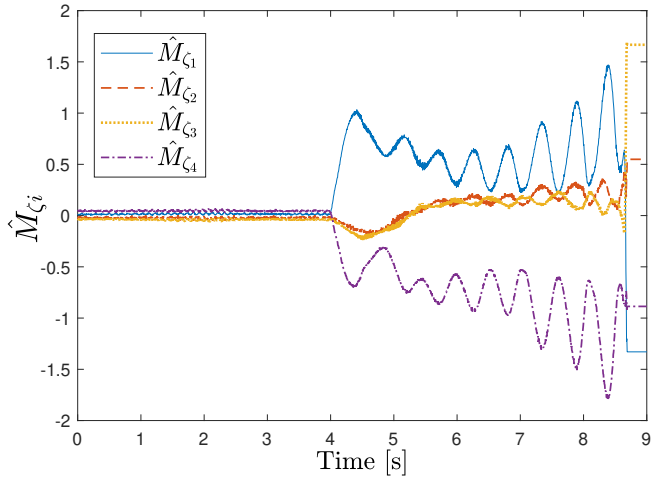


Figure 5.35 – Scenario 4.- Rotors faults estimations $\hat{M}_{\zeta_i}$, $i : 1, \cdots, 4$.

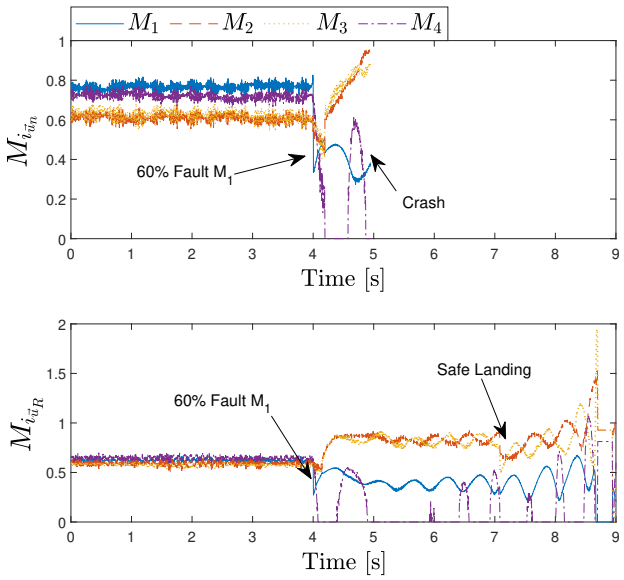


Figure 5.36 – Scenario 4.- Rotors control action's performances, when a critical fault of 60% in M_1 is applied.

5.3.2 Discussion

In this section, we have carried out a performance analysis, with the three experimental scenarios, of the advantages of the proposed architecture with respect to use only the linear controller. For this purpose, the performance indices and a qualitative comparison of the controllers are obtained. The performance indices are the Integral Absolute Error (IAE), and the Root Mean Square Error (RMSE). A summary of the analysis of the three experimental tests is presented in Table 5.2. It is worth to mention that, all the experiments were developed using the same control gains, i.e., re-tuning these variables was not needed. Concerning the steady state regime, from table results, it can be observed that the proposed robust control scheme achieves the best performance in all the axes. Nevertheless, it is true that is possible to have a better performance for the linear controller but in this case, re-tuning the control variables most be done for each experiment.

Table 5.2 – Performance indices of the three scenarios.

		Scenario 1		Scenario 2		Scenario 3	
		IAE	RMSE	IAE	RMSE	IAE	RMSE
z	u_q	3.34	0.19	inf	inf	inf	inf
	$\bar{u}$	0.39	0.04	1.52	0.41	1.8	0.35
x	u_q	13.02	1.18	inf	inf	inf	inf
	$\bar{u}$	5.29	1.02	5.55	0.46	2.17	0.24
y	u_q	6.01	0.38	inf	inf	inf	inf
	$\bar{u}$	0.35	0.04	1.49	0.22	0.98	0.09

Regarding the design, tuning and the implementation effort, the linear controller is easier to tune because its structure can be seen as a simple controller. However, the proposed scheme is composed mainly of an observer, a simple reference model of the vehicle and a band-pass filter. All these parameters need to be obtained experimentally in order to have a good approximation of the system. Nevertheless, once the observer is well-tuned for the prototype, it can be used with almost all kind of controllers. The advantage of using the proposed scheme is that it contains an immediate and smooth control action when undesired dynamics appears, especially rotors fault. This can be verified in Table 5.2, column two and three, where it can be observed that, using only a linear controller, when a critical rotor fault is injected, it wont be enough to overcome it.

For that, the good performance of the system is guaranteed when critical fault occurs and the proposed control scheme is being used. Nevertheless, as all kind of control schemes, there are some limitations that are related with the physical characteristics of the prototype. For example, we have observed from real-time flight tests that when injecting critical faults more than 70%, the observer sends the respective compensation which exceeds the capacity of effectiveness of all the motors, such that the quadrotor becomes unstable. This is mainly due the fact, the observer does not take into account the limit of operating range of the rotors. In addition, we have analyzed the Total Variation (TV) of the control input ($TV = \sum_{i=1}^{i=\infty} |u_{i+1} - u_i|$) as a measure of the control effort. This index can easily show the aggressively of the controller and can be seen as a measurement of the used energy by the control action. In Table 5.3, scenario 1, the TV of the first experiment is presented.

Table 5.3 – Total Variation (TV) of the control inputs.

Scenario 1								
	u_2	u_3	u_4	u_1	M_1	M_2	M_3	M_4
u_q	26.80	39.43	95.11	4.72	64.68	116.12	54.06	89.43
$\bar{u}$	46.62	65.33	112.80	6.52	76.10	150.75	58.20	117.04
Scenario 2								
	u_2	u_3	u_4	u_1	M_1	M_2	M_3	M_4
u_q	inf (17.35)	inf (29.13)	inf (32.59)	inf (4.70)	inf (26.36)	inf (29.28)	inf (34.76)	inf (29.47)
$\bar{u}$	30.03 (21.54)	68.45 (32.14)	89.03 (40.61)	10.33 (6.07)	58.05 (31.39)	76.19 (37.62)	69.95 (36.02)	114.97 (40.80)
Scenario 3								
	u_2	u_3	u_4	u_1	M_1	M_2	M_3	M_4
u_q	inf (10.42)	inf (11.78)	inf (17.54)	inf (1.12)	inf (9.67)	inf (20.91)	inf (16.73)	inf (15.56)
$\bar{u}$	26.31 (12.31)	32.52 (14.90)	38.88 (18.68)	4.45 (1.98)	24.22 (12.75)	53.78 (27.51)	34.06 (20.09)	37.66 (18.64)

Remark from Figure 5.37, a similar performance of both controllers is encountered. Nevertheless, the $\vec{u}_n$ controller has a lower TV because the designed controller is not aggressive as the $\vec{u}_R$, but with the lack of the disturbance rejection property, since the $\vec{u}_n$ cannot deal with the set-point tracking problem in the presence of faults.

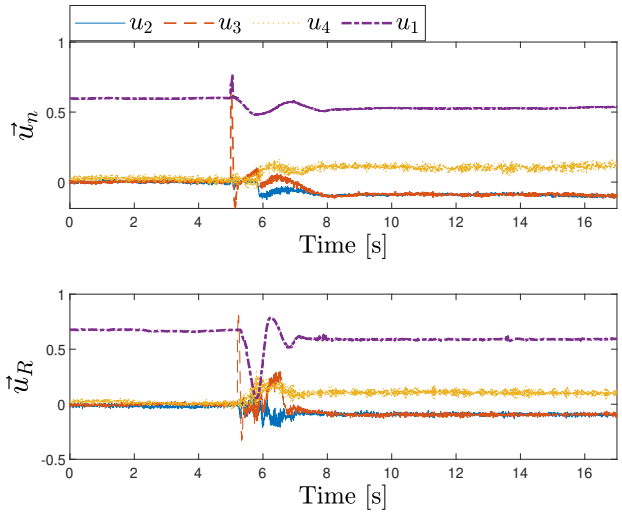


Figure 5.37 – Control inputs $\vec{u}_n$ and $\vec{u}_R$ of the first scenario.

Moreover, based on the performance of the control laws depicted in Figure 5.38, the TV of the second experiment is presented in Table 5.3, scenario 2. In contrast to the previous cases, the $\vec{u}_n$ controller presents an infinity TV when the fault was injected since the system becomes unstable. This is mainly due to the fact that, the control algorithm is not robust enough to reject this kind of disturbances. However, the $\vec{u}_R$ control can overcome the fault, even continue flying safely, thus, the safe landing can be carried-out.

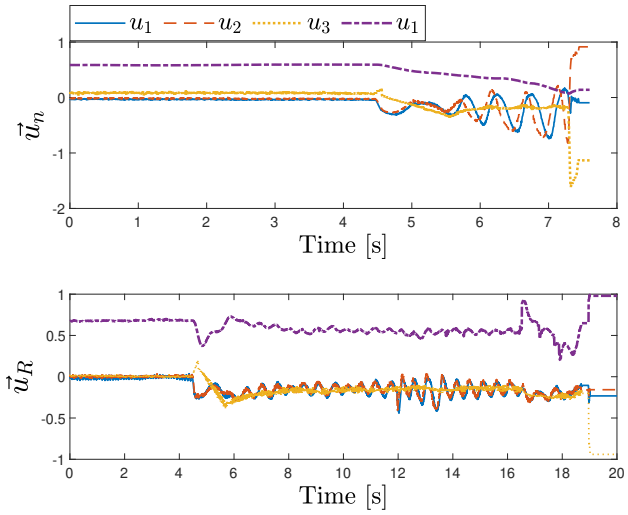


Figure 5.38 – Control inputs $\vec{u}_n$ and $\vec{u}_R$ of the second scenario.

Finally, in Table 5.3, scenario 3, the TV of the last experiment is provided. In this experiment, an aggressive fault was introduced, as can be observed in Figure 5.39. Therefore, the $\vec{u}_n$ controller is not able to continue flying and thus, it becomes unstable such that, the TV becomes infinity. Nevertheless, the $\vec{u}_R$ controller is able to overcome this problem since is prepared to deal with this kind of high disturbances. We show in parenthesis, the TV of these last two experiments, before the faults were injected.

