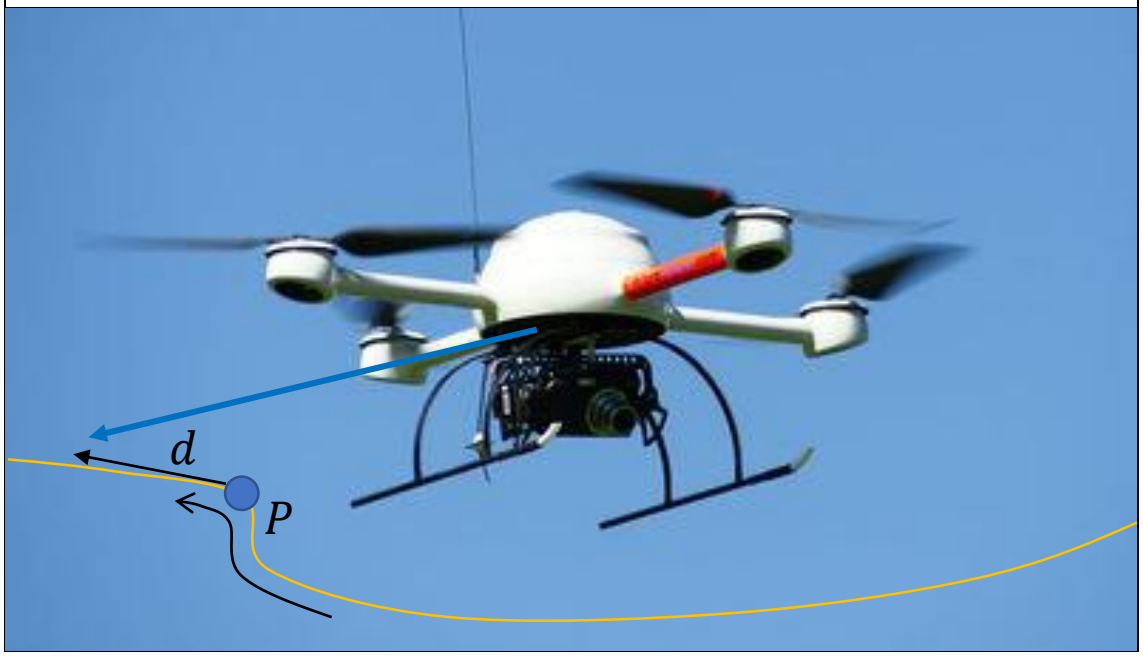
Universidad Nacional de Educación a Distancia
Universidad Complutense de Madrid

Trabajo Fin de Máster

Máster en Ingeniería de Sistemas y Control

Design and implementation of a path-following and a ship-tracking algorithms for quadcopters to monitor spills on water



Autor: Pablo García Auñón

Directores: Jesus Manuel de la Cruz García
Matilde Santos Peñas

Madrid, junio de 2015

Abstract

In this master thesis and within the *SALACOM* project, two algorithms to guide a quadcopter have been developed: a path-following and a target-tracking algorithms. Given a trajectory in the tri-dimensional space, the first algorithm will control the velocity direction of the UAV to accurately fly along the path; the kinematics of the movement is analyzed and the parameters tuned to improve the performance.

The second algorithm is in charge of tracking a ship and guiding the quadcopter to it in an efficient way. Simulation results have proved the efficiency of these proposals.

Resumen

Este trabajo fin de máster se enmarca dentro del proyecto nacional SALACOM: Sistema Autónomo para la Localización y Actuación ante Contaminantes en el Mar, DPI 2013-46665-C1, del Grupo ISCAR (UCM) y del Dpto. Informática y Automática (UNED). Se ha pretendido realizar alguna contribución que pudiera resultar de utilidad para el desarrollo de este proyecto de investigación. En el trabajo se han desarrollado e implementado dos algoritmos para guiar un UAV (vehículo aéreo no tripulado), en concreto un cuatrirotor. El primero algoritmo realiza un seguimiento de trayectoria (path-following). El segundo es un algoritmos de tracking, donde interviene el tiempo.

En el primer caso se define una trayectoria en el espacio tridimensional y el algoritmo controla la dirección de la velocidad del UAV para seguirlo de la forma más precisa posible. Se ha analizado la cinemática del movimiento y se han sintonizado algunos parámetros del algoritmo para mejorar su rendimiento. El segundo algoritmo se encarga de que el UAV siga a otro vehículo en movimiento, en este caso un barco. Los resultados son satisfactorios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation of this project

While this master thesis was being finished, the liberian-flag oil tanker *Lady M*, with 90.000 Tm of fuel, suffered a fired in its engine room, 200 km away from shore of the Canary Islands. Alarm bells rang. Only two months before, a fishing vessel with 1.400 Tm of fuel sank close to the same shore because another fire. That time, some spills reached the coast, although luckily it was not an important incident. All the Spaniards have in their memories the *Prestige* disaster, an accident of an oil tanker that resulted in a 53.000 Tm oil-spill on 2.000 km of the Portuguese, Spanish and French coasts. This was one the worst natural disasters in the European history.

These are some examples of how dangerous the hydrocarbons can be and they show the necessity of being prepared in case of an oil tanker, an oil offshore platform or any other floating system has a problem and a spill might occur. Many other small and big accidents on the seas take place all around the world, some of which result in spills on open waters. Note that most of the natural resources are presently transported by vessels and, therefore, the likelihood of a disaster is not that remote.

If the preventive methods fail and there is a spill, a contention system must be used to avoid the fuel to reach the coast, where it can provoke an irreparable damage. These systems consist of a group of vessels, which deploy a barrier to content the spill, while other ships extract the fuel from the surface of the water (the fuel normally floats) and put it in a safe place. In swallow waters, close to the coast for instance, the draft of the ships becomes a problem and lighter ships might need to be used.

It is in these cases where unmanned vehicles became interesting due to their low weight and operational costs, and their use in scenarios where a fast actuation is crucial. Having as well aerial systems to support these operations, by providing visual information from a high point of view, is normally also necessary. For this duty, the use of standard helicopters or airplanes is normally slower, less operative and way more expensive than unmanned aircraft.

In the last years, there have been many studies about the possibility of fighting against spills by using unmanned systems. A complete report of the means of detecting contaminants with the help of remote sensing systems, and

the advantage of using UAVs, can be found in [32]. In [34], a vision system was boarded in a fixed-wing UAV and was able to identify a spill (simulated with yellow popcorn deploy on the water) in front of the Portuguese coast, track it and transmit the information. More in general, in [33], it is claimed the use of UAVs for coastal and environmental research. Another approach was taken in [2], where the spill was delimited and monitored by a swarm of small quadcopters.

With the aim of developing a complete unmanned system to monitor and remove spills on open waters, the project *Sistema Autónomo para la Localización y Actuación ante Contiminantes en el Mar*[1] (SALACOM) was proposed by the *Spanish Ministry of Industry* and is being developed by the *Universidad Complutense de Madrid* (UCM) and *Universidad Nacional de Educación a Distancia* (UNED), between January 2014 and December 2016. The global objective of this project is the development of a fast-response system, composed of *Unmanned Surface Vehicles* (*USVs*) and *Unmanned Aerial Vehicles* (UAVs), capable of deploying a protecting barrier along a specific zone or of dragging a contention net with the most adequate configuration to remove the spill. The UAVs, transported by the USVs to the zone close to the spill, are in charge of locating the contaminant and to transmit the information to the USVs for their intervention. Moreover, the possibility of using a submarine vehicle, to support the localization of the contaminant spills, will be studied.

The global objectives of the system are more specifically detailed in the next points:

1. Movement of the USVs to draft the containing barrier: experimentation, modeling and control.
2. UAVs searching and coordination algorithms.
3. Perception system: definition, methods and techniques of recognizing textures and objects to detect spills and to navigate.
4. Control center, planning and monitoring of the autonomous vehicles.
5. Failures detection and recovering. Actuation protocols.
6. Coordination with a submarine vehicle.

As already said, the aerial system will provide visual information about the location and movement of the spills to the control center, and more specifically to the USVs. Since they will need to take-off and land on reduced areas, vertical take-off aircraft are mandatory to be used.

Basically, there exists two different rotorcraft which might be used in the situations to be faced in this project: helicopters and quadcopters. The first ones have the advantage that they can be propelled by an alternative engine, which means that the endurance, range and payload are normally higher. Their drawbacks are that the mechanical system to control them is heavy and expensive to maintain, and that they are unstable and difficult to control. Quadcopters, on the other hand, are normally propelled by 4 electric motors, which ease the

[1] *Autonomous System for the Localization and Intervention against Contaminants on the Sea*. Project Reference: DPI2013-46665-C2-1-R.

control considerably, since their speed is controlled by the frequency of their inputs. Also, they are more stable. However, since their power source is a battery, their range, endurance and payload is lower than the helicopter's ones. Having studied the advantages and disadvantages, quadcopters were selected for this project.

The quadcopters will be boarded on the USVs and will receive orders from the control center. They will take-off from the ships and travel along the aerial space, searching for the spills and monitoring them. Is this control center, the one in charge of ordering where the quacopters will have to fly and in general, of organizing their movement.

1.2 Objectives

This master thesis will be devoted to develop a specific control section of the aerial system. In the aerial system, to fulfill their duties within this project, we can distinguish three different levels of control:

- **Upper level**: this level of control will be in charge of generating the paths to be followed by the quadcopters to search and monitor the spills. They are planning and coordination algorithms and, in general, they organize the movement of the quadcopters, creating the paths to be followed or ordering them to return to the ships. This level of control will be implemented in a centralized control system (control center), and will command all the quadcopters of the system.

- **Intermediate level**: once a path has been generated, or the quadcopter has been commanded to return to the ship, these algorithms are in charge of guiding them. They generate as output the velocity vector and the altitude that the quadcopter must have.

- **Low level**: as input, this controller has the velocity vector and the altitude that the quadcopter is required to achieve. It actuates then on the electrical motors, stabilizing the dynamics of the system.

In this master thesis, the intermediate level of control will be developed, that is, the needed algorithms to guide the quadcoper along a given path (the so called *path-following algorithms*). Also, another algorithm to guide the UAV to return to the ship (normally called in the literature *target-tracking algorithm*) will be studied. Therefore, it will be considered that the upper and low level controllers are available and will not be studied here. In particular, the next points are the objectives of this master thesis:

1. Analyze the state of the art of path-following algorithms.

2. Propose an algorithm to follow a given path. Analyze this algorithm and propose further analysis and changes to improve the performance.

3. Analyze the state of the art of path- and target-tracking algorithms.

4. Propose an algorithm to track and converge to the ship, knowing only its present position and velocity direction. Landing on the ship will not be considered, only the control to approach it.

5. Implement different working modes, depending on the information sent from the upper level controller.

6. Implementation of the algorithms in *Matlab-Simulink*. Simulate them with different situations. Test the robustness of the algorithms.

It must be said that, since the low level controller manages the quadcopter's horizontal and vertical position independently, this work will focus on the horizontal movement. The altitude will be controlled as well, but no great variations during the mission are expected and it will be the low level controller the one in charge of controlling it.

1.3 Structure of this work

In Chapter 2, the quadcopter under study will be briefly described, with its hardware, low level controller, sensors and physical limitations. Also, an estimation of the time constant of the yaw movement will be proposed, since it will be used in the path-following and ship-tracking algorithms.

In Chapter 3, the path-following algorithm will be proposed, analyzed and tuned; improvements of the performance (in particular, laws to set the parameters) and working modes are presented.

In Chapter 4, the ship-tracking algorithm will be presented, tuned and compared with an existing one; improvements are studied as well.

Finally, in Chapter 5, the conclusions of this master thesis will be exposed and the possible future works presented.

Chapter 2

Description of the model

2.1 Hardware description

The quadcopter to be used is the *Quanser Qball-X4* (Figure 2.1), manufactured by *Quanser Innovative and Educate*, Ontario, Canada. This rotary wing platform has been designed for indoor educational and research purposes, and its open-architecture design allows an easy implementation of new control laws. The model and controls are developed in *Simulink* on the host computer and downloaded and compiled into executables on the target.

The Qball-X4 is propelled by four motors that move 10-inches propellers and it is enclosed by a protective cage to avoid the quadcopter to be damaged when trying new controllers. This cage might be removed for outdoor operations. The main components of the quadcopter are:

- Qball-X4 airframe: crossbeam structure which holds the rest of components and the protective cage.
- Motors and propellers: the quadcopter is propelled by four E-Flite Park 400 motors, fitted with paired counter-rotating APC 10x4.7 propellers.
- HiQ DAQ: data acquisition card. Reads the signals from the sensors and outputs the motor's commands. The HiQ may have an additional daughter board, which contains extra I/O, such as a sonar or a GPS.
- Gumstix computer: target computer, based on Linux with the QuaRc software installed. Gumstix DuoVero Zephyr with integrated 802.11 b/g/n WiFi, ARM Cortex-A9, 1 GHz, 1 GB DDR SDRAM.
- Sensors: the quadcopter has an IMU (*Inertial Measurement Unit*), a ultrasonic height sensor and a GPS.
- Batteries: Two 3-cell, 2500 mAh Lithium-Polymer batteries, providing up to 10 min of flight time.

In Table 2.1 the main characteristics of the quadcopter have been presented. As it has been already said, the Qball-X4 was designed to be used indoor and therefore, its use in open environments could be problematic. Note that, since its maximum velocity is 3 m/s, a wind velocity higher than this value, which

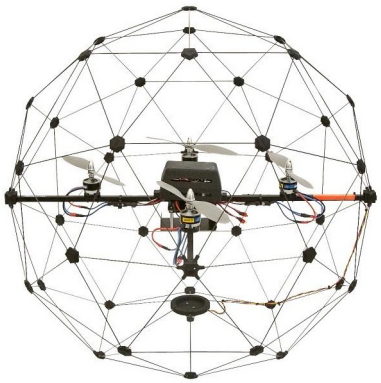


Figure 2.1: Qball-X4

.

is considered low, would avoid the quadcopter to fly in the upstream direction. Moreover, its endurance is only 10 minutes, an insufficient time for a spill searching and monitoring mission. However, it can be used as a small scale demonstrator to test the here studied algorithms. In the real system, another quadcopter will be used, more robust against winds and gusts, capable of transporting higher payloads and with a higher endurance and range.

A maximum yaw rate of the velocity has been defined, $r_{\max} = 0.5$ rad/s. However, when the quadcopter hovers, this limitation does not exist (the quadcopter is able to tilt from one direction to another, which would imply an irregularity in the direction function of the velocity, which jumps from one value to another). This limitation does exist indeed when the quadcopter is flying in one direction; it is in those cases, when this $r_{\max}$ will be considered. It is also important to remark that the here called yaw rate, and also along this work, is referred to the horizontal angle change in the direction of the velocity, not to the yaw movement of the quadcopter itself.