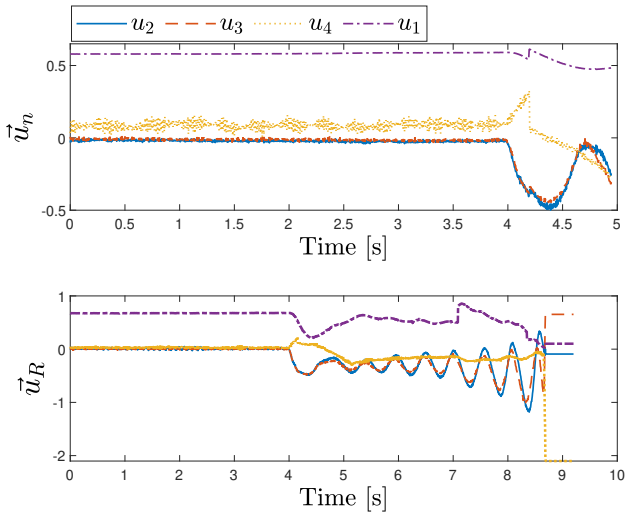


Figure 5.39 – Control inputs $\vec{u}_n$ and $\vec{u}_R$ of the third scenario.

5.4 Enhancing robust control architecture for handling rotor's constraints

Notice that equation (5.13) represents the real robust control input applied to the actuator M_i for controlling the aerial vehicle. Moreover and as stated before, rotors have physical constraints that can be seen as the limit of operating speed range $\bar{\sigma}_i$ and $\underline{\sigma}_i$ which describes the upper and lower bound of the $i-th$ motor, respectively. This brings out a non trivial problem with respect to, when is needed to saturate a controller or when is not. To make this clear let us introduce two cases; assuming the upper and lower bound of motors are $\bar{\sigma}_i = 1$, $\underline{\sigma}_i = 0$ respectively, and considering the control action as $\vec{u}_R = [0.80\ 0.05\ 0.05\ 0.05]^T$, this produce the control actions of motors $M = [0.95\ 0.75\ 0.75\ 0.85]^T$. Nevertheless, if $\vec{u}_R = [0.80\ 0.15\ 0.05\ 0.05]^T$, gives $M = [1.05\ 0.65\ 0.85\ 0.95]^T$.

Therefore notice that even if the values of $\vec{u}_{R_i}$ are almost the same for both cases, it is not obvious if actuators could be or not saturated, since that to saturate the actuators, it will depend of the set of control inputs and not only in one in particular. A popular solution is to bound each control input, $\vec{u}_{R_i}$, however, this is not at all an efficiently solution since, bounding the control input does not means that one of the actuators has not been saturated. For avoiding that, it is necessary to compute a set of the optimal bounded control inputs, $\vec{u}_R^*$, from $\vec{u}_R$.

Hence, $\vec{u}_R^*$ can be obtained as

$$\vec{u}_R^* = \vec{\delta}_M^T \vec{u}_R. \tag{5.16}$$

where $\vec{\delta}_M^T = [\delta_1\ \delta_2\ \delta_3\ \delta_4]$ with δ_i defining a factor for fulfilling the rotors constraints. The bounds for the motors control inputs can be find solving the following quadratic problem

$$\min_{\vec{\delta}_M} \ \frac{1}{2}\vec{\delta}_M^T Q \vec{\delta}_M + \vec{c}^T \vec{\delta_M} \tag{5.17}$$

$$st.\ A_M \vec{\delta}_M \leq \vec{b} \tag{5.18}$$

where $Q, \vec{c}, A_M, \vec{b}$ are matrices and vectors that will be defined later.

For computing matrices and vectors in (5.17) and (5.18), the following cost-to-go function can be defined

$$f_o = \sum_{i=1}^{4} Q_i (\vec{u}_{R_i} - \vec{u}_{R_i}^*)^2, \tag{5.19}$$

where Q_i is a weight value penalizing the control input $\vec{u}_{R_i}$. Developing the above equation, it follows that

$$f_o = \sum_{i=1}^{4} Q_i (\vec{u}_{R_i}^2 - 2\delta_i \vec{u}_{R_i}^2 + \delta_i^2 \vec{u}_{R_i}^2). \tag{5.20}$$

From (5.20), note that the term $\vec{u}_{R_i}^2$ does not affect to find the minimum of the quadratic problem, thus, it can be neglected. Rewriting the above equation in matrix form, it yields

$$fo = \frac{1}{2}\vec{\delta}_M^T Q \vec{\delta}_M + \vec{c}^T \vec{\delta}_M$$

with

$$Q = \begin{bmatrix} Q_1 u_{R_1}^2 & 0 & 0 & 0 \\ 0 & Q_2 u_{R_2}^2 & 0 & 0 \\ 0 & 0 & Q_3 u_{R_3}^2 & 0 \\ 0 & 0 & 0 & Q_4 u_{R_4}^2 \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} -Q_1 u_{R_1}^2 \\ -Q_2 u_{R_2}^2 \\ -Q_3 u_{R_3}^2 \\ -Q_4 u_{R_4}^2 \end{bmatrix}.$$

Let now introduce the constraints in the motors control input with the form $\underline{\sigma}_i \leq M_i \leq \bar{\sigma}_i$, where $\bar{\sigma}_i$ and $\underline{\sigma}_i$ are the upper and lower bounds of the motors, respectively. Moreover, define u_{R0}^* as the equilibrium point of the aerial robot at hover position, in other words, the u_1 needed to compensate the weight of the vehicle.

For finding the condition (5.18), consider the case of the motor $M_1 \leq |\bar{\sigma}_1|$. Thus, taking into account the weight compensation u_{R0}^* and using (5.13) with the optimal value of $\vec{u}_R$ as in (5.16), it follows that

$$|\sum_{i=1}^{4} \pm\delta_i \vec{u}_{Ri}| \leq |\bar{\sigma}_1| - u_{R0}^*. \tag{5.21}$$

Then, following the same procedure for the rest of motors and rewriting them in a matrix form, it follows that (5.18) can be founded with

$$A_M = \begin{bmatrix} +u_{R_1} & +u_{R_2} & +u_{R_3} & -u_{R_4} \\ +u_{R_1} & -u_{R_2} & +u_{R_3} & +u_{R_4} \\ +u_{R_1} & -u_{R_2} & -u_{R_3} & -u_{R_4} \\ +u_{R_1} & +u_{R_2} & -u_{R_3} & +u_{R_4} \\ -u_{R_1} & -u_{R_2} & -u_{R_3} & +u_{R_4} \\ -u_{R_1} & +u_{R_2} & -u_{R_3} & -u_{R_4} \\ -u_{R_1} & +u_{R_2} & +u_{R_3} & +u_{R_4} \\ -u_{R_1} & -u_{R_2} & +u_{R_3} & -u_{R_4} \end{bmatrix}, \vec{b} = \begin{bmatrix} \bar{\sigma}_1 - u^*_{R0} \\ \bar{\sigma}_2 - u^*_{R0} \\ \bar{\sigma}_3 - u^*_{R0} \\ \bar{\sigma}_4 - u^*_{R0} \\ -\underline{\sigma}_1 + u^*_{R0} \\ -\underline{\sigma}_2 + u^*_{R0} \\ -\underline{\sigma}_3 + u^*_{R0} \\ -\underline{\sigma}_4 + u^*_{R0} \end{bmatrix}. \tag{5.22}$$

Thus using (5.16) with constraints (5.17) and (5.18), it is possible to find the set of optimal bounded values of $\vec{u}_R$. The proposed enhanced bounded robust control architecture is depicted in Figure 5.40

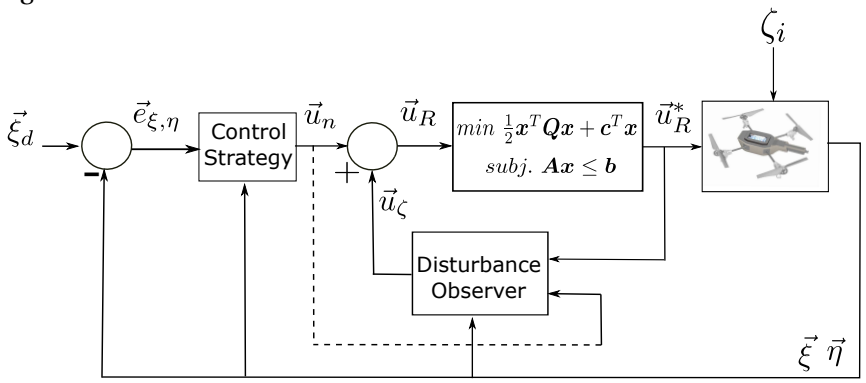


Figure 5.40 – Block diagram of the control structure. $\vec{u}_n$ and $\vec{u}_\zeta$ correspond to the the nominal control action and the control rejection part, respectively, ξ_d is the desired reference. $\vec{\xi}$, $\vec{\eta}$ mean the position and attitude states. $\vec{u}_R$ and $\vec{u}^*_R$ represent the proposed controller and the set of its optimal and bounded values, respectively. The quadratic problem block deals with the motors constraints and ζ_i represents external disturbances.

5.4.1 Experimental results of bounded control

The practical goal of these experiments is to validate the proposed control architecture. For better illustrate the good performance of the optimal bounded robust controller (5.16), it is compared, in two scenarios, with respect to its nominal form (5.8). It is worth to mention that both algorithms have a low computational cost, which permits to be implemented in low cost CPU. The experimental tests are performed in a quadrotor vehicle AR Drone 2 using the open source Fl-AIR. The optimal control problem described by (5.17), (5.18) was solved using the Goldfarb-Idnani algorithm [189] and implented based on the open source code *uQuadProg* [190]. The control parameters for the experimental validation are the same as presented in Table 5.1.

First scenario: aggressive wind-gust

Most of the aerial robot applications are developed in outdoors environments where robustness with respect to wind is a primary task. The goal of the first scenario is to show the performance of the quadcopter while it is at hover, subject to aggressive constant and intermittent wind-gust. For this end, a leaf blower *Bosch AVS1* was used to emulate external wind gust with an airflow speed of 60km/h and placed at 1.5m from the quadrotor, see Figure 5.41. The hover position is at: $x(0) = 0,\ y(0) = 0,\ z(0) = 1$, all in meters. A video of the experimental results can be seen in https://youtu.be/6Fhjyb1JVXY.

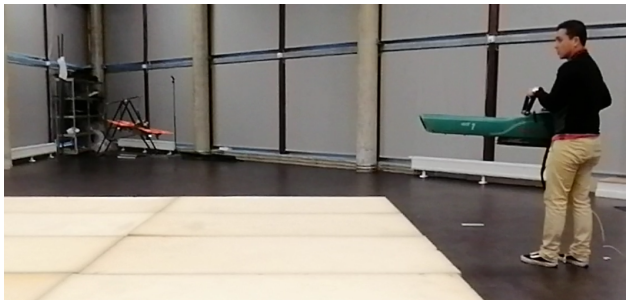


Figure 5.41 – Set-up of the wind-gust scenario.

In Figure 5.42 the performance of the vehicle in 3D space subject to high wind-gust is introduced. Notice the better performance of the proposed bounded control architecture $\vec{u}_R^*$ with respect to $\vec{u}_R$. The position behavior of the drone is depicted in Figure 5.43. In the experiment the wind-gust was directed to the $y-$axis the firsts 20s. Note that both approaches suffer a degradation in their performance, nevertheless it is clear that when using $\vec{u}_R^*$, this degradation (especially in z) is smoother, see Figure 5.44.

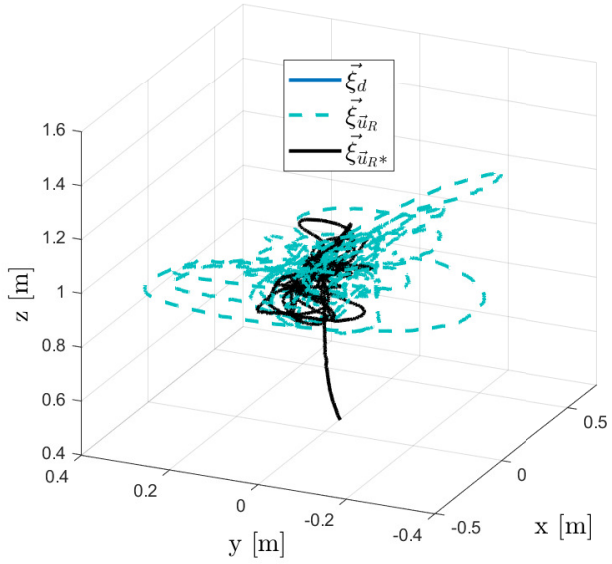


Figure 5.42 – Scenario 1.- System performance during flight tests.

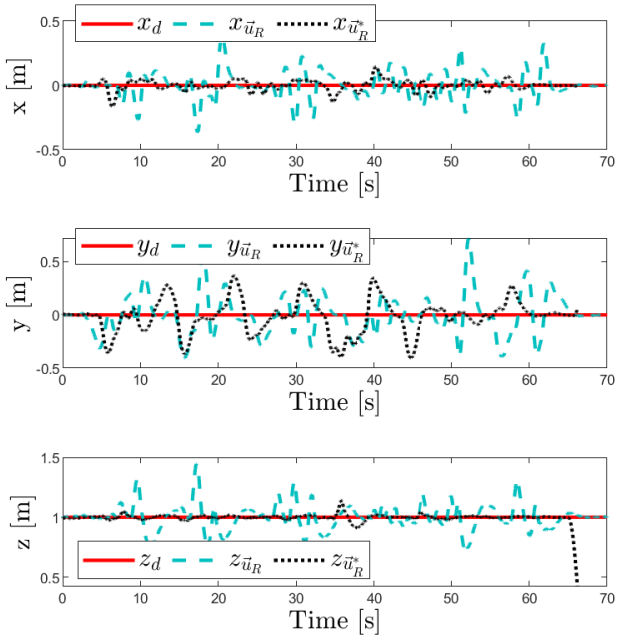


Figure 5.43 – Scenario 1.- Performance of the quadrotor position subject to wind-gust.

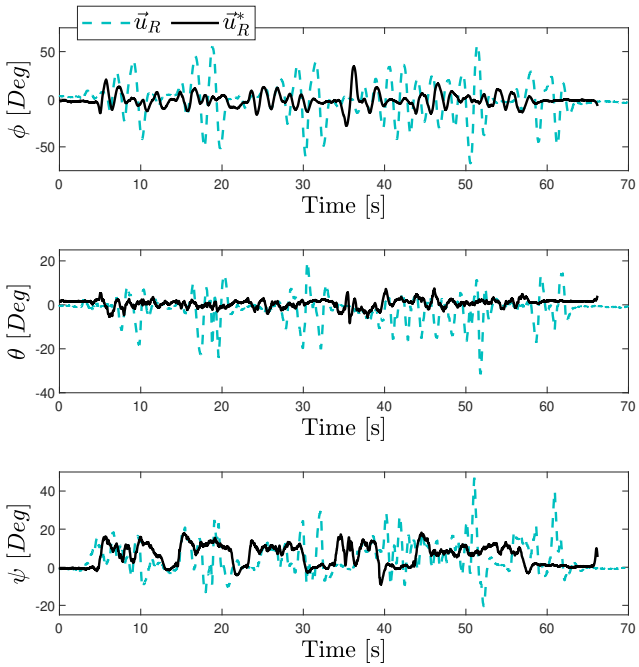


Figure 5.44 – Scenario 1.- Attitude Performance of the quadrotor system subject to wind-gust.

Figures 5.45, 5.46 depict the performance of the disturbance estimator and the motors control input. Notice that these both figures are extremely related. This due to the fact that the disturbance estimator sends the compensation (see Figure 5.45) without knowing the motor's constraints (for $\vec{u}_R$). This makes the control motors inputs exceed their limits (see Figure 5.46 in 18s$\leq t \leq$ 22s, then in 28s$\leq t \leq$ 32s and 48s$\leq t \leq$ 52s). However, when the vehicle is controlled by $\vec{u}_R^*$, the aforementioned problem does not appear.

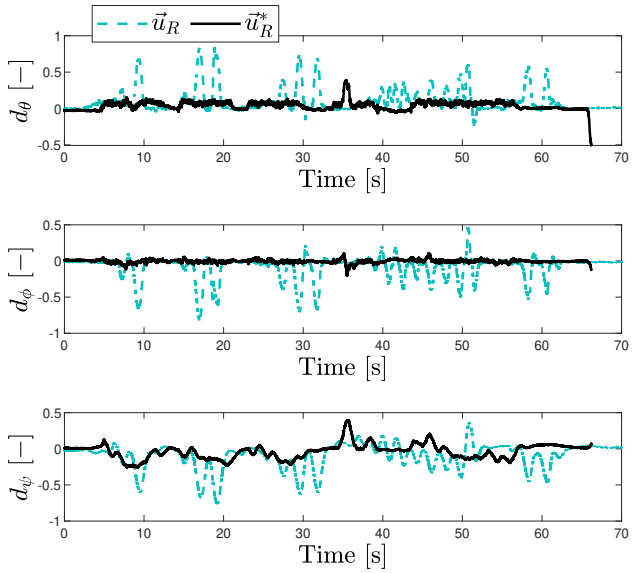


Figure 5.45 – Scenario 1.- Attitude disturbance estimations.

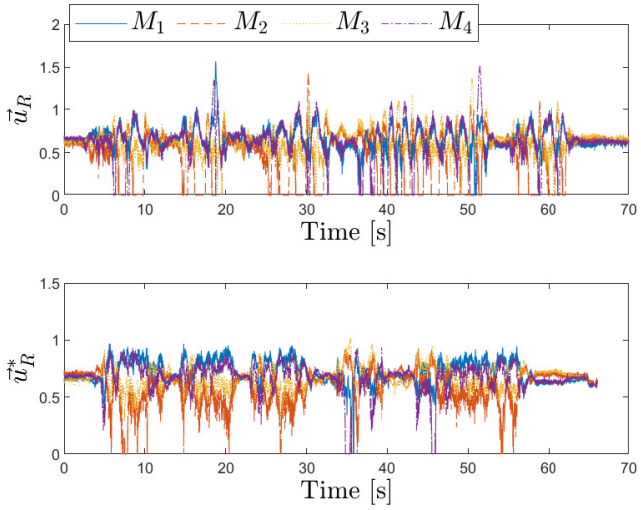


Figure 5.46 – Scenario 1.- Performance of the motor control actions.

Second scenario: loss of effectiveness in rotors when performing a trajectory

Conducting electrical inspections with drones has been an application developed through the last years. In this kind of tasks, the vehicle needs to be robust with respect to many factors, as wind-gust, possible loss of effectiveness in rotors (repetitive tasks), among others. The goal of the second scenario is to show the performance of the quadrotor when two LoE in motors M_1 and M_2 occur while performing a desired trajectory emulating conduct electrical inspection. In this experiment, the efficiency in M_1 is reduced 40% at time $t \sim 28$s and, later, at time $t \sim 44$s, the efficiency in M_2 is reduced 20%.

An helical trajectory with the form $x_r(t) = t$, $y_r(t) = a\sin(t)$ and $z_r(t) = a\cos(t)$ was chosen as desired trajectory, where a indicating the radius of the circumference. The initial conditions are defined as $\xi_r(0) = [-1,\ 1,\ 2]^T$ in meters. A video of this experiment can be seen at https://youtu.be/CTBkhfcTYXc.

Figures 5.47- 5.49 show the performance of the quadcopter, while tracking a desired trajectory subject to LoE in two rotors. The behavior of the vehicle in 3D space is presented in Figure 5.47. Notice from this figure that when the first LoE (40%) in rotor M_0 was injected, the practical stability, when using $\vec{u}_R$, is compromised perturbing the drone. Moreover, this controller was not capable of assuring the stability of the system and therefore, it was not possible to apply the second LoE in rotor M_1 because the aerial robot crashed. Nevertheless, when using $\vec{u}^*$, the quadrotor can continue tracking the desired reference and even tough, compensate a second LoE (20%) in rotor M_1 at time $t \sim 44$s.