2.2 Modelization

The dynamics of the actuators can be modeled by a first order system:

$$F = v \cdot u = K \frac{\omega_m}{s + \omega_m} \cdot u \tag{2.1}$$

where v is the state variable, u is the PWM input to the actuator and ω_m is the bandwidth.

The dynamics of the quadcopter are modeled as usual; the complete process of modeling and equation simplification can be consulted, for instance, in [37] and [39]. The Qball-X4 user manual provides directly the simplified equations used to model the quadcopter [12].

Parameter	Value
m	1.40 kg
d	0.20 m
K	120 N
I_x	0.03 kg·m^2
I_y	0.03 kg·m^2
I_z	0.03 kg·m^2
ω_m	15 rad/s
K_y	4 N·m
g	9.8 m/s^2
$r_{\max}$	0.5 rad/s
$V_{\max}$	3 m/s
$\theta_{\max}$, $\phi_{\max}$	$\frac{15\pi}{180}$ rad
Endurance	10 min

Table 2.1: Characteristics of the quadcopter model.

The pitch and roll equations are:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{K \cdot d}{I_x} & 0 \\ 0 & 0 & -\omega_m & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_m \\ 0 \end{bmatrix} u_\theta \tag{2.2}$$

where θ is the pitch angle and $u_\theta = u_3 - u_1$ is the difference between the traction force of propellers 3 and 1. Note that the variable α is the integral of the angle θ. The dynamics of the roll angle, ϕ, is exactly the same but u_θ must be substitute by $u_\phi = u_4 - u_2$. The vertical movement is ruled by the following equation:

$$\begin{bmatrix} \dot{Z} \\ \ddot{Z} \\ \dot{v} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4K}{m} & 0 \\ 0 & 0 & -\omega_m & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z \\ \dot{Z} \\ v \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_m \\ 0 \end{bmatrix} u_z + \begin{bmatrix} 0 \\ -g \\ 0 \\ 0 \end{bmatrix} \tag{2.3}$$

being $u_z = u_1 + u_2 + u_3 + u_4$ the total traction of the four propellers. The movement in the horizontal direction is described by the equations:

$$\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{v} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4K}{m}\theta & 0 \\ 0 & 0 & -\omega_m & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ v \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_m \\ 0 \end{bmatrix} u_x \tag{2.4}$$

$$\begin{bmatrix} \dot{Y} \\ \ddot{Y} \\ \dot{v} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{4K}{m}\phi & 0 \\ 0 & 0 & -\omega_m & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y \\ \dot{Y} \\ v \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_m \\ 0 \end{bmatrix} u_y \tag{2.5}$$

And finally, the yaw equation is:

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_y}{I_z} \end{bmatrix} u_\psi \tag{2.6}$$

where $u_\psi = u_1 + u_2 - u_3 - u_4$. In Table 2.1 the value of all the parameters of the quadcopter have been presented. Beside the parameters and constants used in the model, also a maximum velocity yaw rate, $r_{\max}$, and a maximum velocity modulus, $V_{\max}$, have been also considered.

The above described modelization is the one proposed by the manufacturer of the aircraft and the controller design is based on it. However, since *Simulink* provides a 6dof body block, it will be rather used. The quadcopter will be then considered as a rigid body, able to rotate and move in the three axes, whose inputs are the forces and moments in the body frame system and its outputs are the position and velocity in the inertial frame, the Euler angles and the angular velocity in the body frame.

2.3 Sensors

The algorithm needs information about the position and attitude of the quadcopter. Firstly, the position in the inertial frame will be provided. The position will be assumed to be measured by a GPS, and its signal filtered. The position in the model will be directly extracted from the 6dof block and passed to the controller. Note that, this way, it will be assumed that the position can be perfectly measured without any error. Although it is clearly not possible in the real world, if there is a specific system able to extrapolate the position so the the signal is continuous, this approximation is acceptable.

In case of the velocity, Euler angles and angular velocity, the transfer function will be:

$$H(s) = \frac{V_s}{V_e} = \frac{\omega_s^2}{s^2 + 2\xi_s\omega_s s + 1} \tag{2.7}$$

where:

$$\omega_s = 40 \cdot 2 \cdot \pi \text{ rad/s}$$
$$\xi_s = 0.7$$

The sampling time is $T_s = 0.01$ s and a white noise is present, with a power of:

$$Npw_v = \left(\frac{0.1}{3}\right)^2$$
$$Npw_a = \left(\frac{\pi}{180}\right)^2$$
$$Npw_w = \left(\frac{3\pi}{5 \cdot 180}\right)^2$$

for the velocity, Euler angles and angular velocity signals, respectively.

2.4 Controller model

2.4.1 Pitch and roll control

The pitch and roll angles are controlled by a LQR controller with state feedback. The control law is:

$$u_\theta = -\mathbf{k}_\theta \cdot \mathbf{X}_\theta \tag{2.10}$$

being $\mathbf{X}_\theta$ the state variable of the pitch control. There is no need of deriving the variable θ, since it can be directly measured by the gyroscope. The proportional vector is chosen as follows:

$$\mathbf{k}_\theta = [0.063, 0.0175, 0.9669, 0.0183] \tag{2.11}$$

2.4.2 Vertical position control

In case of the vertical position, a PID controller will be used, being the control law as follows:

$$u_z = u_{z_0} + k_{P_z} \cdot e_z + k_{D_z} \cdot \dot{e}_z + k_{I_z} \cdot \int e_z \cdot dt \tag{2.12}$$

where e_z is the error in the vertical positioning, u_{z_0} is the take-off throttle and k_{P_z}, k_{D_z} and k_{I_z} are the coefficients of the PID controller:

$$\begin{aligned} u_{z_0} &= 0.022 \\ k_{P_z} &= 0.013 \\ k_{D_z} &= 0.009 \\ k_{I_z} &= 0.010 \end{aligned}$$

The derivative is calculated with the next following transfer function, to filter and eliminate high frequencies in the signal:

$$D(s) = \frac{30s}{s + 30} \tag{2.14}$$

Also, the value of $\dot{e}_z$ is limited to 0.25 m/s and e_z to 0.5 m.

2.4.3 Horizontal position control

An LQR controller will be also used here, with:

$$\mathbf{k}_x = [0.366, 0.382, 0.439, 0.045] \tag{2.15}$$

The derivative will be calculated as follows:

$$D(s) = \frac{25s}{s + 25} \tag{2.16}$$

2.4.4 Yaw control

For the yaw control a LQR controller is implemented too, with:

$$\mathbf{k}_\psi = [0.032, 0.015] \tag{2.17}$$

There is no need of deriving any signal, since the yaw rate can be directly measured.

2.5 Estimation of the system time constant

As it will be seen in the kinematics analysis of the path-following and target-tracking algorithm, Sections 3.5 and 4.5, an estimation of the time constants of the system is needed. This constant, normally denoted in the literature as τ, gives an idea of how fast the system response is. In our case, we are interested in knowing how fast the quadcopter is able to change its heading, i.e. its velocity direction. Normally, one can propose a theoretical model of the system and analyze it, obtaining its time constant. However, since in this case there are several elements involved (physics of the quadcopter, response of the electrical engines, sensors and the low level controller), it is tough to get a final analytic expression of it. Another approach will be then used. The simulated system response will be directly analyzed when a change in the heading is performed. This response can then be compared to a low order linear system to estimate the time constant.

The transfer function of a first order system is the following, see [21]:

$$H(s) = \frac{K}{1 + \tau_l \cdot s} e^{-s\tau_0} \tag{2.18}$$

where K is the gain of the response, τ_l is the time constant of the 1^{st} order system and τ_0 is the time delay. The system under consideration does not show any time delay, so we can consider $\tau_0 = 0$. We will test the system with different changes in the yaw angle and, to determine the gain K, only the 10 seconds after the step input will be considered. On the other hand, the time constant is the time needed to reach the 63% of the final value, i.e. $0.63 \cdot K \cdot \psi_c$, where ψ_c is the input value.

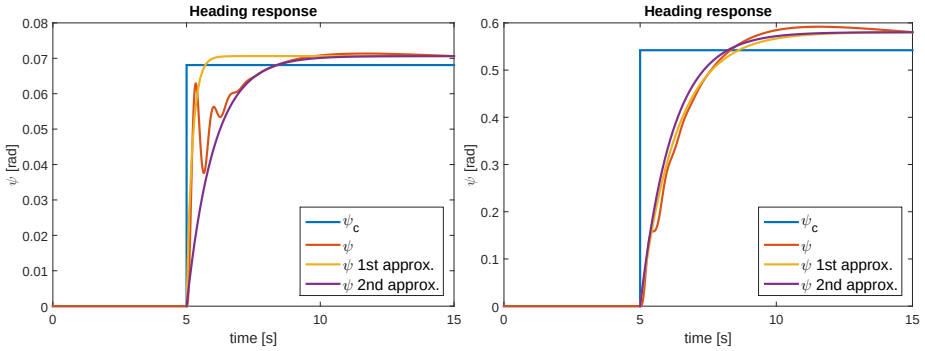
For low values of ψ_c, the system response shows the so called *kickback* behavior, i.e. a backward movement when trying to reach the commanded value, due to the nonlinearities of the system. This leads to estimate the system with Equation 2.18, faster than it actually is. Therefore, a 2^{nd} approximation will be also used; instead of considering the 63% of the final value, the time constant will be also calculated considering the 95% of the final value, being then:

$$\tau_l = \frac{t_{95\%}}{\log(0.05)} \tag{2.19}$$

This way the kickback behavior will be omitted. If one of the oscillations reaches the 95% of the final value, before being the response stable, it will not be considered.

The system is simulated with step changes in its heading ψ, from 0.02 to $\pi/5$ rad. In Figure 2.2 the response of the system has been represented for two different headings, $\psi_c = 0.068$ and 0.5 rad. Both 1^{st} order linear approximations have been shown, both calculated as above explained.

Analyzing the real response of the system, one can see that when the heading change ψ_c is low, the response turns out to be irregular, presenting the kickback behavior. This behavior, which is clearly visible up to $\psi_c = 0.4$ rad, can be seen in Figure 2.2a (red line). In Figure 2.2b the response for $\psi_c = 0.5$ is shown; in that case, there is no relevant kickback behavior. Note that the gain is higher than 1 and that the higher the input is, the more similar both approximations to the real system are. In any case, one might expect that the time constant of the system is between the ones of the two linear approximations.



(a) Low value heading change. (b) High value heading change.

Figure 2.2: System response when a heading change is commanded. Comparison with both 1st order linear system approximations.

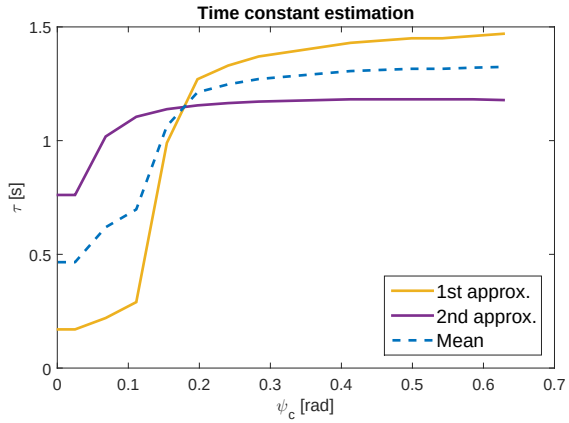


Figure 2.3: Time constant estimated with a 1st and a 2nd order system. The dashed line is the average value between them.

.

The time constant has been calculated for both linear approximations for a wide range of values of ψ_c and represented in Figure 2.3. As it can be seen, for values of the heading command lower than 0.2 rad the time constant tends to sink in the case of the 1st approximation, in yelow. The time constant estimated by the 2nd approximation, in purple, is practically constant until 0.1 rad and decreases a 20% approximately for lower values. With dashed line, in blue, the average value between both approximation has been represented.

Since we will study afterwards the behavior of the system around the equilibrium point, where $\psi \simeq 0$ is expected, we will assume the time constant of the heading response as the average value close to this equilibrium point:

$$\tau_l \simeq 0.45 \text{ s} \tag{2.20}$$

being τ_l the low lever controller time constant.

Chapter 3

Path-following algorithm

3.1 Introduction

One of the most basic mission that a Unmanned Aerial Vehicle (UAV) has to fulfill is to be able to accurately follow a given path. One can distinguish between two concepts:

- Path-following: To follow a given geometric trajectory independently from the variable time.
- Path-tracking: To follow a given trajectory and imposing a determined position depending on time.

In path-tracking, it is not only important to travel along the path, but also to converge to a determined position on it, which is externally given and depends on time, i.e. a temporal constraint is present. In this chapter, a path-following type algorithm will be presented, since such a temporal constraint is not required in the mission.

Most of the path-following algorithms are implemented in a complete separate outer loop from the inner loop controller, which is in charge of stabilizing the quadcopter's dynamics. If this is the case, a sufficient difference in the time response between them must be ensured; typically the inner loop must be ∼2-4 times faster that the outer loop [29]. This is the most common situation, the path-following algorithm controls the kinematics and generates the commands to the inner loop controller (usually commercial autopilots).

The path-following algorithms might be grouped into geometrical methods and based on control techniques. Many of them were developed to be used with fixed-wing aircraft, although they are adaptable to rotorcraft. Geometrical methods try to chase a virtual point, which moves along the path, with a guide law mainly based on the Dublin's car model [11]. Control techniques applied to path-following are normally more robust against external perturbations; PID, non-linear and LQR controllers have been widely used for this purpose. There exist several bidimensional methods, five of which were chosen in [38], compared and the results analyzed:

- Carrot-chasing algorithm: a virtual point, located on the path and at a distance d from projection of the quadcopter to the path, is chased, i.e.

the quadcopter is headed to it. It is simple to be implemented but it is not robust.

- Non-linear guidance law (NLGL, [30] and [31]): the virtual point is located on the path at a distance R from the quadcopter and chased, as in the Carrot-chasing algorithm. Usable for different types of paths.

- Pure pursuit ([13]) and line-of-sight-based (LOS, [15] and [35]): is a combination of a pure pursuit term, which leads the quadcopter to the next way-point of the path, and a LOS term, which tries to line it up with the path between the two corresponding way-points. It is simple and intuitive to implement but sensitive to the gains in it and cannot be used for high wind velocities. An example of this technique can be found in [20].

- Vector-field-based laws (VFL, [24] and [26]): a vector field is created which steers the quadcopter to the desired trajectory. The convergence is ensured by a Lyapunov function. Small cross-track error but difficult to tune and there might be chattering.

- Linear Quadratic Regulator (LQR, see for instance [27]): calculated the optimum control effort to minimize the cross-track error. The control efforts are lower but the cross-track error is higher than with other methods.

These techniques are tested following linear and circular paths, with constant winds and random gusts, covering a wide combination of sets of values of the constants in the algorithms. Monte Carlo simulations were carried out and the results evaluated measuring the perpendicular distance between the reference path and the actual quadcopter trajectory (the so called *cross-track* error) and the control efforts. The VFL and the NLGL gave the best results.

Recently, in [14], a further development of the methods presented in [30] and [31] has been carried out, combining them with other methods. Basically, a virtual tracking point is used, located at a certain distance from the unmanned vehicle, but instead of keeping that distance fixed, the velocity of the points is also set. Although there are two parameters in the algorithm, one of them can be chosen by a relationship between other variables, making the tuning easier.

There are many other algorithms developed specifically for quadcopters, but mainly based on algorithms previously proposed to control other types of unmanned systems. In [19], for instance, a simple linear controller was used to follow a path generated by way-points. For a high maneuverability quadcopter, a more complex predictive control is presented in [18], keeping the yaw angle fixed and being the trajectories recalculated online. A similar technique was also used in [23]. In [37], Lyapunov functions were applied, firstly to control the dynamics of the quadcopter, and then to track a given path. An improvement on path following strategy is described in [10], where a robust compensator was added to a classical PD controller. Intelligent control techniques ([36]) have also provided useful tools to improve the results of guiding autonomous systems.

In this master thesis, an adaptation of the control law presented in [41] is used, which was originally designed for fixed-wing aircraft and the algorithm was able to track the path as well. In that work, time and space are decoupled; the path is followed without considering the time and then the velocity in set independently to reach the points along the path at the required moments.

Although here it is not considered, with this algorithm, we will be able in the future to include a path-tracking ability if the mission requires it. This algorithm is modern, robust, easy to implement and configurable.

The quadcopter is steered to chase a point located at a certain distance d from a virtual point P, in the tangential trajectory of the path. What makes this method different from a carrot-chasing one, is the law that describes the movement of that virtual point; instead of simply computing it as the projection of the quadcopter to the path, the point P has its own kinematics, which depends on its relative position from the quadcopter and the velocity of this last one. The global convergence is ensured by Lyapunov arguments. Note that this method has certain relationship with [14], in the sense that the virtual point, which leads the yaw commands, has its own kinematics, similar in both methods. However, in the algorithm here applied, the quadcopter converges directly to the virtual point, whereas in [14] it converges to a point at a certain distance L_1.

The algorithm presented here not only gives good results following given trajectories, as it will be shown, but it can also be adapted to better follow different paths. Before carrying out tests on the real system, simulations of the algorithm must be performed.

3.2 Reference frames

The system under consideration is a real quadcopter with an embedded low level controller, whose inputs and outputs are the following:

Inputs

- Desired altitude.
- Velocity.
- Direction of the velocity in the xy plane of the inertial frame, $\mathcal{I}$.

Outputs

- Global position in the earth frame, $\mathcal{E}$.
- Altitude.
- Velocity in the $\mathcal{E}$ frame.
- Orientation in the $\mathcal{E}$ frame, i.e. the attitude of the quadcopter.
- Roll, pitch and yaw angular velocities.

It is also assumed that the controller is able to stabilize the quadcopter when the magnitude and direction of the reference speed is given. Independently of the flight direction, the controller is also able to take the aerial vehicle to a given altitude.

The next reference frames will be used:

Earth frame $\{\mathcal{E}\}$

- Origin O: point on the earth surface.
- $\mathbf{e_1}_{\mathcal{E}}$: pointing to the North direction.

- $\mathbf{e_{2}}_{\mathcal{E}}$: pointing to the East direction.
- $\mathbf{e_{3}}_{\mathcal{E}}$: pointing down, right-hand rule.

Inertial frame $\{I\}$

- Origin O: point on the earth surface, same as in $\{\mathcal{E}\}$.
- $\mathbf{e_{1}}_{I}$: pointing to the East direction.
- $\mathbf{e_{2}}_{I}$: pointing to the North direction.
- $\mathbf{e_{3}}_{I}$: pointing up, right-hand rule.

Expressing the basis vectors in the $\{\mathcal{E}\}$ frame, the rotation matrix from frame $\{\mathcal{E}\}$ to $\{I\}$ is:

$$\mathbf{R}_{\mathbf{E}}^{\mathbf{I}} = \begin{bmatrix} \mathbf{e_{1}}_{I} \\ \mathbf{e_{2}}_{I} \\ \mathbf{e_{3}}_{I} \end{bmatrix}_{\{\mathcal{E}\}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.1}$$

Parallel transport frame $\{\mathcal{F}\}$

- Origin P: virtual point on the path.
- $\mathbf{t}$: tangent to the path at P.
- $\mathbf{n_1}$: first normal vector.
- $\mathbf{n_2}$: second normal vector.

The relationship between the basis vectors of the Bishop and the Frenet reference frames [4] is:

$$\begin{bmatrix} \mathbf{t} \\ \mathbf{n} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{n_1} \\ \mathbf{n_2} \end{bmatrix} \tag{3.2}$$

where

$$\frac{d\beta(l)}{dl} = \lambda(l) \tag{3.3a}$$
$$k_1 = \rho\cos\beta \tag{3.3b}$$
$$k_2 = \rho\sin\beta \tag{3.3c}$$

being ρ the curvature of the path, l the length, λ its torsion, and k_1 and k_2 the Bishop curvatures, which will be used afterwards. The transformation matrix can be calculated as follows:

$$\mathbf{R}_{\mathbf{I}}^{\mathbf{F}} = \begin{bmatrix} \mathbf{t} \\ \mathbf{n_1} \\ \mathbf{n_2} \end{bmatrix}_{\{I\}} \tag{3.4}$$

if the basis vectors are row vectors expressed in the $\{I\}$ frame.

Body frame $\{\mathcal{B}\}$

- Origin Q: center of gravity of the quadcopter.
- $\mathbf{e_{1_\mathcal{B}}}$: pointing to the front direction, parallel to the plane which contains the propellers.
- $\mathbf{e_{2_\mathcal{B}}}$: perpendicular to $\mathbf{e_{1_\mathcal{B}}}$ and parallel to the plane which contains the propellers.
- $\mathbf{e_{3_\mathcal{B}}}$: pointing up, right-hand rule.

To calculate the rotation matrix, the Euler angles (ϕ, θ and ψ) will be used:

$$\mathbf{R_E^B} = \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi & \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi - \mathrm{c}\phi\mathrm{s}\psi & \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\psi \\ \mathrm{c}\theta\mathrm{s}\psi & \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi + \mathrm{c}\phi\mathrm{c}\psi & \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\psi \\ -\mathrm{s}\theta & \mathrm{s}\phi\mathrm{c}\theta & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix} \tag{3.5}$$

Velocity frame $\{\mathcal{W}\}$

- Origin Q: center of gravity of the quadcopter.
- $\mathbf{w_1}$: pointing to the velocity direction.
- $\mathbf{w_2}$: right-hand rule.
- $\mathbf{w_3}$: perpendicular to $\mathbf{w_1}$, in the plane determined by $\mathbf{e_{1_\mathcal{B}}}$ and $\mathbf{e_{3_\mathcal{B}}}$.

These basis vectors will be then calculated as follows:

$$\mathbf{w_3} = \frac{\mathbf{w_1} \wedge \mathbf{e_{2_\mathcal{B}}}}{|\mathbf{w_1} \wedge \mathbf{e_{2_\mathcal{B}}}|}$$

$$\mathbf{w_2} = \mathbf{w_3} \wedge \mathbf{w_1}$$

Auxiliary frame $\{\mathcal{D}\}$

- Origin Q: center of gravity of the quadcopter.
- $\mathbf{b_{1D}}$: pointing d meters ahead of P in the $\mathbf{t}$ direction.
- $\mathbf{b_{2D}}$: perpendicular to $\mathbf{b_1}$ in the plane determined by $\mathbf{t}$ and $\mathbf{n_1}$.
- $\mathbf{b_{3D}}$: right-hand rule.

These basis vectors will be then calculated as follows:

$$\mathbf{b_{1D}} = \frac{d \cdot \mathbf{t} - y_F \cdot \mathbf{n_1} - z_F \cdot \mathbf{n_2}}{\sqrt{d^2 + y_F^2 + z_F^2}} \tag{3.7a}$$

$$\mathbf{b_{2D}} = \frac{y_F \cdot \mathbf{t} + d \cdot \mathbf{n_1}}{\sqrt{y_F^2 + d^2}} \tag{3.7b}$$

$$\mathbf{b_{3D}} = \mathbf{b_1} \wedge \mathbf{b_2} \tag{3.7c}$$

where d is a constant parameter (a distance), and y_F and z_F are the coordinates of the point Q in the $\mathbf{n_1}$ and $\mathbf{n_2}$ plane, $\{\mathcal{F}\}$ frame.

In Figure 3.1 the frames $\{\mathcal{F}\}$, $\{\mathcal{W}\}$ and $\{\mathcal{D}\}$ have been represented. Recall that P is the virtual point to be followed and Q is the center of gravity of the quadcopter. The other frames have not been represented.

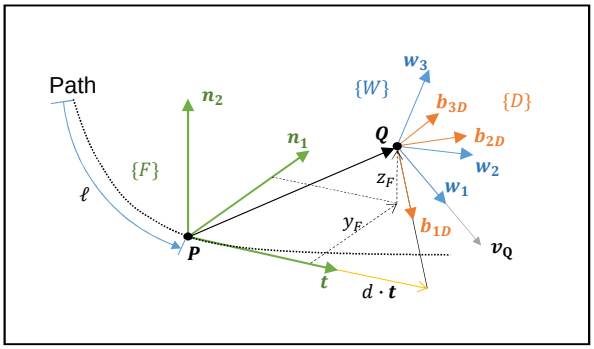


Figure 3.1: Reference frames used.

.

3.3 Control algorithm

The high level control algorithm will calculate the direction of the velocity so that the quadcopter is led to the virtual point P and the vector $\mathbf{w_1}$ (frame $\{\mathcal{W}\}$) is made coincident with $\mathbf{b_{1D}}$ (frame $\{\mathcal{D}\}$). The magnitude of the velocity will be constant and the altitude will be controlled by the own quadcopter controller.

The position of the center of gravity of the quadcopter Q in the frame $\{\mathcal{F}\}$ and its derivative are expressed as:

$$\mathbf{r}_Q|_{\{\mathcal{F}\}} = \begin{bmatrix} x_F \\ y_F \\ z_F \end{bmatrix} \tag{3.8}$$

$$\dot{\mathbf{r}}_Q|_{\{\mathcal{F}\}} = \begin{bmatrix} \dot{x}_F \\ \dot{y}_F \\ \dot{z}_F \end{bmatrix} = -\dot{l} \cdot \mathbf{t} - \mathbf{w_{F/I}} \wedge \mathbf{r}_Q + v_Q \cdot \mathbf{R_W^F} \cdot \mathbf{w_1} \tag{3.9}$$

where

$$\mathbf{w_{F/I}}\big|_{\{\mathcal{F}\}} = [0, -k_2(l) \cdot \dot{l}, k_1(l) \cdot \dot{l}] \tag{3.10}$$

is the angular velocity of the frame $\{\mathcal{F}\}$ with respect to $\{I\}$ expressed in $\{\mathcal{F}\}$, v_Q is the magnitude of the absolute velocity of the quadcopter and $\mathbf{R_W^F}$ is the rotation matrix from frame $\{\mathcal{W}\}$ to $\{\mathcal{F}\}$. The velocity of the virtual point P along the path is chosen taking into account the following expression:

$$\dot{l} = \left(v_Q \cdot \mathbf{w_1} + k_l \cdot \mathbf{r_Q}|_{\{\mathcal{F}\}}\right) \cdot \mathbf{t} \tag{3.11}$$

where k_l is the coefficient which controls the convergence of the virtual point P, proportionally to the distance between this point and the quadcopter (vector $\mathbf{r_Q}|_{\{\mathbf{F}\}}$). With Equations 3.9 and 3.11 one can finally get the dynamics of the point Q with respect to frame $\{\mathcal{F}\}$. Note that the algorithm's aim is to lead the vector $\mathbf{r}_Q|_{\{\mathcal{F}\}}$ to zero and to align $\mathbf{w_1}$ with $\mathbf{b_{1D}}$, which will tend to be coincident with tangent vector $\mathbf{t}$. An error function Ψ will be defined so:

$$\Psi(\hat{R}) = \frac{1}{2}\text{tr}\left[(\mathbf{I_3} - \mathbf{\Pi_R^T \Pi_R})(\mathbf{I_3} - \hat{\mathbf{R}})\right] = \frac{1}{2}(1 - \hat{R}_{11}) \tag{3.12}$$

which is a positive-defined function and accounts the misalignment between the velocity direction $\mathbf{w_1}$ and the basis vector $\mathbf{b_{1D}}$. In the previous equation $\hat{\mathbf{R}} = \mathbf{R}^{\mathbf{D}}_{\mathbf{W}}$ and $\mathbf{\Pi_R} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, and by deriving it we finally get:

$$\dot{\Psi}(\hat{R}) = \mathbf{e}_{\hat{\mathbf{R}}} \cdot \left(\begin{bmatrix} q \\ r \end{bmatrix} - \mathbf{\Pi_R} \hat{\mathbf{R}}^T \left(\mathbf{R}^{\mathbf{D}}_{\mathbf{F}} \cdot \mathbf{w_{F/I}}\big|_{\mathcal{F}} + \mathbf{w_{D/F}}\big|_{\mathcal{D}} \right) \right) \tag{3.13a}$$

$$\mathbf{e}_{\hat{\mathbf{R}}} = \frac{1}{2} \mathbf{\Pi_R} \left((\mathbf{I_3} - \mathbf{\Pi}^{\mathbf{T}}_{\mathbf{R}} \mathbf{\Pi_R}) \hat{\mathbf{R}} - \hat{\mathbf{R}}^T (\mathbf{I_3} - \mathbf{\Pi}^{\mathbf{T}}_{\mathbf{R}} \mathbf{\Pi_R}) \right)^{\mathrm{v}}$$

$$= \frac{1}{2} \left[\hat{R}_{13}, -\hat{R}_{12} \right]^T \tag{3.13b}$$

$$\left(\mathbf{w_{D/F}}\big|_{\mathcal{D}} \right)^{\mathrm{v}} = \mathbf{R}^{\mathbf{D}}_{\mathbf{F}} \dot{\mathbf{R}}^{\mathbf{F}}_{\mathbf{D}} \tag{3.13c}$$

where q and r are the pitch and yaw velocities and $(\cdot)^v$ denotes *vee map* [28]. Equations 3.9, 3.11 and 3.13 define the overall path-following kinematic error dynamics. In order to ensure that the error function converges to zero, its derivative must be negative-defined and therefore, the next control law is proposed:

$$\begin{bmatrix} q_c \\ r_c \end{bmatrix}_{\{\mathcal{W}\}} = \Pi_R \hat{\mathbf{R}}^T \left(\mathbf{R}^{\mathbf{D}}_{\mathbf{F}} \cdot \mathbf{w_{F/I}}\big|_{\mathcal{F}} + \mathbf{w_{D/F}}\big|_{\mathcal{D}} \right) - 2 \cdot k_{\hat{R}} \cdot \mathbf{e}_{\hat{\mathbf{R}}} \tag{3.14}$$

being $k_{\hat{R}}$ a positive convergence constant. The derivative of the error function is finally $\dot{\Psi}(\hat{R}) = -2 \cdot k_{\hat{R}} \cdot {\mathbf{e}_{\hat{\mathbf{R}}}}^2$, which is negative-defined, ensuring the convergence of the system. The above pitch and yaw commands are expressed in the $\{\mathcal{W}\}$ frame and must be transformed to the body frame since the low level controller in the quadcopter has as input the yaw rate in that frame:

$$\begin{bmatrix} 0 \\ q_c \\ r_c \end{bmatrix}_{\{\mathcal{B}\}} = \mathbf{R}^{\mathbf{B}}_{\mathbf{I}} \cdot \mathbf{R}^{\mathbf{I}}_{\mathbf{W}} \cdot \begin{bmatrix} 0 \\ q_c \\ r_c \end{bmatrix}_{\{\mathcal{W}\}} \tag{3.15}$$

It must be recalled that the commanded yaw rate, r_c, is actually the yaw rate of the velocity direction, not the yaw rate of the quadcopter itself. The quadcopter yaw will be kept equal to 0, and if because of payload or of aerodynamic reasons, it must be controlled, a future work will consider it.