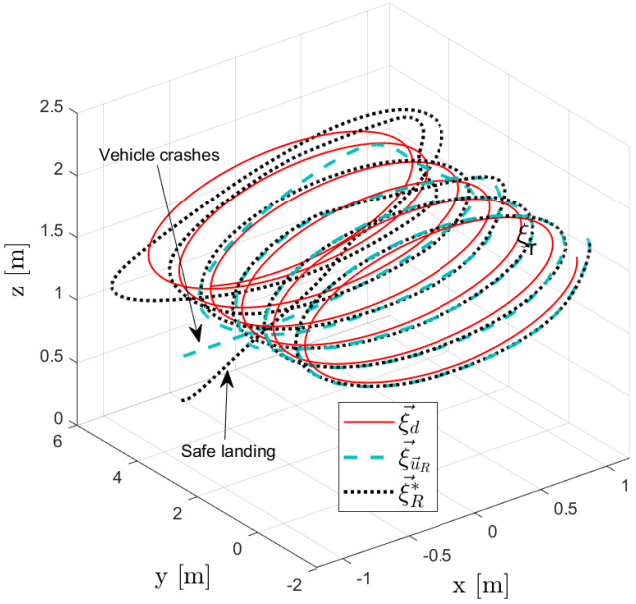


Figure 5.47 – Scenario 2.- 3D state performances in flight tests.

Figures 5.48, 5.49 introduce the behavior of the disturbance estimation and the motor control inputs. Observe from these figures that when using $\vec{u}_R^*$, the vehicle suffers a small degradation on their performance without exceeding the physical constraints of the motors and even tough, it can overcome the second LoE around $t \sim 44$s allowing to continue tracking the desired reference and landing safely.

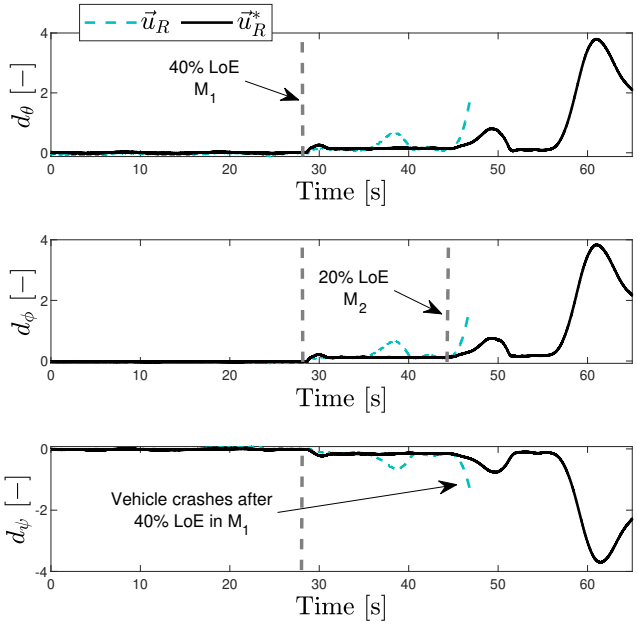


Figure 5.48 – Scenario 2.- Performance of the attitude disturbance estimation.

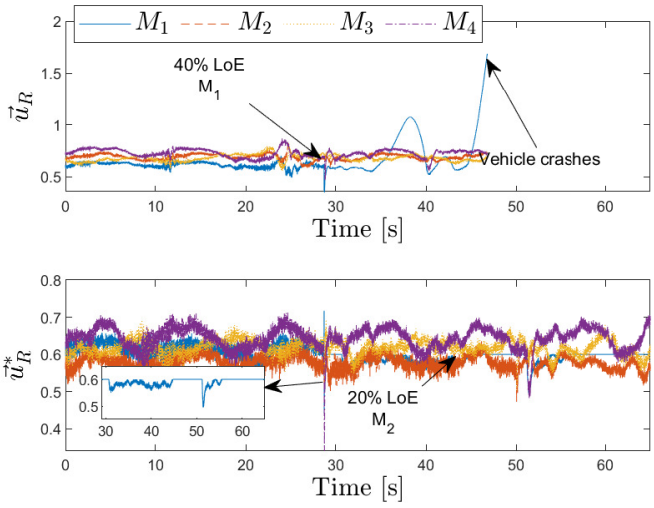


Figure 5.49 – Scenario 2.- Behavior of motor control actions.

5.5 Robust control architecture conclusions

The goal of this chapter was to introduce a robust control architecture based on disturbance estimator. The scheme is composed of a linear/nonlinear controller and an observer to give robustness to the closed-loop system with respect to nonlinear uncertainties and external disturbances. Three experiments were performed to validate the proposed architecture: facing small airflow speed, adding charge in the quadcopter and LoE in a rotor. The experimental results have shown the good performance of the architecture facing the aforementioned scenarios. Nevertheless, if wind gust or reduced LoE in rotors are applied, this can lead to the damage of actuators or make the aerial drone to become unstable.

A novel robust fault tolerant control scheme was obtained enhancing the control strategy previously discussed. The difference from the first scheme is that, this novel architecture contains a rotors fault observer to detect and estimate the loss of effectiveness in actuators. Simulations and experimental tests were carried-out for validating the performance of the proposed scheme: a rotor failure in a non hover position, a critical rotor fault with safe displacement and emergency landing and, a critical rotor fault with forced landing. The maximum value of LoE in rotors that the architecture counteracted was 30% and 60% for non hover and hover positions, respectively. This is occurs for three reasons. The compensating parameter $\vec{u}_\zeta$ is not aware of the limits of the rotors and therefore it sends a large value to counteract the fault. The other one is related with the physical constraints of the actuators and their operating velocity range. The last one and perhaps more importantly, because the architecture is not based on classical fault tolerant controls which are integrated with a reconfiguration control block. Nevertheless, the analysis of the experimental and simulations results have shown the high performance capacity on rotor faults rejection and smooth convergence with low computational cost.

Finally, a bounded robust control architecture was developed from the disturbance observer and optimal control properties. Besides estimating and compensating unknown dynamics, this scheme improves its robustness by solving the physical constraints of the motors. The resultant controller avoids damaging the actuators of the system such that, its stability is guaranteed for a class of perturbations. The proposed architecture was validated in practice in two different scenarios: applying several wind-gust while the vehicle was hovering and, LoE in two motors while performing a desired trajectory. From these experiments, a high performance and smoother convergence can be obtained when using $\vec{u}_R^*$. The aforementioned is the result of scaling the values of the compensating parameter, as well that of the control input into the motors constraints value. In other words, solving these constraints as a quadratic problem. In addition, regardless the computational limitations of the embedded system, the proposed architecture has low computational cost even when solving the quadratic problem.

Chapter 6 Part VI

6 Conclusions

In this thesis the problem of robustly tracking moving targets has been investigated. The contributions herein reported have been focused on the three objectives: i.) Introducing a unified and accessible analysis of control strategies aimed at solving the stabilization and tracking problems, ii.) exhibiting novel autonomous and semi-autonomous navigation algorithms capable of following individual targets and iii) improving disturbance rejection performance with several types of perturbations as well overcoming the implementation issues of conventional strategies.

The first goal has been achieved introducing popular controllers: nonlinear and linear backstepping and bounded state; virtual controllers based on hyperbolic functions, nonlinear backstepping based on the quaternion formalism and the finite-time convergence based on the well-known SMC. The second goal has been pursued by exploring navigation algorithms based on Model Predictive Control, vector field in cooperation with a vision algorithm based on the well-known optical flow and Kalman filter techniques and, a novel interaction Drone Exocentric Advanced Metaphor (DrEAM) relying on virtual reality properties. Finally, the last goal has been reached by exploiting the so-called uncertainty disturbance estimator, which has gained popularity over the past years. This strategy has been enhanced to deal with loss of effectiveness in rotors as properly handling rotor's constraints.

The performance of the studied control methodologies for solving the stabilization and tracking problem has been proved in Chapter 3. These results guarantee from a mathematical standpoint that the increment in complexity of the controller is paid back in terms of performance. In practice, not only the achievable performance has to be taken into account, but also the simplicity when tuning the controller. From these results, we conclude that linear backstepping controller is the simplest one to analyze and implement. This is as a consequence of assuming that the dynamic model is linear and therefore, the tuning process and implementation can be seen as a classic PD.

In Chapter 4 the performance of the explored navigation algorithms is addressed. The first result relies on the optimal trajectory generation based on the target's position.The optimal problem was solved without constraints and considering lower values of H_p for avoiding computational issues. An interesting improvement would be to consider in a new prototype for implementing the algorithm alongside the input and output constraints. In addition, using polynomial trajectories and the target model should improve the performance for reaching a target. The second one shows the resultant path of a quadrotor immersed into a visual velocity field that converges to the ideal 2D mobile robot's velocity field. For avoiding mismatching when looking for the centroid of the target, the algorithm is restricted to a maximum number of features. Moreover, the ground robot must to remain in the FoV of the airborne camera, with smooth displacements. A reasonable continuation of this work is considering the frontal camera of the quadrotor for tracking flying objects. Furthermore, considering the use of a predictive model of the target for anticipating its movements should improve the performance of the control algorithm. The last navigation algorithm result is referred to the easy maneuverability and controllability of a real drone, when a user was controlling it by means of a virtual environment. In addition, a reduction of the cognitive overload was observed when using the virtual proposed architecture. The extension of the architecture for other types of robots and the analysis of time delays in the communication could be a feasible line of research in the future of telerobotics field.

The performance improvement of a robust control architecture equipped with a disturbance observer and a nonlinear/linear controller has been introduced in Chapter 5. This is a remarkable result of this thesis in which, asymptotic trajectory tracking and rejection of disturbances with unknown dynamics was achieved, while guaranteeing a prescribed level of attenuation of unmodeled disturbances. The proposed architecture has been rigorously tested with different kind of perturbations which are related with many aerial applications. Besides to be robust towards unmodeled disturbances, this scheme is capable to handle rotors faults without a reconfiguration block as traditional FTC does it. Furthermore, high performance and smoother convergence can be obtained when using a bounded control $\vec{u}_R^*$ subject to several wind gust. Additionally, the tuning of the primary controller can be performed using conventional design techniques, while the observer is adjusted to reach a trade-off between disturbance rejection performance and robustness. This is a highly celebrated feature for control engineers because, in practice, not only the achievable performance has to be taken into account, but also the simplicity when tuning the controller. A reasonable continuation of this work is to analyze the system performance when considering outdoors experiments using a GPS as well exploring the introduction of time delays. Some primary ideas have been proposed combining disturbance observers with sequential predictor controllers but further development has to be performed.