With the previous algorithm the paths can be followed perfectly but, once the quadcopter reaches the final position, it continues flying in the same direction. Recall that this algorithm was developed for fixed-wing UAVs, which need air speed to produce lift. This is not the case of a rotary-wing UAV and therefore it is convenient to change the velocity law so that it stops once the end of the path is reached:

$$v_{Q_c} = v_{Q_N} \cdot \left(1 - e^{a_v (\tilde{l} - 1)} \right) \tag{3.16}$$

Here v_{Q_c} is the commanded velocity magnitude, v_{Q_N} is the nominal velocity to be reached while following the path, a_v is a parameter with a high value (~ 100) and $\tilde{l} = l/l_{max}$ is the adimensional length along the path, being its maximum value 1.

3.3.1 Implementation details

In this subsection, details about how the control algorithm has been implemented will be given.

Firstly, a higher level controller calculates off-line the path to be followed by the quadcopter, as well as the Bishop reference frame, length along the path and curvatures. The path is then sent to the quadcopter.

The on-line controller has two main tasks. On the one hand, the virtual point P is calculated integrating Equation 3.11, given its previous position, the desired velocity, the actual velocity direction and the tangent vector at that point[1]. The desired altitude to be reached in every moment is simply the altitude of the point P and therefore it is passed to the low level controller.

Once the point P, the frame $\{\mathcal{F}\}$ and the angular velocity $\mathbf{w_{F/I}}\big|_{\mathcal{F}}$ from Equation 3.10 have been determined, the other main task of the algorithm takes place. All rotation matrices, the error function $\mathbf{e_{\hat{R}}}$ and the rotation velocity $\mathbf{w_{D/F}}\big|_{\mathcal{D}}$ are calculated and introduced in Equation 3.14 to finally get q_c and r_c. Note that in order to calculate $\mathbf{w_{D/F}}\big|_{\mathcal{D}}$, see Equation (3.13), the time derivative of the rotation matrix from frame $\{\mathcal{D}\}$ to $\{\mathcal{F}\}$ must be calculated and it will be obtained by a simple step approximation:

$$\left.\frac{d\mathbf{R_D^F}}{dt}\right|_{t=t_i} = \frac{\mathbf{R_D^F}\big|_{t=t_i} - \mathbf{R_D^F}\big|_{t=t_{i-1}}}{t_i - t_{i-1}} \tag{3.17}$$

Once the pitch and yaw commands have been calculated from Equation 3.14, these must be transformed to the body frame $\{\mathcal{B}\}$, since the low lever controller needs the desired yaw angle in that frame:

$$r_c|_{\{\mathcal{B}\}} = [0,0,1] \cdot \mathbf{R_I^B} \cdot \mathbf{R_W^I} \cdot \begin{bmatrix} 0 \\ q_c \\ r_c \end{bmatrix}_{\{\mathcal{W}\}} \tag{3.18}$$

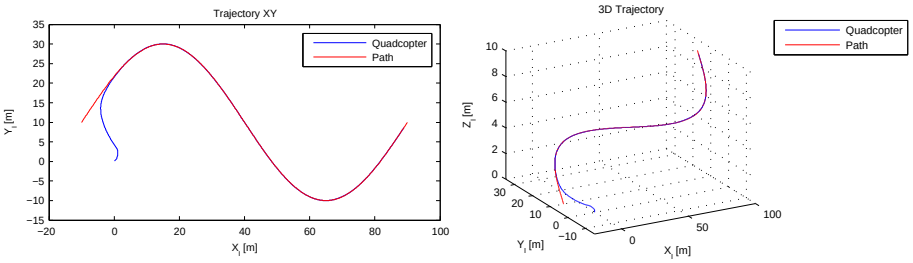
As an example of the result of the trajectory using the above explained algorithm, a path showed in Figure 3.2 has been tested. It starts at point [0,10] and has a sinusoidal shape in the $X_{\{I\}}$ and $Y_{\{I\}}$ plane, while it is linear ascending in the $Z_{\{I\}}$ direction. In Figure 3.2a the trajectory in the $XY|_{\{I\}}$ plane has been represented. The quadcopter starts at position [0,0], converges to the path quickly and does not leave it during the whole trajectory. In Figure 3.2b the same trajectory in the space has been shown and here it is possible to see that the quadcopter follows the change of altitude as well.

In Figure 3.3 an example of the signal r_c generated from Equation 3.18 has been represented, which has been limited to $r_{c_{max}} = \pm 0.5$ rad/s, according to the real physical constraints of the UAV. Note that noise from the sensors is present.

3.4 Parameters Determination by Fuzzy Logic

As it has been shown, there are three parameters involved in the algorithm: d (Equation 3.7), k_l (Equation 3.11) and $k_{\hat{R}}$ (Equation 3.14). Those parameters

[1]Despite the vector $\mathbf{r_Q}\big|_{\{\mathcal{F}\}}$ is here expressed in the reference frame $\{\mathcal{F}\}$, it is actually easier to calculate $\dot{l}$ in the $\{I\}$ frame.



(a) Path and actual trajectory of the quadcopter in the $XY|_{\{I\}}$ plane.

(b) Path and actual trajectory of the quadcopter in 3D.

Figure 3.2: Example of trajectory with the path-following algorithm.

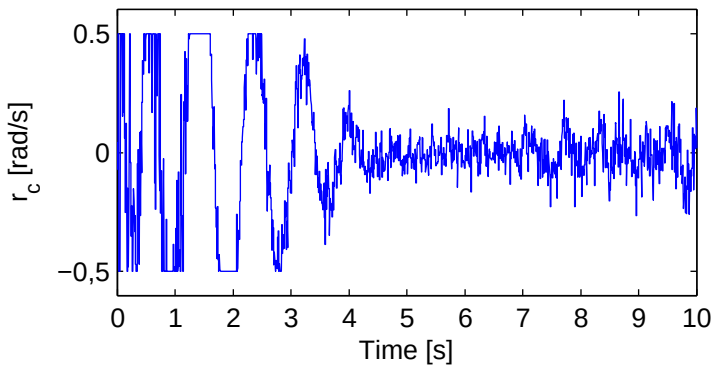


Figure 3.3: Example of the control signal generated by the algorithm.

.

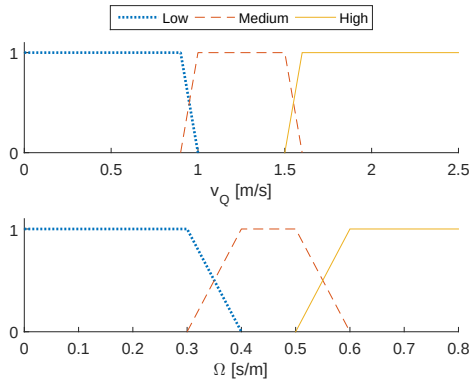


Figure 3.4: Membership functions of the input variables.

.

have an important influence on the trajectory, mainly depending on two factors: the curvature of the path and the velocity of the quadcopter. If the desired velocity and the curvature of the path are too high, the quadcopter will not be able to follow it due to the limitation of the yaw rate.

In the original work [41] these parameters were chosen as constant, although after a fast exploration of the results showed that they have a high impact on them. Initially and as a first approximation to improve the performance of the algorithm, a fuzzy logic was proposed. This solution showed to fulfill the expectation and was easy and fast to be implemented. As a result of this investigation, a paper [17] was proposed and accepted for the *10th International Conference on Soft Computing Models in Industrial and Environmental Applications* and will be published in June 2015.

Afterwards, a deep analysis of the kinematics was carried out, Section 3.5, and a general law to set the parameters obtained. The results applying that law turned out to be better and and it was the one to be used in the final algorithm.

For the fuzzy logic, two variables will be used, the velocity of the quadcopter, v, and Ω, whose definition is:

$$\Omega = \frac{\bar{\tau}}{q_{max}} \tag{3.19}$$

where q_{max} is the maximum yaw rate (constant) and $\bar{\tau}$ is the mean curvature of the reference path. Note that $1/\Omega$ is the maximum velocity with which the quadcopter could follow the trajectory. In Figure 3.4 the membership functions for both variables have been represented.

A dimensionless variable, κ, is proposed to obtain the parameters of the algorithm. Its value is defined by the fuzzy rules described in Table 3.1, based on the fuzzy variables v and Ω. They have been defined using expert knowledge. For instance, low values of κ imply that the path can be followed easily; and the higher κ is, the more difficult following the trajectory is, due to the yaw rate limitation.

For each of the three fuzzy values of κ, a set of values of the algorithm parameters are defined. The knowledge regarding how these parameters influence

Table 3.1: Fuzzy rules to estimate κ, based on the fuzzy variables v and Ω.

		v		
		low	**medium**	**high**
Ω	**low**	low	low	medium
	medium	medium	medium	high
	high	medium	high	high

Table 3.2: Set of values chosen for each value of κ.

$\boldsymbol{\kappa}$	**d**	$\mathbf{k_l}$	$\mathbf{k_{\hat{R}}}$
high	5	0.05	2
medium	3	0.15	4
low	1	0.50	6

the trajectories can be summarized as follows:

- For medium values of the distance d, if $k_{\hat{R}}$ is low, results give medium errors, independently of κ.
- For high values of d, better results are achieved by decreasing $k_{\hat{R}}$.
- For low values of κ, d must be low.
- The higher κ is, the lower k_l and the higher d must be.
- For high values of κ, $k_{\hat{R}}$ must decrease
- For low values of κ, $k_{\hat{R}}$ must increase.
- Despite the fact that k_l has smaller influence on the results than the other two parameters, high values of this parameter provide better results when κ is low.

Having said that, the set of values, depending on κ, have been chosen, see Table 3.2. To calculate the final values of the parameters, the weighted average of the fuzzy variables will be used. For instance, if $v = 1.3$ m/s and $\Omega = 0.55$ s/m, we have:

$$\kappa = 0.5 \cdot medium + 0.5 \cdot high$$

The values of the parameter are finally calculated as follows in this example:

$$d = 0.5 \cdot 3 + 0.5 \cdot 5 = 4$$
$$k_l = 0.5 \cdot 0.15 + 0.5 \cdot 0.05 = 0.10$$
$$k_{\hat{R}} = 0.5 \cdot 6 + 0.5 \cdot 2 = 4$$

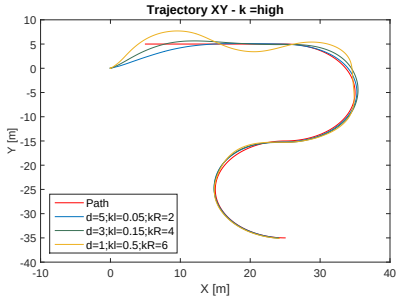


Figure 3.5: Trajectories for $\kappa = high$ with different configurations.

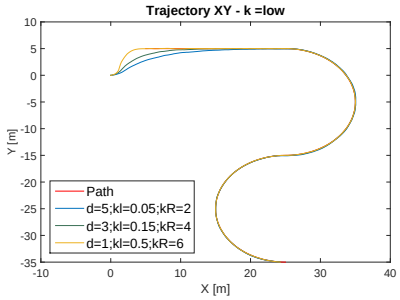


Figure 3.6: Trajectories for $\kappa = low$ with different configurations.

3.4.1 Discussion of the Results

Once the mean curvature of the path has been calculated, and knowing the velocity, κ is then determined and the best values of the parameter are selected. For instance, in Figure 3.5 a path with $\kappa = high$ has been represented as well as three trajectories with different configurations. The best configuration of the system is given by the fuzzy-logic system, i.e. $d = 5$, $k_l = 0.05$ and $k_{\hat{R}} = 2$ (blue line). The other two configurations correspond to $\kappa = medium$ (green line, $d = 3$, $k_l = 0.15$ and $k_{\hat{R}} = 6$), and $\kappa = low$ (yellow line, $d = 1$, $k_l = 0.50$ and $k_{\hat{R}} = 4$). These two last configurations converge faster to the desired path but they present oscillations, specially the configuration for $\kappa = low$, in yellow. Even though the configuration for $\kappa = medium$ provides good results as well, it is slightly worse.

Another example of the behavior depending on the configuration has been represented in Figure 3.6, this time with $\kappa = low$. As it has been already said, the lower the value of κ is, the easier the path can be followed, and therefore all three configurations accomplish the path-following task successfully. However, the values of the parameters proposed by the fuzzy system provide better results, being able to stay on the track from the very beginning.

The results have been measured by integrating the perpendicular distance between the reference path and the actual quadcopter trajectory. In Figure 3.7 this error, in $[m^2]$, has been represented for a wide range of κ values, which is now numerically defined as follows for the sake of this comparison:

$$\kappa = v \cdot \Omega \tag{3.21}$$

The three configurations proposed by the fuzzy system and the trajectories of Figures 3.5 and 3.6 were used to measure the error. It can be seen that the parameters have been properly chosen; for every range of values of κ, the best configuration is the one given by the fuzzy decision system. Note that for $\kappa > 0.8$ the error increases quickly, since the necessary yaw rate to follow the path approaches to the maximum achievable by the quadcopter.

3.5 Kinematics analysis

As already indicated, after proposing a fuzzy logic to set the algorithm parameters, a deep analysis was carried out and a new parameter law proposed,

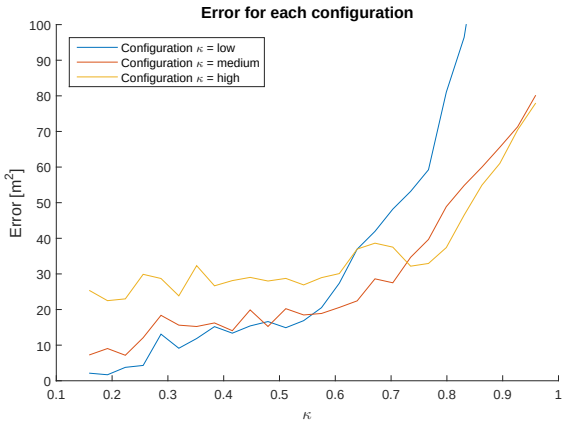


Figure 3.7: Path following error, depending on the configuration and the value of κ.