A Publications

International journals

1. **J. Betancourt-Vera**, P. Castillo and R. Lozano, Stabilization and tracking control algorithms for VTOL aircraft: Theoretical and practical overview, Journal of Intelligent and Robotic Systems, 100, 1249–1263 (2020).

2. **J. Betancourt-Vera**, B. Wojtkowski, P. Castillo, and I. Thouvenin, Exocentric control scheme for robot applications: An immersive virtual reality approach, IEEE Transactions on Visualization and Computer Graphics, ***submitted in 2020**.

3. V. Balaguer, **J. Betancourt-Vera**, P. Castillo, P. García and R. Lozano, Robust fault tolerant architecture based on the Uncertainty and Disturbance Estimator: a case study in a quadrotor system, IEEE Transactions on Aerospace and Electronic Systems, ***submitted in 2021**

International conferences

1. **J. Betancourt-Vera**, V. Balaguer, P. Castillo, P. García and R. Lozano, Robust linear control scheme for nonlinear aerial systems: an experimental study on disturbance rejection, IEEE ITSC, September 20 - 23, 2020, Rhodes, Greece.

2. A. Sánchez-Orta, P. Castillo , F. Oliva-Palomo, **J. Betancourt-Vera**, V. Parra-Vega, L. Gallegos-Bermudez, and F. J. Ruiz-Sanchez, Aerial Following of a Non-holonomic Mobile Robot subject to Velocity Fields: A Case Study for Autonomous Vehicles Surveillance, ICUAS, September 1-4, 2020, Athens, Greece.

3. **J. Betancourt-Vera**, P. Castillo, R. Lozano, B. Vidolov, Robust control scheme for trajectory generation and tracking for quadcopters vehicles: Experimental results, ICUAS, June 12 - 15, 2018, Dallas, TX, USA.

4. **J. Betancourt-Vera**, V. Balaguer, P. Castillo, P. García and R. Lozano, Bounded robust

control scheme for quadcopter vehicles, IEEE CDC 2021, 13-15 Dec 2021, Austin Texas, ***submitted in 2021**

Scientific activities

1. C. de Souza Jr, **J. Betancourt-Vera**, P. Castillo, R. Lozano, Trajectory and Estimation for low cost drones in outdoor environment, Journeée Reégionale des Doctorants en Automatique, July 3, 2018, Amiens, France. Poster.

2. Journée Nationales sur ROS, Toulouse, 2018

3. Fête de la science, UTC, Compiègne, 2017, 2018.

Preparing for submission

1. **J. Betancourt-Vera**, P. Castillo, V. Balaguer, P. García and R. Lozano, Bounded control algorithms for quadcopter vehicles: Theoretical and practical overview in disturbance rejection.

2. **J. Betancourt-Vera**, P. Castillo, E. Ibarra Jimenez, R. Lozano, Autonomous catching ball using a quadcopter vehicle: an integral sliding mode approach

Bibliography

[1] Dreamstime. url: https://www.dreamstime.com. Accessed 20-04-2020.

[2] Edwin Vattapparamban, Ismail Guvenc, Ali I. Yurekli, Kemal Akkaya, and Selcuk Uluagac. Drones for smart cities: Issues in cybersecurity, privacy, and public safety. 2016 International Wireless Communications and Mobile Computing Conference, IWCMC 2016, pages 216–221, 2016.

[3] Andrew Zulu and Samuel John. A review of control algorithms for autonomous quadrotors. arXiv preprint arXiv:1602.02622, 2016.

[4] Jesse Stafford. How a quadcopter works| clay allen. University of Alaska, Fairbanks. Retrieved, 20, 2015.

[5] Hamid Menouar, Ismail Guvenc, Kemal Akkaya, A Selcuk Uluagac, Abdullah Kadri, and Adem Tuncer. Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. IEEE Communications Magazine, 55(3):22–28, 2017.

[6] Chad Kerr, Raed Jaradat, and Niamat Ullah Ibne Hossain. Battlefield mapping by an unmanned aerial vehicle swarm: Applied systems engineering processes and architectural considerations from system of systems. IEEE Access, 8:20892–20903, 2020.

[7] R. T. Ogan, D. Lott, and W. Paden. Electrical transmission line inspection using unmanned aircraft. In 2019 SoutheastCon, pages 1–7, 2019.

[8] S. Sudhakar, V. Vijayakumar, C. Sathiya Kumar, V. Priya, Logesh Ravi, and V. Subramaniyaswamy. Unmanned aerial vehicle (uav) based forest fire detection and monitoring for reducing false alarms in forest-fires. Computer Communications, 149:1–16, 2020.

[9] Malavika Bhaskaranand and Jerry D. Gibson. Low-complexity video encoding for uav reconnaissance and surveillance. In 2011 - MILCOM 2011 Military Communications Conference, pages 1633–1638, 2011.

[10] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward

a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. IEEE Robotics Automation Magazine, 19(3):46–56, 2012.

[11] Jianxing Liu, Chengwei Wu, Zhenhuan Wang, and Ligang Wu. Reliable filter design for sensor networks using type-2 fuzzy framework. IEEE Transactions on Industrial Informatics, 13(4):1742–1752, 2017.

[12] Patrick Doherty and Piotr Rudol. A uav search and rescue scenario with human body detection and geolocalization. In Mehmet A. Orgun and John Thornton, editors, AI 2007: Advances in Artificial Intelligence, pages 1–13, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[13] Hyon Lim and Sudipta N. Sinha. Monocular localization of a moving person onboard a quadrotor mav. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 2182–2189, 2015.

[14] Roohul Amin, Li Aijun, and Shahaboddin Shamshirband. A review of quadrotor UAV: control methodologies and performance evaluation. International Journal of Automation and Control, 10(2):87–103, 2016.

[15] Weihua Zhao and Tiauw Hiong Go. Quadcopter formation flight control combining MPC and robust feedback linearization. Journal of the Franklin Institute, 351(3):1335–1355, 2014.

[16] 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.

[17] Hossein Beikzadeh and Guangjun Liu. Trajectory tracking of quadrotor flying manipulators using l1 adaptive control. Journal of the Franklin Institute, 355(14):6239 – 6261, 2018.

[18] W. Lei, C. Li, and M. Z. Q. Chen. Robust adaptive tracking control for quadrotors by combining pi and self-tuning regulator. IEEE Transactions on Control Systems Technology, 27(6):2663–2671, Nov 2019.

[19] Stanislav Tomashevich and Andrey Belyavskyi. Passification based simple adaptive control of quadrotor. IFAC-PapersOnLine, 49(13):281 – 286, 2016. 12th IFAC Workshop on Adaptation and Learning in Control and Signal Processing ALCOSP 2016.

[20] J. Santiaguillo-Salinas, M.A. Rosaldo-Serrano, and E. Aranda-Bricaire. Observer-based time-varying backstepping control for parrot's ar.drone 2.0. IFAC-PapersOnLine, 50(1):10305 – 10310, 2017. 20th IFAC World Congress.

[21] M.A. Rosaldo-Serrano and E. Aranda-Bricaire. Trajectory tracking for a commercial quadrotor via time-varying backstepping. IFAC-PapersOnLine, 51(13):532 – 536, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[22] U. Ansari and A. H. Bajodah. Tracking control of quadrotor using generalized dynamic inversion with constant-proportional rate reaching law. In 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), pages 1–7, Feb 2019.

[23] A. Das, K. Subbarao, and F. Lewis. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. IET Control Theory Applications, 3(3):303–314, March 2009.

[24] M. Elena Antonio-Toledo, Edgar N. Sanchez, Alma Y. Alanis, J.A. Flórez, and Marco A. Perez-Cisneros. Real-time integral backstepping with sliding mode control for a quadrotor uav. IFAC-PapersOnLine, 51(13):549 – 554, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[25] J. Escareño, S. Salazar, H. Romero, and R. Lozano. Trajectory control of a quadrotor subject to 2d wind disturbances. Journal of Intelligent & Robotic Systems, 70(1):51–63, Apr 2013.

[26] Markus Hehn and Raffaello D Andrea. Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters. IEEE Transactions on Robotics, pages 1–16, 2015.

[27] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. 2017.

[28] Daniel Mellinger and Vijay Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. International Conference on Robotics and Automation, pages 2520–2525, 2011.

[29] Rui Wang and Jinkun Liu. Trajectory tracking control of a 6-dof quadrotor uav with input saturation via backstepping. Journal of the Franklin Institute, 355(7):3288 – 3309, 2018.

[30] Zhuofan Xu, Ruixuan Wei, Qirui Zhang, Kai Zhou, and Renke He. Obstacle avoidance algorithm for UAVs in unknown environment based on distributional perception and decision making. CGNCC 2016 - 2016 IEEE Chinese Guidance, Navigation and Control Conference, pages 1072–1075, 2017.

[31] Rakesh R. Warier, Amit K. Sanyal, Mani H. Dhullipalla, and Sasi Prabhakaran Viswanathan. Trajectory tracking control for underactuated thrust-propelled aerial vehicles. IFAC-PapersOnLine, 51(13):555 – 560, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[32] B. P. ICKES. A new method for performing digital control system attitude computations using quaternions. AIAA Journal, 8(1):13–17, 1970.

[33] Chutiphon Pukdeboon, Alan S. I. 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.

[34] Zheng Zhu, Yuanqing Xia, and Mengyin Fu. Adaptive sliding mode control for attitude stabilization with actuator saturation. IEEE Transactions on Industrial Electronics, 58(10):4898–4907, 2011.

[35] A. Chovancová, T. Fico, P. Hubinský, and F. Duchon. Comparison of various quaternion-based control methods applied to quadrotor with disturbance observer and position estimator. Robotics and Autonomous Systems, 79:87–98, 2016.

[36] C. Izaguirre-Espinosa, A.J. 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.

[37] 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.

[38] Yonggen Ling, Tianbo Liu, and Shaojie Shen. Aggressive quadrotor flight using dense visual-inertial fusion. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1499–1506, 2016.

[39] S.P. Bhat and D.S. Bernstein. Finite-time stability of homogeneous systems. In Proceedings of the 1997 American Control Conference (Cat. No.97CH36041), volume 4, pages 2513–2514 vol.4, 1997.

[40] Sanjay P. Bhat and Dennis S. Bernstein. Finite-time stability of continuous autonomous systems. SIAM Journal on Control and Optimization, 38(3):751–766, 2000.

[41] S. Bhat and D. Bernstein. Geometric homogeneity with applications to finite-time stability. Mathematics of Control, Signals and Systems, 17:101–127, 2005.

[42] Yiguang Hong, Yangsheng Xu, and Jie Huang. Finite-time control for robot manipulators. Systems & Control Letters, 46(4):243–253, 2002.

[43] Haibo Du and Shihua Li. Finite-time attitude stabilization for a rigid spacecraft using homogeneous method. IFAC Proceedings Volumes, 44(1):2620–2625, 2011. 18th IFAC World Congress.

[44] Haibo Du, Shihua Li, and Chunjiang Qian. Finite-time attitude tracking control of spacecraft with application to attitude synchronization. IEEE Transactions on Automatic Control, 56(11):2711–2717, 2011.

[45] Amit K. Sanyal, Jan Bohn, and Anthony M. Bloch. Almost global finite time stabilization of rigid body attitude dynamics. In 52nd IEEE Conference on Decision and Control, pages 3261–3266, 2013.

[46] Jan Bohn and Amit K. Sanyal. Almost global finite-time stabilization of rigid body attitude dynamics using rotation matrices. International Journal of Robust and Nonlinear Control, 26(9):2008–2022, 2016.

[47] Arthur E. Bryson, Yu-Chi Ho, and George M. Siouris. Applied optimal control: Optimization, estimation, and control. IEEE Transactions on Systems, Man, and Cybernetics, 9(6):366–367, 1979.

[48] Emanuel Todorov. Optimal Control Theory. Environment and Planning C Government and Policy, 4(2):1–28, 2006.

[49] P. R. Chandler and M. Pachter. Research issues in autonomous control of tactical UAVs. Proceedings of the American Control Conference, 1(June):394–398, 1998.

[50] E. Frazzoli, M.a. A Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. Proceedings of the 39th IEEE Conference on Decision and Control, 2000, 1(iii):821–826 vol.1, 2000.

[51] Steven M. LaValle. Planning Algorithms. Cambridge University Press, 2006.

[52] Li-chun Lai, Chi-ching Yang, and Chia-ju Wu. Time-Optimal Control of a Hovering Quad-Rotor Helicopter. Journal of Intelligent and Robotic Systems, pages 115–135, 2006.

[53] C. L. Castillo, W. Moreno, and K. P. Valavanis. Unmanned helicopter waypoint trajectory tracking using model predictive control. In 2007 Mediterranean Conference on Control Automation, pages 1–8, 2007.

[54] L Beji, A Abichou, and N Azouz. Modeling, Motion planning and Control of the drone with revolving Aerofoils: an Outline of the XSF Project. Robot Motion and Control, (335, pp.):165–177, 2006.

[55] Marcus Nolte, Marcel Rose, Torben Stolte, and Markus Maurer. Model predictive control based trajectory generation for autonomous vehicles — an architectural approach. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 798–805, 2017.

[56] Pengkai Ru and Kamesh Subbarao. Nonlinear Model Predictive Control for Unmanned Aerial Vehicles. MDPI aerospace, pages 1–26, 2017.

[57] Michael Neunert, Cédric de Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1398–1404, 2016.

[58] Fabio Morbidi, Roel Cano, and David Lara. Minimum-Energy Path Generation for a Quadrotor UAV. International Conference on Robotics and Automation, (May):2–8, 2016.

[59] Pedro Serra, Rita Cunha, Tarek Hamel, David Cabecinhas, and Carlos Silvestre. Landing of a quadrotor on a moving target using dynamic image-based visual servo control. IEEE Transactions on Robotics, 32(6):1524–1535, 2016.

[60] Hamed Jabbari Asl and Jungwon Yoon. Bounded-input control of the quadrotor unmanned aerial vehicle: A vision-based approach. Asian Journal of Control, 19(3):840–855, 2017.

[61] Rafik Mebarki and Bruno Siciliano. Velocity-free image-based control of unmanned aerial vehicles. In 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pages 1522–1527, 2013.

[62] Dongliang Zheng, Hesheng Wang, Weidong Chen, and Yong Wang. Planning and tracking in image space for image-based visual servoing of a quadrotor. IEEE Transactions on Industrial Electronics, 65(4):3376–3385, 2018.

[63] Dongliang Zheng, Hesheng Wang, Jingchuan Wang, Siheng Chen, Weidong Chen, and Xinwu Liang. Image-based visual servoing of a quadrotor using virtual camera approach. IEEE/ASME Transactions on Mechatronics, 22(2):972–982, 2017.

[64] Geoff Fink, Hui Xie, Alan F. Lynch, and Martin Jagersand. Nonlinear dynamic image-based visual servoing of a quadrotor. Journal of Unmanned Vehicle Systems, 3(1):1–21, 2015.

[65] H. Jabbari, G. Oriolo, and H. Bolandi. An adaptive scheme for image-based visual servoing of an underactuated uav. Int. J. Robotics Autom., 29, 2014.

[66] Francois Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. IEEE Robotics Automation Magazine, 13(4):82–90, 2006.

[67] Javier Gomez-Avila, Carlos Lopez-Franco, Alma Y. Alanis, Nancy Arana-Daniel, and Michel Lopez-Franco. Ground vehicle tracking with a quadrotor using image based visual servoing. IFAC-PapersOnLine, 51(13):344–349, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[68] Erdinc Altug, James P. Ostrowski, and Camillo J. Taylor. Control of a quadrotor helicopter using dual camera visual feedback. The International Journal of Robotics Research, 24(5):329–341, 2005.

[69] Bruno Herisse, Tarek Hamel, Robert Mahony, and Francois-Xavier Russotto. A nonlinear terrain-following controller for a vtol unmanned aerial vehicle using translational optical flow. In 2009 IEEE International Conference on Robotics and Automation, pages 3251–3257, 2009.

[70] Heera Lal Maurya, Archit Krishna Kamath, Nishchal K. Verma, and Laxmidhar Behera. Vision-based fractional order sliding mode control for autonomous vehicle tracking by a quadrotor uav. In 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–6, 2019.

[71] Fátima Oliva-Palomo, Aldo Jonathan Muñoz-Vázquez, Anand Sánchez-Orta, Vicente Parra-Vega, Carlos Izaguirre-Espinosa, and Pedro Castillo. A fractional nonlinear pi-structure control for robust attitude tracking of quadrotors. IEEE Transactions on Aerospace and Electronic Systems, 55(6):2911–2920, 2019.

[72] Justin Thomas, Jake Welde, Giuseppe Loianno, Kostas Daniilidis, and Vijay Kumar. Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor. IEEE Robotics and Automation Letters, 2(3):1762–1769, 2017.

[73] C. Huang and Tzu-Shun Hung. Visual servoing of micro aerial vehicle landing on ground platform. 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 2071–2076, 2014.

[74] Vincenzo Lippiello, Rafik Mebarki, and Fabio Ruggiero. Visual coordinated landing of a uav on a mobile robot manipulator. In 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 1–7, 2013.

[75] Shakeeb Ahmad and R. Fierro. Real-time quadrotor navigation through planning in depth space in unstructured environments*. 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1467–1476, 2019.

[76] Minjie Wan, Guohua Gu, Weixian Qian, Kan Ren, Xavier Maldague, and Qian Chen. Unmanned aerial vehicle video-based target tracking algorithm using sparse representation. IEEE Internet of Things Journal, 6(6):9689–9706, 2019.

[77] Hamid Alturbeh and James F. Whidborne. Visual flight rules-based collision avoidance systems for uav flying in civil aerospace. Robotics, 9(1), 2020.

[78] Dongxu Li and J.B. Cruz. Better cooperative control with limited look-ahead. In 2006 American Control Conference, pages 6 pp.–, 2006.

[79] A. Siddiqui, M. Verma, and David M. Tulett. A periodic planning model for maritime transportation of crude oil. EURO Journal on Transportation and Logistics, 2:307–335, 2013.

[80] Edwyn Gray. Nineteenth-century torpedoes and their inventors. Annapolis, Md.: Naval Institute Press ; London : Greenhill, 2004.

[81] Robin R. Murphy and Jennifer L. Burke. Up from the rubble: Lessons learned about hri from search and rescue. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 49(3):437–441, 2005.

[82] Mica R. Endsley. Design and evaluation for situation awareness enhancement. Proceedings of the Human Factors Society Annual Meeting, 32(2):97–101, 1988.

[83] G. Gioioso, M. Mohammadi, A. Franchi, and D. Prattichizzo. A force-based bilateral teleoperation framework for aerial robots in contact with the environment. In 2015

IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, pages 318–324, 2015.

[84] Jiayu Li, Bo You, Liang Ding, Jiazhong Xu, Weihua Li, Hannan Chen, and Haibo Gao. A novel bilateral haptic teleoperation approach for hexapod robot walking and manipulating with legs. Robotics and Autonomous Systems, 108:1 – 12, 2018.

[85] C. A. López Martínez, I. Polat, R. v. d. Molengraft, and M. Steinbuch. Robust high performance bilateral teleoperation under bounded time-varying dynamics. IEEE Transactions on Control Systems Technology, 23(1):206–218, 2015.

[86] D.D. Santiago, E. Slawinski, and V. Mut. Human-inspired stable bilateral teleoperation of mobile manipulators. ISA Transactions, 95:392 – 404, 2019.

[87] Y. Ye, Y. Pan, and T. Hilliard. Bilateral teleoperation with time-varying delay: A communication channel passification approach. IEEE/ASME Transactions on Mechatronics, 18(4):1431–1434, 2013.

[88] Seung-Ju Lee and Hyo-Sung Ahn. Controller designs for bilateral teleoperation with input saturation. Control Engineering Practice, 33:35 – 47, 2014.