.

presented in this section.

As in [14] and [25], among others, a kinematics analysis of the movement is needed to understand how the parameters involved affect its behavior. To do that in a simple way, firstly it will be considered that the quacopter moves with a constant velocity and that the time response of the low level controller is negligible. Secondly, we will simplify the problem even more by considering only simple paths, such as a straight line or a circle. The results of the analysis will be then confirmed by simulating the complete model.

3.5.1 Kinematics of the movement

Equations of the quadcopter

The movement of the point $\mathbf{Q}$, see Figure 3.8, can be expressed as:

$$\left.\frac{d\mathbf{Q}}{dt}\right|_{\{I\}} = \left.\frac{d\mathbf{P}}{dt}\right|_{\{I\}} + \left.\frac{d\mathbf{r_Q}}{dt}\right|_{\{I\}} \tag{3.22a}$$

$$\left.\frac{d\mathbf{Q}}{dt}\right|_{\{I\}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}_{\{I\}} \tag{3.22b}$$

$$\left.\frac{d\mathbf{P}}{dt}\right|_{\{I\}} = \begin{bmatrix} \dot{x}_P \\ \dot{y}_P \\ 0 \end{bmatrix}_{\{I\}} = \begin{bmatrix} \dot{l} \\ 0 \\ 0 \end{bmatrix}_{\{\mathcal{F}\}} \tag{3.22c}$$

where the derivatives are given in the inertial frame $\{I\}$, but dP/dt has been expressed in $\{\mathcal{F}\}$; remember that $\dot{l}$ is the linear velocity along the path (Figure 3.1). The relative derivative of the vector $\mathbf{r_Q}$ with respect to frame $\{\mathcal{F}\}$, which is not inertial, is:

$$\left.\frac{d\mathbf{r_Q}}{dt}\right|_{\{I\}} = \left.\frac{d\mathbf{r_Q}}{dt}\right|_{\{\mathcal{F}\}} + \mathbf{w_{F/I}}_{\{\mathcal{F}\}} \wedge \mathbf{r_Q}_{\{\mathcal{F}\}} \tag{3.23}$$

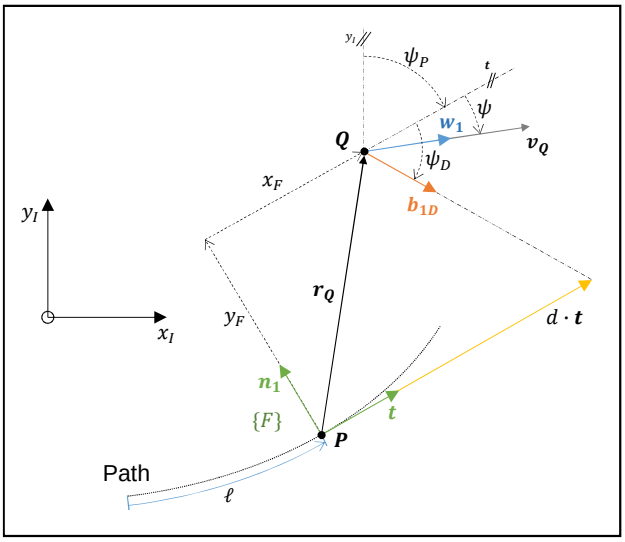


Figure 3.8: 2D kinematics of point P.

.

where

$$\left.\frac{d\mathbf{r_Q}}{dt}\right|_{\{\mathcal{F}\}} = \begin{bmatrix} \dot{x}_F \\ \dot{y}_F \\ 0 \end{bmatrix}_{\{\mathcal{F}\}} \tag{3.24a}$$

$$\mathbf{w_{F/I}}\big|_{\{\mathcal{F}\}} \wedge \mathbf{r_Q}_{\{\mathcal{F}\}} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & -\dot{\psi}_P \\ x_F & y_F & 0 \end{bmatrix}_{\{\mathcal{F}\}} = \begin{bmatrix} \dot{\psi}_P \cdot y_F \\ -\dot{\psi}_P \cdot x_F \\ 0 \end{bmatrix}_{\{\mathcal{F}\}} \tag{3.24b}$$

Note that the previous result (3.24) is expressed in frame $\{\mathcal{F}\}$, so it is needed to change its frame multiplying by:

$$\mathbf{R_F^I} = \begin{bmatrix} \sin\psi_P & -\cos\psi_P & 0 \\ \cos\psi_P & \sin\psi_P & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

With Equations (3.22), (3.23) and (3.24) we get:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}_{\{I\}} = \left(\begin{bmatrix} \sin\psi_P & -\cos\psi_P & 0 \\ \cos\psi_P & \sin\psi_P & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{l} + \dot{x}_F + \dot{\psi}_P \cdot y_F \\ \dot{y}_F - \dot{\psi}_P \cdot x_F \\ 0 \end{bmatrix}_{\{\mathcal{F}\}} \right)_{\{I\}} \tag{3.26}$$

Since the control of the vertical movement will be managed by the low level controller, we can discard the coordinate z in the previous equation, and inverting the matrix (3.25) we have:

$$\dot{x}\sin\psi_P + \dot{y}\cos\psi_P = \dot{l} + \dot{x}_F + \dot{\psi}_P \cdot y_F \tag{3.27a}$$

$$-\dot{x}\cos\psi_P + \dot{y}\sin\psi_P = \dot{y}_F - \dot{\psi}_P \cdot x_F \tag{3.27b}$$

The left side of both equations are nothing but the projection of the absolute velocity vector of the quadcopter on $\mathbf{t}$ direction (see Figure 3.8) and therefore:

$$\dot{x}\sin\psi_P + \dot{y}\cos\psi_P = v_Q \cdot \cos\psi \tag{3.28a}$$

$$-\dot{x}\cos\psi_P + \dot{y}\sin\psi_P = -v_Q \cdot \sin\psi \tag{3.28b}$$

where v_Q is the modulus of the velocity and is considered constant. We finally have:

$$v_Q \cdot \cos\psi = \dot{l} + \dot{x}_F + \dot{\psi}_P \cdot y_F \tag{3.29a}$$

$$-v_Q \cdot \sin\psi = \dot{y}_F - \dot{\psi}_P \cdot x_F \tag{3.29b}$$

On the other hand, because the geometry of the path will be given, we have:

$$\dot{\psi}_P = -\dot{l} \cdot \rho(l) \tag{3.30}$$

where $\dot{l}$ is the velocity of the point P along the curve and $\rho(l)$ is the curvature, having finally:

$$v_Q \cdot \cos\psi = \dot{x}_F + \dot{l} \cdot (1 - \rho \cdot y_F) \tag{3.31a}$$

$$-v_Q \cdot \sin\psi = \dot{y}_F + \dot{l} \cdot \rho \cdot x_F \tag{3.31b}$$

Control law equations

If only 2 dimensions are considered, the control law can be simplified. Recall that the control law is (Equation (3.14)):

$$\begin{bmatrix} q_c \\ r_c \end{bmatrix}_{\{\mathcal{W}\}} = \Pi_R \hat{R}^T \left(\mathbf{R}^{\mathbf{D}}_{\mathbf{F}} \cdot \mathbf{w}_{\mathbf{F/I}}\big|_{\mathcal{F}} + \mathbf{w}_{\mathbf{D/F}}\big|_{\mathcal{D}} \right) - 2 \cdot k_{\hat{R}} \cdot \mathbf{e}_{\hat{\mathbf{R}}} \tag{3.32}$$

Every term of the above equation turns out to be:

$$\hat{\mathbf{R}} = \mathbf{R}^{\mathbf{D}}_{\mathbf{W}} = \begin{bmatrix} \cos(\psi_D - \psi) & \sin(\psi_D - \psi) & 0 \\ -\sin(\psi_D - \psi) & \cos(\psi_D - \psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.33a}$$

$$\mathbf{R}^{\mathbf{D}}_{\mathbf{F}} = \begin{bmatrix} \cos\psi_D & -\sin\psi_D & 0 \\ \sin\psi_D & \cos\psi_D & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.33b}$$

$$\mathbf{w_{F/I}}\big|_{\mathcal{F}} = \begin{bmatrix} 0 \\ 0 \\ -\dot{\psi}_P \end{bmatrix}_{\mathcal{F}} \tag{3.33c}$$

$$\mathbf{w_{D/F}}\big|_{\mathcal{D}} = \begin{bmatrix} 0 \\ 0 \\ -\dot{\psi}_D \end{bmatrix}_{\mathcal{D}} \tag{3.33d}$$

$$\mathbf{e}_{\hat{\mathbf{R}}} = \frac{1}{2}\begin{bmatrix} \hat{R}_{13} \\ -\hat{R}_{12} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 \\ \sin(\psi_D - \psi) \end{bmatrix} \tag{3.33e}$$

With these last results, the rotation commands are:

$$\begin{bmatrix} q_c \\ r_c \end{bmatrix}_{\{w\}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\psi_D - \psi) & \sin(\psi_D - \psi) & 0 \\ -\sin(\psi_D - \psi) & \cos(\psi_D - \psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot$$
$$\left(\begin{bmatrix} \cos\psi_D & -\sin\psi_D & 0 \\ \sin\psi_D & \cos\psi_D & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\dot{\psi}_P \end{bmatrix}_{\mathcal{F}} + \begin{bmatrix} 0 \\ 0 \\ -\dot{\psi}_D \end{bmatrix}_{\mathcal{D}} \right) - k_{\hat{R}} \begin{bmatrix} 0 \\ \sin(\psi_D - \psi) \end{bmatrix}$$
$$= \begin{bmatrix} \sin(\psi_D - \psi) & \cos(\psi_D - \psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\dot{\psi}_P - \dot{\psi}_D \end{bmatrix} - k_{\hat{R}} \begin{bmatrix} 0 \\ \sin(\psi_D - \psi) \end{bmatrix}$$
$$= -\begin{bmatrix} 0 \\ \dot{\psi}_P + \dot{\psi}_D + k_{\hat{R}} \sin(\psi_D - \psi) \end{bmatrix} \tag{3.34}$$

As can be seen, the pitch rate is zero as expected. Let's suppose now that the kinematics of the yaw angle is a 1^{st} order system. Also, let's consider that the low level controller is capable of reaching the yaw set-point fast enough to assume that the system behaves as if the input were a step, i.e. as described in Equation (2.18). Then, the equation describing the yaw angle would be:

$$\psi = \psi_c \left(1 - e^{-t/\tau_l}\right) \tag{3.35}$$

If the time is in the order of magnitude of τ_h, we obtain:

$$\psi = \psi_c \left(1 - e^{-\tau_h/\tau_l}\right) = \psi_c \cdot \alpha_\tau \tag{3.36}$$

In Figure 3.9 the function $\alpha_\tau(\tau_h/\tau_l)$ has been represented. If we consider values:

$$\tau_h/\tau_l > 2 \Rightarrow 0.85 < \alpha_\tau < 1 \tag{3.37}$$

This last result establishes the condition to separate the low level controller, which stabilizes the attitude of the quadcopter, from the high level one, which

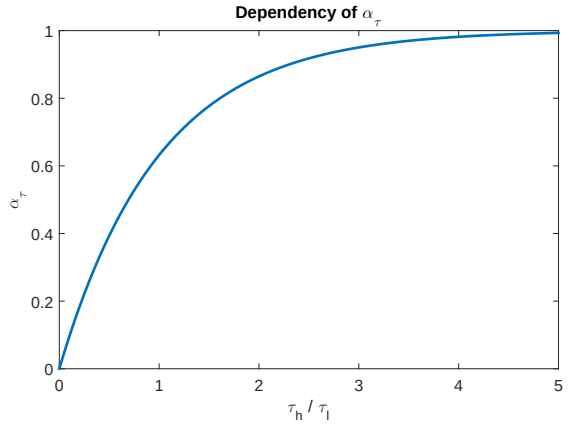


Figure 3.9: Dependency of α_τ with the relationship τ_h/τ_l.

.

controls the heading to follow the given path. Note that this assumption is basically the same we made when Equation (3.35) was proposed: the low level controller response is faster than the high level one, being capable of reaching the reference angle fast enough to consider it as a step input. We can assume then that:

$$\psi \simeq \psi_c \Rightarrow \dot{\psi} \simeq \dot{\psi}_c = -\left. r_c \right|_{\{\mathcal{W}\}} \tag{3.38}$$

Having this last result into consideration, $\left. r_c \right|_{\{\mathcal{W}\}} = -\dot{\psi}$, and considering Equation (3.30):

$$\dot{\psi} = -\dot{l} \cdot \rho(l) + \dot{\psi}_D + k_{\hat{R}} \sin(\psi_D - \psi) \tag{3.39}$$

On the other hand, the other control law, which describes the movement of the virtual point P, Equation (3.11), is:

$$\dot{l} = \left(v_Q \cdot \mathbf{w_1} + k_l \cdot \left. \mathbf{r_Q} \right|_{\{\mathcal{F}\}} \right) \cdot \mathbf{t} = v_Q \cdot \cos\psi + k_l \cdot x_F \tag{3.40}$$

Linealization around the equilibrium point

The complete 2D kinematics of the system for a general case is described by Equations (3.31), (3.39) and (3.40), keeping always in mind the condition assumed in Equation (3.37):

$$v_Q \cdot \cos\psi = \dot{x}_F + \dot{l} \cdot (1 - \rho \cdot y_F) \tag{3.41a}$$

$$-v_Q \cdot \sin\psi = \dot{y}_F + \dot{l} \cdot \rho \cdot x_F \tag{3.41b}$$

$$\dot{\psi} = -\dot{l} \cdot \rho + \dot{\psi}_D + k_{\hat{R}} \sin(\psi_D - \psi) \tag{3.41c}$$

$$\dot{l} = v_Q \cdot \cos\psi + k_l \cdot x_F \tag{3.41d}$$

where the angel ψ_D and its derivative are:

$$\psi_D = \arctan\left(\frac{y_f}{d - x_F} \right) \tag{3.42a}$$

$$\dot{\psi}_D = -\dot{\psi}_P + \frac{\dot{y}_f(d - x_F) + \dot{x}_F y_F}{(d - x_F)^2 + y_F^2} = \dot{l} \cdot \rho + \frac{\dot{y}_f(d - x_F) + \dot{x}_F y_F}{(d - x_F)^2 + y_F^2} \tag{3.42b}$$

Recall that ψ_D is defined in the non-intertial frame $\{\mathcal{F}\}$ and therefore the angular velocity of this frame, $-\dot{\psi}_P$, must be added.