[89] R. Mahony, F. Schill, P. Corke, and Y. S. Oh. A new framework for force feedback teleoperation of robotic vehicles based on optical flow. In 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, pages 1079–1085, 2009.

[90] Hala Rifaï, Minh-Duc Hua, Tarek Hamel, and Pascal Morin. Haptic-based bilateral teleoperation of underactuated unmanned aerial vehicles. Proceedings of the 18th World Congress, Milano, Italy, 44(1):13782 – 13788, 2011.

[91] Xiaolei Hou, Hua Lan, Xiaojun Xing, Yaohong Qu, Dongli Yuan, Jianguo Yan, and Panfeng Huang. Environmental force reflection in an admittance configured haptic interface for teleoperation of vtol aerial robots. 20th IFAC World Congress, Toulouse, France, 50(1):10262 – 10267, 2017.

[92] E. Slawiñski, D. Santiago, and V. Mut. Control for delayed bilateral teleoperation of a quadrotor. ISA Transactions, 71:415 – 425, 2017.

[93] R. A. S. Fernández, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina, and P. Campoy. Natural user interfaces for human-drone multi-modal interaction. In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, pages 1013–1022, 2016.

[94] J. Regenbrecht, A. Tavakkoli, and D. Loffredo. A robust and intuitive 3d interface for teleoperation of autonomous robotic agents through immersive virtual reality environments. In 2017 IEEE Symposium on 3D User Interfaces (3DUI), Los Angeles, CA, USA, pages 199–200, 2017.

[95] T. Abut and S. Soygüder. Haptic industrial robot control and bilateral teleoperation by using a virtual visual interface. In 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, pages 1–4, 2018.

[96] A. Naceri, D. Mazzanti, J. Bimbo, D. Prattichizzo, D. G. Caldwell, L. S. Mattos, and N. Deshpande. Towards a virtual reality interface for remote robotic teleoperation. In 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, Brazil, pages 284–289, 2019.

[97] S. N. F. Nahri, S. Du, and B. Van Wyk. Haptic system interface design and modelling for bilateral teleoperation systems. In 2020 International SAUPEC/RobMech/PRASA Conference, Cape Town, South Africa, South Africa, pages 1–6, 2020.

[98] João Marcelo Teixeira, Ronaldo Ferreira, Matheus Santos, and Veronica Teichrieb. Teleoperation using google glass and ar, drone for structural inspection. In XVI Symposium on Virtual and Augmented Reality, pages 28–36. IEEE, 2014.

[99] A. Suarez, P. Sanchez-Cuevas, M. Fernandez, M. Perez, G. Heredia, and A. Ollero. Lightweight and compliant long reach aerial manipulator for inspection operations. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, pages 6746–6752, 2018.

[100] John Paulin Hansen, Alexandre Alapetite, I Scott MacKenzie, and Emilie Møllenbach. The use of gaze to control drones. In ETRA '14: Proceedings of the Symposium on Eye Tracking Research and Applications, Safety Harbor, Florida, pages 27–34. ACM, 2014.

[101] Jzau-Sheng Lin and Zi-Yang Jiang. Implementing remote presence using quadcopter control by a non-invasive bci device. Computer Science and Information Technology, 3(4):122–126, 2015.

[102] N. Liu, T. Lu, Y. Cai, J. Lu, H. Gao, B. Li, and S. Wang. Design of virtual reality teleoperation system for robot complex manipulation. In 2019 Chinese Automation Congress (CAC), Hangzhou, China, China, pages 1789–1793, 2019.

[103] S. Kim, Y. Kim, J. Ha, and S. Jo. Mapping system with virtual reality for mobile robot teleoperation. In 2018 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, South Korea, pages 1541–1541, 2018.

[104] A.W.W. Yew, S.K. Ong, and A.Y.C. Nee. Immersive augmented reality environment for the teleoperation of maintenance robots. Procedia CIRP, 61:305 – 310, 2017. The 24th CIRP Conference on Life Cycle Engineering.

[105] Bryan Walter, Jared Knutzon, Adrian Sannier, and James Oliver. Virtual uav ground control station. In AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit, Chicago, Illinois, page 6320, 2004.

[106] Baichuan Huang, Deniz Bayazit, Daniel Ullman, Nakul Gopalan, and Stefanie Tellex. Flight, camera, action! using natural language and mixed reality to control a drone. In Proc. IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, 2019.

[107] Matteo Macchini, Thomas Havy, Antoine Weber, Fabrizio Schiano, and Dario Floreano. Hand-worn haptic interface for drone teleoperation. In Proc. IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020.

[108] Siyang Yang, Jiang Han, Lian Xia, and Ye-Hwa Chen. Adaptive robust servo constraint tracking control for an underactuated quadrotor uav with mismatched uncertainties. ISA Transactions, 106:12–30, 2020.

[109] Xiao-Zheng Jin, Tao He, Xiao-Ming Wu, Hai Wang, and Jing Chi. Robust adaptive neural network-based compensation control of a class of quadrotor aircrafts. Journal of the Franklin Institute, 357(17):12241–12263, 2020.

[110] Moussa Labbadi and Mohamed Cherkaoui. Robust adaptive nonsingular fast terminal sliding-mode tracking control for an uncertain quadrotor uav subjected to disturbances. ISA Transactions, 99:290 – 304, 2020.

[111] Lei Guo and Songyin Cao. Anti-disturbance control theory for systems with multiple disturbances: A survey. ISA transactions, 53(4):846–849, 2014.

[112] Shihong Ding, Wen-Hua Chen, Keqi Mei, and David J Murray-Smith. Disturbance observer design for nonlinear systems represented by input–output models. IEEE Transactions on Industrial Electronics, 67(2):1222–1232, 2019.

[113] Jingqing Han. From PID to active disturbance rejection control. IEEE transactions on Industrial Electronics, 56(3):900–906, 2009.

[114] Shihua Li, Jun Yang, Wen-Hua Chen, and Xisong Chen. Generalized extended state observer based control for systems with mismatched uncertainties. IEEE Transactions on Industrial Electronics, 59(12):4792–4802, 2011.

[115] Vicente Balaguer, Ricardo Sanz, Pedro Garcia, and Pedro Albertos. Two-degree-of-freedom PID tuning based on an uncertainty and disturbance estimator. In 2018 7th International Conference on Systems and Control (ICSC), pages 424–429. IEEE, 2018.

[116] Antonio González, Angel Cuenca, Vicente Balaguer, and Pedro García. Event-triggered predictor-based control with gain-scheduling and extended state observer for networked control systems. Information Sciences, 491:90–108, 2019.

[117] Antonio Gonzalez, Vicente Balaguer, Pedro Garcia, and Angel Cuenca. Gain-scheduled predictive extended state observer for time-varying delays systems with mismatched disturbances. ISA transactions, 84:206–213, 2019.

[118] B. Kurkçu, C. Kasnakoglu, and M. Ö. Efe. Disturbance/uncertainty estimator based robust control of nonminimum phase systems. IEEE/ASME Transactions on Mechatronics, 23(4):1941–1951, 2018.

[119] Jiuqiang Deng, Xi Zhou, and Yao Mao. On vibration rejection of nonminimum-phase long-distance laser pointing system with compensatory disturbance observer. Mechatronics, 74:102490, 2021.

[120] Alireza Mohammadi, Mahdi Tavakoli, Horacio J Marquez, and Farzad Hashemzadeh. Nonlinear disturbance observer design for robotic manipulators. Control Engineering Practice, 21(3):253–267, 2013.

[121] Jialu Du, Xin Hu, Miroslav Krstić, and Yuqing Sun. Robust dynamic positioning of ships with disturbances under input saturation. Automatica, 73:207–214, 2016.

[122] Divyesh Ginoya, Chandrashekhar M Gutte, PD Shendge, and SB Phadke. State-and-disturbance-observer-based sliding mode control of magnetic levitation systems. Transactions of the Institute of Measurement and Control, 38(6):751–763, 2016.

[123] Qi Lu, Beibei Ren, Siva Parameswaran, and Qing-Chang Zhong. Uncertainty and disturbance estimator-based robust trajectory tracking control for a quadrotor in a global positioning system-denied environment. Journal of Dynamic Systems, Measurement, and Control, 140(3), 2018.

[124] Longhao Qian and Hugh HT Liu. Path-following control of a quadrotor uav with a cable-suspended payload under wind disturbances. IEEE Transactions on Industrial Electronics, 67(3):2021–2029, 2019.

[125] Qi Lu, Beibei Ren, and Siva Parameswaran. Uncertainty and disturbance estimator-based global trajectory tracking control for a quadrotor. IEEE/ASME Transactions on Mechatronics, 2020.

[126] Shen Yin, Bing Xiao, Steven X Ding, and Donghua Zhou. A review on recent development of spacecraft attitude fault tolerant control system. IEEE Transactions on Industrial Electronics, 63(5):3311–3320, 2016.

[127] G. Ortiz-Torres, P. Castillo, F. D. J. Sorcia-Vázquez, J. Y. Rumbo-Morales, J. A. Brizuela-Mendoza, J. De La Cruz-Soto, and M. Martínez-García. Fault estimation and fault tolerant control strategies applied to vtol aerial vehicles with soft and aggressive actuator faults. IEEE Access, 8:10649–10661, 2020.

[128] Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. Annual reviews in control, 32(2):229–252, 2008.

[129] Iman Sadeghzadeh, Ankit Mehta, Abbas Chamseddine, and Youmin Zhang. Active fault tolerant control of a quadrotor uav based on gainscheduled pid control. In 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pages 1–4. IEEE, 2012.

[130] Abdel-Razzak Merheb, Hassan Noura, and François Bateman. Passive fault tolerant control of quadrotor uav using regular and cascaded sliding mode control. In 2013 Conference on Control and Fault-Tolerant Systems (SysTol), pages 330–335. IEEE, 2013.

[131] Luis Gracia, J Ernesto Solanes, Pau Muñoz-Benavent, Jaime Valls Miro, Carlos Perez-Vidal, and Josep Tornero. Adaptive sliding mode control for robotic surface treatment using force feedback. Mechatronics, 52:102–118, 2018.