One can check that the point $x_F = y_F = \psi = 0$ is indeed an equilibrium point by introducing these values in Equations (3.41):

$$v_Q = \dot{x}_F + v_Q \Rightarrow \dot{x}_F = 0 \tag{3.43a}$$
$$\dot{y}_F = 0 \tag{3.43b}$$
$$\dot{\psi} = -\dot{l} \cdot \rho + \dot{l} \cdot \rho = 0 \tag{3.43c}$$
$$\dot{l} = v_Q \tag{3.43d}$$
$$\psi_D = 0 \tag{3.43e}$$

When the quadcopter converges to the equilibrium point, variables x_F, y_F and ψ and their derivatives are equal to zero, having been proved that it is an equilibrium point. The quadcopter then moves along the path with a speed $\dot{l} = v_Q$, so the point P does.

To simplify Equations (3.41), we can linearize them around the equilibrium point, considering then:

$$\psi \ll 1 \Rightarrow \sin\psi \simeq \psi, \cos\psi \simeq 1 \tag{3.44a}$$
$$\psi_D \ll 1 \Rightarrow \psi_D = \arctan\left(\frac{y_F}{d - x_F}\right) \simeq \frac{y_F}{d}, \sin\psi_D \simeq \psi_D, \dot{\psi}_D \simeq \frac{\dot{y_F}}{d} \tag{3.44b}$$
$$\dot{l} = v_Q \cdot \cos\psi + k_l \cdot x_F \simeq v_Q \tag{3.44c}$$

we get the linearized equations:

$$-k_l \cdot x_F = \dot{x}_F - v_Q \cdot \rho \cdot y_F \tag{3.45a}$$
$$-v_Q \cdot \psi = \dot{y}_F + v_Q \cdot \rho \cdot x_F \tag{3.45b}$$
$$\dot{\psi} = -v_Q \cdot \rho + \frac{\dot{y}_F}{d} + k_{\hat{R}}(\frac{y_F}{d} - \psi) \tag{3.45c}$$
$$\dot{l} = v_Q \tag{3.45d}$$

3.5.2 Kinematics following a straight line

Let's suppose that the system follows a straight line:

$$\rho = 0 \tag{3.46a}$$
$$\dot{\psi}_P = 0 \tag{3.46b}$$

and that the quadcopter has already converged to the virtual point P, so that we can simplify Equations (3.45) to get:

$$-k_l \cdot x_F = \dot{x}_F \tag{3.47a}$$
$$-v_Q \cdot \psi = \dot{y}_F \tag{3.47b}$$
$$\dot{\psi} = \frac{\dot{y_F}}{d} + k_{\hat{R}}\left(\frac{y_F}{d} - \psi\right) \tag{3.47c}$$

From Equation 3.47a we can get the dynamics of x_F:

$$\dot{x}_F = -k_l \cdot x_F \Rightarrow x_F = x_{F_0} e^{-k_l t} \tag{3.48}$$

which is asymptotically stable if $k_l > 0$. This parameter has a low influence on the algorithm and might be set independently from the other parameters without a deep analysis:

$$k_l = 2 \text{ s}^{-1} \tag{3.49}$$

Note that k_l rules the dynamic of a virtual point, there is not any physical element involved, and therefore x_F will converge fast without any problem. Its time constant is 0.5 s and, as we will see, is in the same order as the other time constants present in the movement. Moreover, higher or lower values show low influence on the system response. On the other hand, by the Laplace transformation of Equations (3.47b) and (3.47c), we obtain:

$$Y_F(s)\left(s^2 + \left(\frac{v_Q}{d} + 1\right)s + k_{\hat{R}}\frac{v_Q}{d}\right) = s \cdot y_F(0) \tag{3.50a}$$

$$\Phi(s)\left(s^2 + \left(\frac{v_Q}{d} + 1\right)s + k_{\hat{R}}\frac{v_Q}{d}\right) = s \cdot \psi(0) \tag{3.50b}$$

where it has been supposed $\dot{y}_F(0) = \dot{\psi}(0) = 0$. Both equations in the Laplace domain have the same characteristic equation, whose roots are:

$$s_{1,2} = \frac{-d' k_{\hat{R}} - 1 \pm (1 - d' k_{\hat{R}})}{2d'} \tag{3.51}$$

$$s_1 = -k_{\hat{R}} \tag{3.52a}$$

$$s_2 = -\frac{1}{d'} = -\frac{v_Q}{d} \tag{3.52b}$$

where $d' = \frac{d}{v_Q}$. The characteristic time of the high level controller, τ_h, can be assumed to be the highest of both time constants and it can be expressed as,

$$\tau_h = \max\left\{\frac{1}{k_{\hat{R}}}, \frac{d}{v_Q}\right\} \tag{3.53}$$

We can also express the behavior of the system in terms of its damping ratio and its natural frequency:

$$\omega_n = \sqrt{k_{\hat{R}}\frac{v_Q}{d}} = \sqrt{\frac{k_{\hat{R}}}{d'}} \tag{3.54a}$$

$$\xi = \frac{1}{2}\frac{\frac{v_Q}{d} + 1}{\sqrt{k_{\hat{R}}\frac{v_Q}{d}}} = \frac{1}{2}\frac{1 + d'}{\sqrt{k_{\hat{R}} d'}} \tag{3.54b}$$

In Figure 3.10 the values of both parameters have been represented depending on $k_{\hat{R}}$ and for some values of d'.

The above results show that the system will asymptotically converge when following a straight line as long as all three parameters are higher than zero. This implies that we are free to choose the values of the parameters, since the system will always converge.

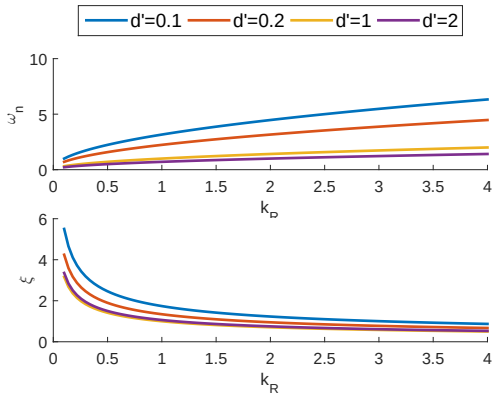


Figure 3.10: Natural frequency and damping ratio of the linearized system following a straight line.

.

Selection of the design parameters

As we have already said, we can freely choose the parameters $k_{\hat{R}}$ and d because theoretically the system will converge asymptotically if both are positive. However, some simulations show that the behavior might become unexpectedly unstable.

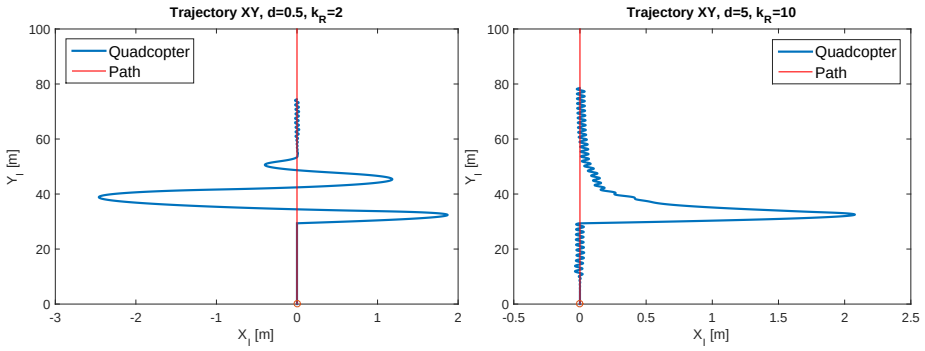
In Figure 3.11 the trajectory of the quadcopter following a straight line with a velocity of 2 m/s has been represented. At some point, a perturbation in the X_I direction is applied and the quadcopter tries to return to the path to be followed. In Figure 3.11a, a configuration of the parameters with an abnormally low value of d, equal to 0.5 m, has been chosen, while the parameter $k_{\hat{R}} = 2$ s^{-1} has a standard value. Note that the quadcopter has difficulties to return to the path (big oscillations) and when it does, there are still small oscillations around the equilibrium point. On the other hand, in Figure 3.11b, a high value of $k_{\hat{R}}$ has been considered, equal to 10 s^{-1}, while the distance d has been chosen equal to 5 m. Even before the perturbation takes place, the quadcopter starts oscillating around the equilibrium point.

These results contradict what was found with Equation (3.52), i.e. that the linearized system converges asymptotically for $k_{\hat{R}}, d > 0$. However, in both examples of Figure 3.11 there are instabilities when the system is close to the equilibrium point, $x_F \simeq y_F \simeq \psi \simeq 0$. The reason why this happens is because the first assumptions we made when developing the kinematics equations: the low level and the high level controller are decoupled, see Equations (3.37) and (3.38). We will now show that this assumption might not be true in some cases. Let's consider that:

$$\frac{v_Q}{d} = k_{\hat{R}} = \frac{1}{\tau_h} \tag{3.55}$$

so that there is only one double root in Equation (3.51). We can define the effort of the yaw control as:

$$\sigma_r = \int |r_c| \cdot dt \tag{3.56}$$



(a) Following a straight line with a perturbation with d low.

(b) Following a straight line with a perturbation with k_R high.

Figure 3.11: Two configurations which present instabilities when following a straight line. $k_l = 0.15$, $v_Q = 2$ m/s.

In Figure 3.12 this effort has been represented, depending on the time constants τ_k and τ_d, defined as:

$$\tau_k = \frac{1}{k_{\hat{R}}} \tag{3.57a}$$

$$\tau_d = \frac{d}{v_Q} \tag{3.57b}$$

Note that in this case $\tau_k = \tau_d = \tau_h$, being τ_h the time constant of the high level controller. The time constant, estimated for the low level controller, $\tau_l = 0.45$ s, see Equation (2.20), has also been represented by a dashed line. As it can be seen, for time constants lower that 0.5 s, the effort increases rapidly, indicating oscillations around the equilibrium point as shown in Figure 3.11. Theoretically we would expect this behavior to appear for time constants $\tau_h \simeq 2 \cdot \tau_l = 0.9$ s; however, τ_h can be reduced up to 0.5 s, practically the same time as the low level time constant. This might indicate that the estimation was too conservative and τ_l could be close to 0.25 s. On the other hand, for values of τ_h higher than 0.5, the effort is practically constant.

In Figure 3.13a the result of Figure 3.12 has been extended; now τ_k and τ_d take different values, showing the complete behavior of the effort when both parameters change. Every curve presents a similar behavior; for high values of τ_k, the effort is low and constant and at some point, it increases quickly because the coupling between the low and the high level controllers. This sudden increase can be delayed by making the time constant τ_d bigger. All the curves tend to coincide as τ_d increases, meaning that there are barely improvements for $\tau_d > 1.3$ s. To separate the set of τ_k and τ_d values for which the controllers start coupling, a *coupling frontier* has been represented, dotted line in Figure 3.13a. Values of the time constant on the left of it, would lead to instabilities such as the one shown in Figure 3.11.

It is clear now which values of the time constants τ_k and τ_d shall be avoided to prevent instabilities when following a straight line. Another indicator to measure how good the chosen parameters are, is the cross-track error, defined

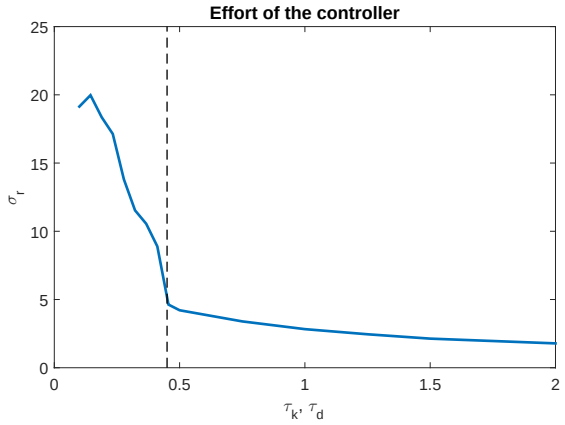
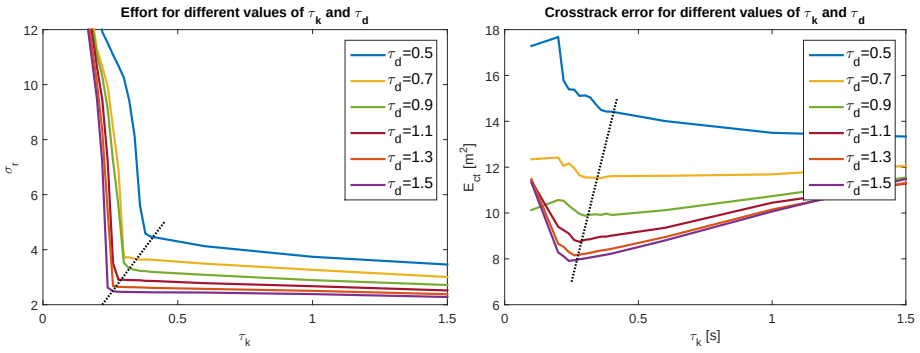


Figure 3.12: Effort of the yaw control, depending on the time constant $\tau_k = \tau_d$. The dashed line represents the estimated low level controller time constant, τ_l, see Equation (2.20).



(a) Yaw control effort following a straight line, depending on τ_k and τ_d.

(b) Cross-track error following a straight line, depending on τ_k and τ_d.

Figure 3.13: Yaw control effort and cross-track error depending on the time constants τ_k and τ_d following a straight line. The dotted line indicates the coupling frontier between the low and the high level controllers.

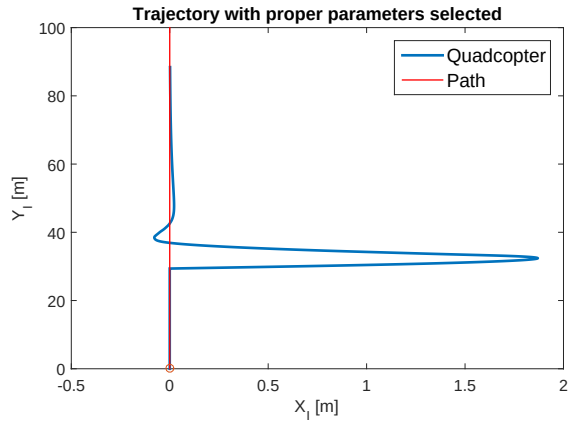


Figure 3.14: Trajectory following a straight line with $v_Q = 2$ m/s, $k_l = 2$, $k_{\hat{R}} = 2.5$ s^{-1} and $d = 3$ m.

as the integral of the perpendicular distance from the path to the quadcopter:

$$E_{ct} = \int_0^{l_{max}} |\mathbf{r}_\perp(l)|\, dl \tag{3.58}$$

where $\mathbf{r}_\perp(l)$ is the perpendicular vector from the point P on the path to the quadcopter trajectory. In Figure 3.13b this error has been represented for the same set of values of τ_k and τ_d. The error increases if τ_d is lower and the curves tend again to coincide when τ_d increases. The coupling frontier has been again represented with a dotted line. For high enough values of τ_d, the error has a minimum for values of τ_k close to this frontier.

Having studied the results of Figure 3.13, we can propose appropriate values of both parameters, far enough from the coupling behavior but with low cross-track error:

$$\tau_d = 1.5 \text{ s} \Rightarrow d = v_Q \cdot \tau_d = 3 \text{ m} \tag{3.59}$$

$$\tau_k = 0.4 \Rightarrow k_{\hat{R}} = \frac{1}{\tau_k} = 2.5 \text{ s}^{-1} \tag{3.60}$$

Note that we could have chosen a lower value for τ_k to reduce the cross-track error, but the improvement would have been small and we would have gotten very close to the coupling frontier. Bigger values of τ_d would not either improve the performance much.

In Figure 3.14, the same simulation as in Figure 3.11 has been represented, but with the proper values of d and $k_{\hat{R}}$ from Equations (3.59) and (3.60). As it can be seen, there is no oscillations around the equilibrium points and the quadcopter returns to the path after the perturbation quickly with an overshoot around 3%.

3.5.3 Kinematics following a circular trajectory

Another trajectory that can be analytically analyzed is the circumference. Knowing that any trajectory can be represented by a set of circumferences whose

centers and curvature radius depend on the position along the path, the consequences and the parameter laws here obtained can be extrapolated. In case of a circular trajectory, we have:

$$\rho = \frac{1}{R} \tag{3.61}$$

where R is the curvature radius of the circular trajectory. Introducing it in Equations (3.45) we get:

$$-k_l \cdot x_F = \dot{x}_F - \frac{v_Q}{R} \cdot y_F \tag{3.62a}$$

$$-v_Q \cdot \psi = \dot{y}_F + \frac{v_Q}{R} \cdot x_F \tag{3.62b}$$

$$\dot{\psi} = -\frac{v_Q}{R} + \frac{\dot{y}_F}{d} + k_{\hat{R}}(\frac{y_F}{d} - \psi) \tag{3.62c}$$

These equations depend not only on the 3 parameters of the algorithm, but also on v_Q and R, which makes the problem difficult to be analyzed. Lets make the equations dimensionless with the following factors:

$$[\text{time}] = \frac{R}{v_Q}$$
$$[\text{distance}] = R$$
$$[\text{velocity}] = v_Q$$

The dimensionless variables will be then:

$$\tilde{x}_F = \frac{x_F}{R},\ \dot{\tilde{x}}_F = \frac{\dot{x}_F}{v_Q}$$
$$\tilde{y}_F = \frac{y_F}{R},\ \dot{\tilde{y}}_F = \frac{\dot{y}_F}{v_Q}$$
$$\tilde{\psi} = \psi,\ \dot{\tilde{\psi}} = \dot{\psi}\frac{R}{v_Q}$$
$$\tilde{k}_{\hat{R}} = k_{\hat{R}}\frac{R}{v_Q}$$
$$\tilde{d} = \frac{d}{R}$$
$$\tilde{k}_l = k_l\frac{R}{v_Q}$$

Using the dimensionless variables, Equations (3.62) become:

$$-\tilde{k}_l \cdot \tilde{x}_F = \dot{\tilde{x}}_F - \tilde{y}_F \tag{3.65a}$$

$$-\tilde{\psi} = \dot{\tilde{y}}_F + \tilde{x}_F \tag{3.65b}$$

$$\dot{\tilde{\psi}} = -1 + \frac{\dot{\tilde{y}}_F}{\tilde{d}} + \tilde{k}_{\hat{R}}(\frac{\tilde{y}_F}{\tilde{d}} - \tilde{\psi}) \tag{3.65c}$$

As can be seen, the kinematic equations do not depend on v_Q nor on R explicitly any more, which makes the analysis easier. These equations can be

written in their matrix form:

$$\begin{bmatrix} \dot{\tilde{x}}_F \\ \dot{\tilde{y}}_F \\ \dot{\tilde{\psi}} \end{bmatrix} = \begin{bmatrix} -\tilde{k}_l & 1 & 0 \\ -1 & 0 & -1 \\ -\frac{1}{\tilde{d}} & \frac{\tilde{k}_{\hat{R}}}{\tilde{d}} & -\frac{1}{\tilde{d}} - \tilde{k}_{\hat{R}} \end{bmatrix} \begin{bmatrix} \tilde{x}_F \\ \tilde{y}_F \\ \tilde{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{3.66}$$

If we take the homogeneous solution, the characteristic equation turns out to be:

$$(\lambda + \tilde{k}_{\hat{R}}) \cdot \left(\lambda^2 + \lambda \left(\tilde{k}_l + \frac{1}{\tilde{d}} \right) + 1 + \frac{\tilde{k}_l}{\tilde{d}} \right) = 0 \tag{3.67}$$

The first root depends only on the parameter $\tilde{k}_{\hat{R}}$ and it does not appear in the other two, so it can be independently analyzed. The time constant of this root is:

$$\tilde{\tau}_1 = \frac{1}{\tilde{k}_{\hat{R}}} \tag{3.68}$$

The other two roots are more complicate expressions of $\tilde{d}$ and $\tilde{k}_l$ and, therefore, its analysis is not easy to carry out. Instead of it, we can expressed the second order equation in terms of its natural frequency and damping ratio:

$$\tilde{\omega}_n = \sqrt{1 + \frac{\tilde{k}_l}{\tilde{d}}} = \omega_n \frac{R}{v_Q} \tag{3.69a}$$

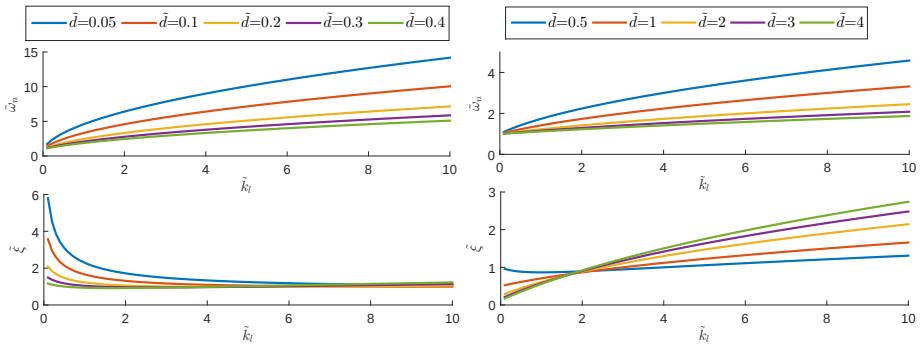
$$\tilde{\xi} = \frac{1}{2} \frac{\tilde{k}_l + \frac{1}{\tilde{d}}}{\sqrt{1 + \frac{\tilde{k}_l}{\tilde{d}}}} = \xi \tag{3.69b}$$

Note that in the previous equation, the natural frequency and the damping ratio are also dimensionless. The time constant of the 2$^{\text{nd}}$ order equation is:

$$\tilde{\tau}_2 = \frac{1}{\tilde{\xi}\tilde{\omega}_n} \tag{3.70}$$

In Figure 3.15 the values of both parameters from Equations (3.69) have been represented for a wide range of values of $\tilde{k}_l$ and for low (Figure 3.15a) and high values (Figure 3.15b) of $\tilde{d}$. The bigger the $\tilde{d}$ is, the lower the natural frequency becomes; as $\tilde{k}_l$ increases, the natural frequency also increases smoothly. In the case of the damping ratio, the behavior is not that regular; for values of $\tilde{d}$ lower than 0.3, the damping ratio decreases fast and remains almost constant for $\tilde{k}_l > 2$. For values of $\tilde{d} > 1$, it increases almost linearly with $\tilde{k}_l$; for values between 0.3 and 1, $\tilde{\xi}$ remains aproximately constant and equal to 1. As it can be seen, the system is stable in any case for any value of the parameters.

We would like to make both time constant as small as possible, since it would mean that the response of the system would be as fast as it can. As we saw in the straight line following case, there is a frontier we should not cross, since the low level and the high level controllers start coupling.



(a) Low values of $\tilde{d}$.

(b) High values of $\tilde{d}$.

Figure 3.15: Natural frequency and damping ratio of the system, both dimensionless, following a circular trajectory for different values of $\tilde{d}$ and $\tilde{k}_l$.

Selection of the design parameters

As it was done in the analysis following a straight line, we will first check for which time constant both controllers start coupling. To do that we will force the 1st order and the 2nd order equations in Equation (3.67) to have the same time constant:

$$\tilde{\tau}_1 = \tilde{\tau}_2 = \tilde{\tau}_h \tag{3.71}$$

being $\tilde{\tau}_h$ the dimensionless time constant of the high level controller. Although all the equations have been made dimensionless and they do not depend on the radius R nor the quadcopter velocity v_Q, we must choose two values to carry out the simulations. We will consider from now on:

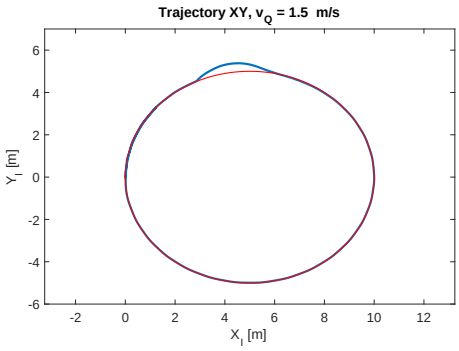
$$R = 5 \text{ m} \tag{3.72a}$$

$$v_Q = 1.5 \text{ m/s} \tag{3.72b}$$

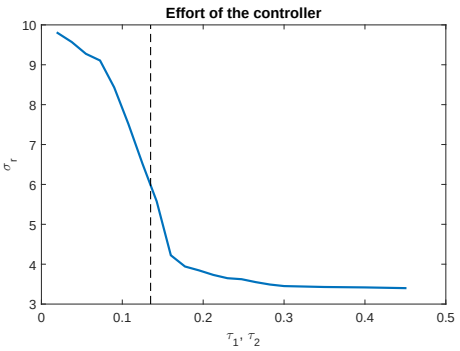
We will also simulate the system response with a perperdincular perturbation on the quadcotper, to check how fast it can return to the reference path. In Figure 3.16a an example of a simulation has been represented. In Figure 3.16b the effort, defined by Equation (3.56), has been represented for a range of values of $\tilde{\tau}_h$. The dashed line is the dimensionless estimated value of the low level controller time constant, which was calculated by:

$$\tilde{\tau}_l = \tau_l \frac{v_Q}{R} = 0.135 \tag{3.73}$$

As it happened with the straight line path, there is a certain value of the time constant of the high level controller for which the effort following the circle increases faster, $\tilde{\tau}_h \simeq 0.16$. Again, as it happened following a straight line, this change in the slope of the curve appears for values approximately equal to $\tilde{\tau}_l$ (dashed line), although it would be expected to show up for $\tilde{\tau}_h \simeq 2 \cdot \tilde{\tau}_l$. Note also that the increase in the effort is not as remarkable as in the straight line case because following a circumference needs itself more effort, and the extra effort due to the coupling between the two controllers is then hidden.



(a) Circular trajectory of the quadcopter with a perturbation.



(b) Effort following the circular path of Figure 3.16a with $\tilde{\tau}_h = \tilde{\tau}_1 = \tilde{\tau}_2$

Figure 3.16: On the left figure, example of circular trajectory with $R = 5$ m and $v_Q = 1.5$ m/s, with a perpendicular perturbation which moves the quadcopter off the path. On the right figure, effort of the yaw control, depending on the time constant $\tilde{\tau}_h = \tilde{\tau}_1 = \tilde{\tau}_2$. The dashed line represents the dimensionless estimated low level controller time constant, $\tilde{\tau}_l$, see Equation (2.20).

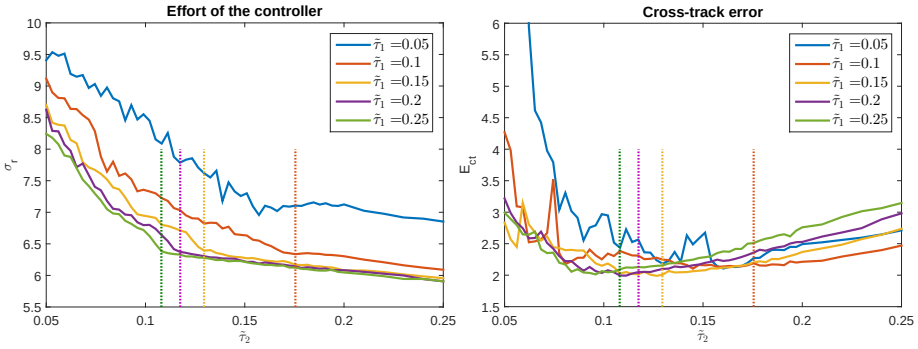
Once we know approximately where the coupling could be, around $\tilde{\tau}_h \simeq 0.16$, we can simulate the system measuring the effort needed to follow the circular path and the cross-track error, as defined in Equation (3.58), for times $\tilde{\tau}_1$ and $\tilde{\tau}_2$ close to it. To do it, we will assign a range of values to $\tilde{\tau}_1$ and with Equation (3.68) we will calculate $\tilde{k}_{\hat{R}}$; we will also assign values to $\tilde{\tau}_2$. From Equations (3.69) and (3.70) we obtain:

$$\tilde{d} = \frac{\tilde{\tau}_2}{2 - \tilde{\tau}_2 \cdot \tilde{k}_l} \tag{3.74}$$

Since in the above equation we have two independent variables, we will take $\tilde{k}_l = 4$, knowing from Figure 3.15 the low influence of this parameter on the damping ratio and on the natural frequency. The values of $\tilde{d}$ will be close to 0.2, as we will see afterwards in Figure 3.18a, and from Figure 3.15a we effectively see that there are barely variations. Once we have chosen k_l, we are able then to determine the value of $\tilde{d}$ with the previous equation.

In Figure 3.17 the effort and the cross-track error have been represented for different values of $\tilde{\tau}_1$ and $\tilde{\tau}_2$. As can be seen in Figure 3.17a, the behavior of the effort is similar to the one presented in Figure 3.16b; at some value of $\tilde{\tau}_2$ the effort increases, depending on $\tilde{\tau}_1$ as well, which delimiters the coupling frontier, represented with dotted lines for every value of $\tilde{\tau}_1$. On the right side of these frontiers both controllers are decoupled and the effort decreases linearly as $\tilde{\tau}_2$ increases. Note that these coupling frontiers are not as well defined as they were in the straight line path. If the coefficient v_Q/R were higher (for these simulations, v_Q/R =0.3 s^{-1}), the frontiers would be even smother. For $\tilde{\tau}_1 = 0.05$, the frontier is not identifiable.

In Figure 3.17b, the cross-track error and the coupling frontiers have been represented. All the curves present a minimum in a range of values of $\tilde{\tau}_2$ on the right side of the coupling frontiers, after which they start increasing slowly. We



(a) Yaw control effort following a circular trajectory, depending on $\tilde{\tau}_1$ and $\tilde{\tau}_2$.