[132] Pin Lyu, Jizhou Lai, Jianye Liu, Hugh HT Liu, and Qingrui Zhang. A thrust model aided fault diagnosis method for the altitude estimation of a quadrotor. IEEE Transactions on Aerospace and Electronic Systems, 54(2):1008–1019, 2017.

[133] Abdel-Razzak Merheb and Hassan Noura. Active fault-tolerant control of quadrotor uavs based on passive controller bank. In Rany Rizk and Mariette Awad, editors, Mechanism, Machine, Robotics and Mechatronics Sciences, pages 231–241. Springer International Publishing, 2019.

[134] Xiaohong Nian, Weiqiang Chen, Xiaoyan Chu, and Zhiwei Xu. Robust adaptive fault estimation and fault tolerant control for quadrotor attitude systems. International Journal of Control, 93(3):725–737, 2020.

[135] R. C. Avram, X. Zhang, and J. Muse. Quadrotor actuator fault diagnosis and accommodation using nonlinear adaptive estimators. IEEE Transactions on Control Systems Technology, 25(6):2219–2226, 2017.

[136] B.R. Andrievsky, N.V. Kuznetsov, G.A. Leonov, and A.Yu. Pogromsky. Hidden oscillations in aircraft flight control system with input saturation. IFAC Proceedings Volumes, 46(12):75 – 79, 2013. 5th IFAC Workshop on Periodic Control Systems.

[137] R. van den Berg, A. Pogromsky, and K. Rooda. Convergent systems design: Anti-windup for marginally stable plants. In Proceedings of the 45th IEEE Conference on Decision and Control, pages 5441–5446, 2006.

[138] B. R. Andrievsky, N. V. Kuznetsov, G. A. Leonov, and A. Y. Pogromsky. Convergence based anti-windup design method and its application to flight control. In 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems, pages 212–218, 2012.

[139] Gennady Leonov, Boris Andrievsky, Nikolay Kuznetsov, and A. Pogromskii. Aircraft control with anti-windup compensation. Differential Equations, 48:1700–1720, 2012.

[140] Matthias Faessler, Davide Falanga, and Davide Scaramuzza. Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight. IEEE Robotics and Automation Letters, 2(2):476–482, 2017.

[141] E.J.J. Smeur, G.C.H.E. de Croon, and Q. Chu. Cascaded incremental nonlinear dynamic inversion for mav disturbance rejection. Control Engineering Practice, 73:79–90, 2018.

[142] Paul Pounds, Robert Mahony, and Joel Gresham. Towards dynamicallyfavourable quadrotor aerial robots. In In Proc. of Australasian Conference on Robotics and Automation, 2004.

[143] Raymond W. Prouty. Helicopter performance, stability and control. Krieger Publishing Company, 1995.

[144] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. Control Engineering Practice, 18(7):691–699, 2010. Special Issue on Aerial Robotics.

[145] Herbert Goldstein. Classical Mechanics. Addison-Wesley, 1980.

[146] P. Castillo, R. Lozano, and A. Dzul. Stabilization of a mini rotorcraft with four rotors. IEEE Control Systems Magazine, 25(6):45–55, 2005.

[147] Pedro Castillo Garcia, R. Lozano, and Alejandro Dzul. Modelling and Control of Mini-Flying Machines. 01 2005.

[148] Richard M. Howard. Dynamics of flight: Stability and control; third edition. Journal of Guidance, Control, and Dynamics, 20(4):839–840, 1997.

[149] R. Lozano. Unmanned Aerial Vehicles Embedded Control. John Wiley-ISTE Ltd, 2010.

[150] H. Abaunza, P. Castillo, and R. Lozano. Quaternion Modeling and Control Approaches, pages 1–29. Springer International Publishing, 2018.

[151] D. C. Gandolfo, L. R. Salinas, A. Brandão, and J. M. Toibero. Stable path-following control for a quadrotor helicopter considering energy consumption. IEEE Transactions on Control Systems Technology, 25(4):1423–1430, July 2017.

[152] J.F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare. Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. Control Engineering Practice, 19(8):790 – 797, 2011.

[153] Alejandro J Malo Tamayo, Cesar A. Villaseñor Ríos, Juan Manuel Ibarra Zannatha, and Santos M. Orozco Soto. Quadrotor input-output linearization and cascade control. IFAC-PapersOnLine", 51(13):437 – 442, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[154] Andrew R. Teel. Global stabilization and restricted tracking for multiple integrators with bounded controls. Systems & Control Letters, 18(3):165 – 171, 1992.

[155] Guillaume Sanahuja, Pedro Christian Ayala Castillo, and Anand Sanchez. Stabilization of n integrators in cascade with bounded input with experimental application to a vtol laboratory system. International Journal of Robust and Nonlinear Control, 2009.

[156] A. Ailon. Simple tracking controllers for autonomous vtol aircraft with bounded inputs. IEEE Transactions on Automatic Control, 55(3):737–743, March 2010.

[157] Trong-Toan Tran, Ge Shuzhi Sam, and He Wei. Adaptive control of a quadrotor aerial vehicle with input constraints and uncertain parameters. International Journal of Control, 2018.

[158] H K Khalil. Nonlinear systems (3rd ed.). Upper Saddle River, NJ: Prentice Hall, 2002.

[159] Antonios Zagaris, Hans G. Kaper, and Tasso J. Kaper. Fast and Slow Dynamics for the Computational Singular Perturbation Method. arXiv Mathematics e-prints, page math/0401206, Jan 2004.

[160] Heudiasyc Lab. FL-AIR framework. accessed: Feb. 27, 2020. [online]., 2012, https://devel.hds.utc.fr/software/flair.

[161] Shtessel Yuri, Leonid Fridman Christopher, Edwards, and Levant Arie. Sliding Mode Control and Observation. Birkhäuser Basel, 2014.

[162] Taeyoung Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). 49th IEEE Conference on Decision and Control (CDC), 2010., pages 5420–5425, 2010.

[163] Zhong Liu, Yuqing He, Liying Yang, and Jianda Han. Control techniques of tilt rotor unmanned aerial vehicle systems: A review. Chinese Journal of Aeronautics, 30(December):135–148, 2016.

[164] Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. IFAC Proceedings Volumes, 47(3):11773–11780, 2014. 19th IFAC World Congress.

[165] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 15–22, 2014.

[166] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments, pages 649–666. 04 2016.

[167] Andreas Bircher, Mina Samir Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon path planning for 3d exploration and surface inspection. Autonomous Robots, 42, 02 2018.

[168] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. Journal of Field Robotics, 33(4):431–450, 2016.

[169] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. Journal of Field Robotics, 29(5):832–841, 2012.

[170] JM M Maciejowski. Predictive control: with constraints. Pearson Education Limited, Harlow, UK, page 331, 2002.

[171] A. Chakrabarty, R. Morris, X. Bouyssounouse, and R. Hunt. Autonomous indoor object tracking with the parrot ar.drone. In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pages 25–30, 2016.

[172] Jianbo Shi and Tomasi. Good features to track. In 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.

[173] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. Intel Corporation, Microprocessor Research Labs, 2000.

[174] Pedro Castillo, Alejandro Dzul, and Rogelio Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. IEEE Transactions on Control Systems Technology, 12(4):510–516, 2004.

[175] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE Robotics Automation Magazine, 19(3):20–32, 2012.

[176] Samir Bouabdallah and Roland Siegwart. Full control of a quadrotor. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 153–158, 2007.

[177] L. Luque-Vega, B. Castillo-Toledo, and Alexander G. Loukianov. Robust block second order sliding mode control for a quadrotor. Journal of the Franklin Institute, 349(2):719 – 739, 2012.

[178] Bo Zheng and Yisheng Zhong. Robust attitude regulation of a 3-dof helicopter benchmark: Theory and experiments. IEEE Transactions on Industrial Electronics, 58(2):660–670, 2011.

[179] Shafiqul Islam, Peter X. Liu, and Abdulmotaleb El Saddik. Robust control of four-rotor unmanned aerial vehicle with disturbance uncertainty. IEEE Transactions on Industrial Electronics, 62(3):1563–1571, 2015.

[180] Bo Zhao, Bin Xian, Yao Zhang, and Xu Zhang. Nonlinear robust adaptive tracking control of a quadrotor uav via immersion and invariance methodology. IEEE Transactions on Industrial Electronics, 62(5):2891–2902, 2015.

[181] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. AIAA Guidance, Navigation and Control Conference and Exhibit, page p.82, 2007.

[182] Steven Waslander and Carlos Wang. Wind disturbance estimation and rejection for quadrotor position control. AIAA Infotech@Aerospace Conference, 2009.

[183] Xiwang Dong, Bocheng Yu, Zongying Shi, and Yisheng Zhong. Time-Varying Formation Control for Unmanned Aerial Vehicles: Theories and Applications. IEEE Transactions on Control Systems Technology, pages 1–1, 2014.

[184] Kooksun Lee, J. Back, and I. Choy. Nonlinear disturbance observer based robust attitude tracking controller for quadrotor uavs. International Journal of Control, Automation and Systems, 12:1266–1275, 2014.

[185] Ricardo Sanz, Pedro Garcia, Qing-Chang Zhong, and Pedro Albertos. Robust control of quadrotors based on an uncertainty and disturbance estimator. Journal of Dynamic Systems, Measurement, and Control, 138(7):71006, 2016.

[186] Qing-Chang Zhong and David Rees. Control of uncertain LTI systems based on an uncertainty and disturbance estimator. Journal of Dynamic Systems, Measurement, and Control(Transactions of the ASME), 126(4):905–910, 2004.

[187] Kamal Youcef-Toumi and Osamu Ito. A time delay controller for systems with unknown dynamics. Journal of dynamic systems, measurement, and control, 112(1):133–142, 1990.

[188] Beibei Ren, Q.C. Zhong, and Jinhao C. Robust control for a class of nonaffine nonlinear systems based on the uncertainty and disturbance estimator. IEEE Transactions on Industrial Electronics, 62(9):5881–5888, 2015.

[189] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. Mathematical Programming, 27:1–33, 1983.

[190] Luca Di Gaspero and Angelo Furfaro. uQuadProg ppen source library. accessed: January. 13, 2021. [online]., 2006, https://github.com/fx74/uQuadProg.