(b) Cross-track error following a circular trajectory, depending on $\tilde{\tau}_1$ and $\tilde{\tau}_2$.

Figure 3.17: Yaw control effort and cross-track error depending on the dimensionless time constants $\tilde{\tau}_1$ and $\tilde{\tau}_2$ following a circular trajectory. The dotted lines indicate the coupling frontiers between the low and the high level controller for every $\tilde{\tau}_1$.

will choose as optimum values for the time constants:

$$\tilde{\tau}_1 = 0.15$$
$$\tilde{\tau}_2 = 0.15$$

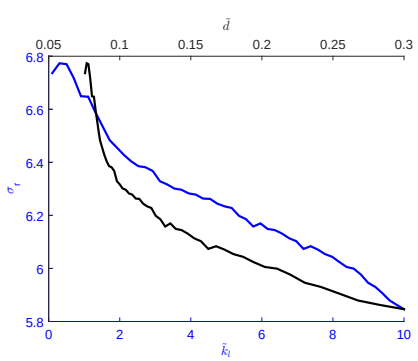
The value $\tilde{\tau}_1$=0.15 has been chosen because it presents lower cross-track error and approximately the same effort as the others. Instead of choosing $\tilde{\tau}_2$ as close as possible to the coupling frontier, it was decided to leave a margin, so finally $\tilde{\tau}_2$=0.15 was selected. Now that we have chosen $\tilde{\tau}_1$ and $\tilde{\tau}_2$, the variable $\tilde{k}_{\hat{R}}$ is determined:

$$\tilde{k}_{\hat{R}} = \frac{1}{\tilde{\tau}_1} = 6.7 \tag{3.76}$$

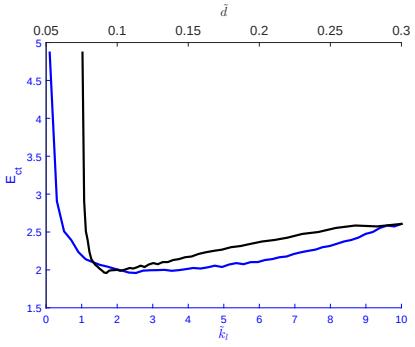
It is not the case for the two parameters $\tilde{d}$ and $\tilde{k}_l$,. There exists infinite combinations of both so that we get $\tilde{\tau}_2 = 0.15$. Having fixed it, in Figure 3.18b the cross-track error has been represented against $\tilde{k}_l$ in blue, bottom axis, and against $\tilde{d}$, in red, upper axis. Both curves are in fact the same Equation (3.74), but taking a different independent variable. It has been considered a wide range of values of $\tilde{k}_l$, being the minimum 10 times lower than the highest value; the variable $\tilde{d}$ is swept until 3 times the minimum value considered.

The error decreases abruptly until $\tilde{k}_l = 1$, having a minimum at $\simeq 1.5$, and for higher values the error increases slowly. In any case, it must be remarked, for medium values of $\tilde{k}_l$ (between 2 and 6) the cross-track error barely changes, having a variation around 10%. The effort, presented in Figure (3.18b), is practically constant for the complete range of values of $\tilde{k}_l$ and close to the corresponding point, $\tilde{\tau}_1 = \tilde{\tau}_2 = 0.15$, in Figure 3.17a.

This result indicates that we have chosen properly the variable $\tilde{\tau}_2$ as design parameter, since high variations of $\tilde{d}$ and $\tilde{k}_l$, keeping $\tilde{\tau}_2$ constant, do not change the performance of the algorithm considerably. We can use this degree of freedom (we are able to choose these two parameters having fixed $\tilde{\tau}_2$) to better adapt the algorithm taking into account a wider range of values of v_Q/R.



(a) Effort depending on $\tilde{k}_l$ (blue) and on $\tilde{d}$ (black).



(b) Cross-track error depending on $\tilde{k}_l$ (blue) and on $\tilde{d}$ (black), with a minimum value at $\tilde{k}_l \simeq 2.5$

Figure 3.18: Effort and cross-track error depending on $\tilde{k}_l$, having fixed $\tilde{\tau}_1 = 0.15$ and $\tilde{\tau}_2 = 0.15$.

Recall that from the very beginning, we have linearized and made dimensionless the equations so that the selection of the parameters is independently from v_Q/R, although for the sake of the simulation, we had to choose two values, see Equation (3.72).

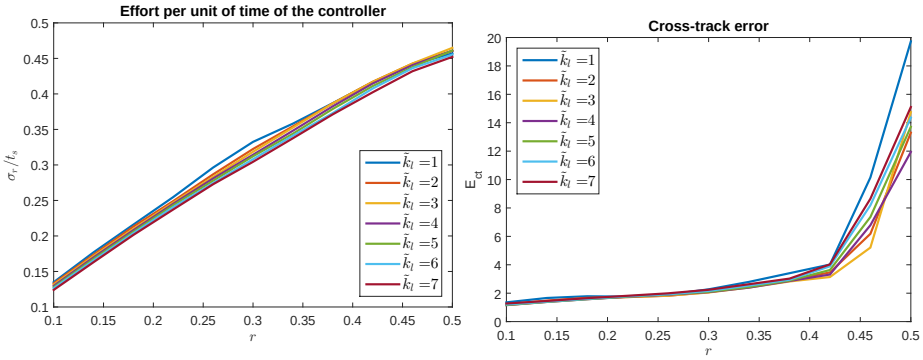
In Figure 3.19 the effort and the cross-track error have been represented for different values of $\tilde{k}_l$ and depending on the value of r, defined as:

$$r = \frac{v_Q}{R} \tag{3.77}$$

This variable is the yaw rate needed to properly follow a circular path. The simulations have been carried out fixing the radius $R = 5$ m and varying the velocity of the quadcopter. Remember that the maximum yaw rate of the quadcopter is $r_{max} = 0.5$ s^{-1}.

For the sake of the cross-track error representation, the performance has been measured until a certain position of the quadcopter along the path; that means that for higher r values, the velocity v_Q is higher and the quadcopter reaches the final point earlier, being then, in this shorter time, the total effort accumulated lower. Therefore, to better represent the effort needed in every case, the effort has been divided by the time of each simulation and represented in Figure 3.19a. We can see that the effort per unit of simulation time increases linearly with r, and its slope, which should be ~ 1, is equal to ~ 0.9. This difference is due to the constant perturbation on the quadcopter when following the circular path, which is the same for all the simulations and adds higher efforts for lower r, making the slope less pronounced. From these figures we also see that the higher the parameter $\tilde{k}_l$ is, the slightly lower the needed effort turns out to be.

On the other hand, in Figure 3.19b the cross-track error has been represented for the same cases. For values of r up to 0.4 s^{-1}, the error increases slowly and there is no variation with $\tilde{k}_l$. This is the expected behavior, since the design parameter $\tilde{\tau}_2$ is kept constant. The higher r is, the more difficult to follow the path is, so that the error increases at a slow pace, but still the path can be



(a) Effort per unit of time depending on the required angular velocity r.

(b) Cross-track error depending on the required angular velocity r.

Figure 3.19: Effort and cross-track error depending on the angular velocity required to properly follow the circular path, where $\tilde{\tau}_1 = \tilde{\tau}_1 = 0.15$; the radius of the circumference $R = 5$ m, and the velocity has been changed accordingly with r.

followed without problem. Note that the analysis done by making the equation dimensionless, implies that there is not dependence on $r = v_Q/R$, and that, indeed, this dependence barely exists for low and medium values of r. For $r > 0.4$ s^{-1}, the error starts increasing rapidly and there are differences between the curves for the different values of $\tilde{k}_l$. This abrupt increase appears as r gets close to the maximum yaw rate of the quadcopter, $r_{max} = 0.5$ s^{-1}, appearing then a limitation not considered in the system analysis.

Since there are only differences for high values of r, we will look at them to choose $\tilde{k}_l$. The solution $\tilde{k}_l = 3$, in yellow, gives good results for values of r up to 0.45 s^{-1} so this value will be chosen:

$$\tilde{k}_l = 3 \tag{3.78}$$

Once we have determined $\tilde{k}_l$, from Equation (3.74) we finally get the value of $\tilde{d}$:

$$\tilde{d} = 0.1 \tag{3.79}$$

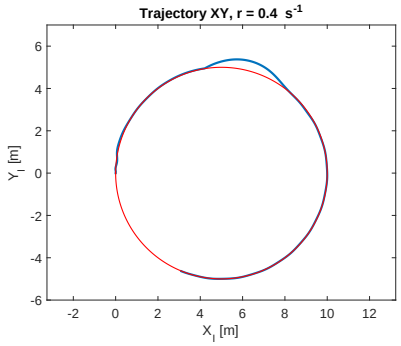
With this selection of the parameters, the natural frequency and the damping ratio of the 2$^{\text{nd}}$ order equation are:

$$\tilde{\omega}_n = 5.66$$
$$\tilde{\xi} = \xi = 1.18$$

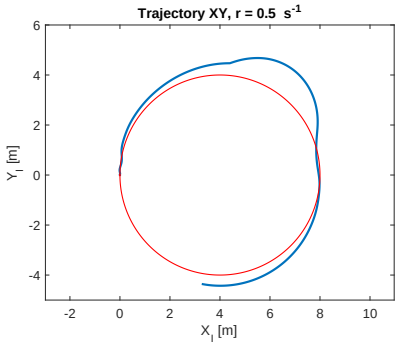
For instance, if $R = 5$ m and $v_Q = 1.5$ m/s, the natural frequency is:

$$\omega_n = \tilde{\omega}_n \frac{v_Q}{R} = 1.70 \text{ s}^{-1} \tag{3.81}$$

We have now all the laws which define the value of the parameters of the



(a) Quadcopter circular trajectory with $r = 0.4$ s^{-1}.



(b) Quadcopter circular trajectory with $r = 0.5$ s^{-1} (maximum yaw rate).

Figure 3.20: Two examples of following a circular path with the parameter configuration finally chosen. In the left figure, $R = 5$ m and $v_Q = 2$ m/s. In the right figure, $R = 5$ m and $v_Q = 2.5$ m/s; the quadcopter is unable to properly follow the reference path.

algorithm:

$$k_{\hat{R}} = 6.7 \cdot \frac{v_Q}{R} \tag{3.82a}$$

$$d = 0.1 \cdot R \tag{3.82b}$$

$$k_l = 3 \cdot \frac{v_Q}{R} \tag{3.82c}$$

In Figure 3.20 two examples of following a circular path with this configuration have been represented. For $r = 0.4$ s^{-1}, Figure 3.20a, the quadcopter is able to follow the path without problem, although r in this case is already quite demanding. In Figure 3.20b, on the contrary, the quadcopter is not able to properly follow the path since the required yaw rate $r = 0.5$ s^{-1} is equal to the maximum possible value.

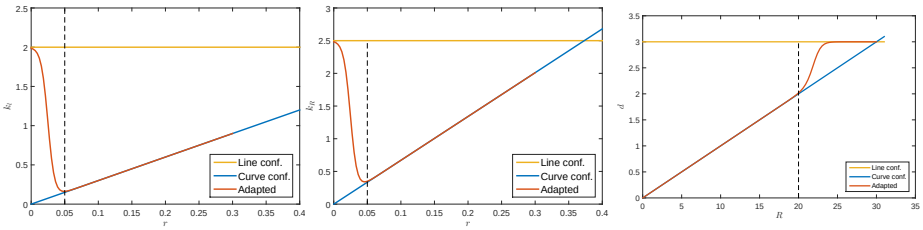
3.6 General parameters law

In the previous sections the kinematics of the quadcopter using the proposed path-following algorithm was analyzed, distinguishing between the configuration when following a straight line and a circular path. As result of such analysis, a set of values of the parameters was chosen for the linear paths, as well as a set of laws depending on the velocity of the aircraft v_Q and the curvature radius R, see Table 3.3. In that table it can be seen that when the curved trajectory laws approach the linear trajectory values by making the radius R tend to infinity, the values of the parameters do not match:

$$R \to \infty \Rightarrow \begin{cases} k_{\hat{R}} \to 0 \\ d \to \infty \\ k_l \to 0 \end{cases} \tag{3.83}$$

Linear trajectory	**Curved trajectory**
$k_{\hat{R}} = 2.5$	$k_{\hat{R}} = 6.7 \cdot \dfrac{v_Q}{R}$
$d = 3$	$d = 0.1 \cdot R$
$k_l = 2$	$k_l = 3 \cdot \dfrac{v_Q}{R}$

Table 3.3: Values of the parameters chosen to follow a linear trajectory, and parameters law to follow a curved path.



(a) $k_{\hat{R}}$ adaptation. (b) k_l adaptation. (c) d adaptation.

Figure 3.21: Adaptation of the parameters $k_{\hat{R}}$, k_l and d for linear and curved trajectories. The vertical dashed line marks the limit considered between linear and curved trajectory. To draw the limit in the d case, it has been considered $v_Q = 1$ m/s.

This means that both results must be adapted so that a general law for all parameters is obtained. In Figure 3.21, the values of the parameters chosen when following a straight line, in yellow, and when following a circular trajectory, in blue, have been represented. In the ordinate axis, the variable r is the yaw rate necessary to follow the path, as it was defined in Equation 3.77. As can be seen, the more the r is reduced, the more both solutions differ.

To create a transition function from one solution to the other, we must first define a r_{lim}, which is a limit to separate what we consider as an almost-linear trajectory, and as a curved trajectory. We can take a low value of r, such us:

$$r_{lim} = 0.05 \text{ rad/s} \tag{3.84}$$

This value, which is 10 times lower than the maximum yaw rate, is low enough to consider that the trajectory is almost a line. For $r < r_{lim}$, the parameters should start getting closer to the line trajectory values. In Figures 3.21a and 3.21b, r_{lim} has been drawn with a dotted line.

For the parameter d, this value is not straightforward applicable, since it does not depend on r directly. This means that it might happen that $r < r_{lim}$, i.e. the trajectory might be considered linear, but the parameter d takes a value which does not correspond to this case. To avoid this situation, we can define

a limiting radius dependent on the velocity of the quadcopter as follows:

$$R_{lim} = \frac{v_Q}{r_{lim}} \tag{3.85}$$

R_{lim} must be always equal or lower to the corresponding value when both solutions cross themselves (Figure 3.21c), so that $d \leq 3$ m. In this figure, R_{lim} has been drawn with a dashed line with $v_Q = 1$ m/s, as an example.

We can now propose a unique law for the three parameters, which is applicable for the complete range of values of r and R:

$$k_{\hat{R}} = (1 - s_{k_{\hat{R}}}) \cdot 2.5 + s_{k_{\hat{R}}} \cdot 6.7 \cdot r \tag{3.86a}$$

$$d = s_d \cdot 3 + (1 - s_d) \cdot 0.1 \cdot R \tag{3.86b}$$

$$k_l = (1 - s_{k_l}) \cdot 2 + s_{k_l} \cdot 3 \cdot r \tag{3.86c}$$

where s are functions to adapt one solution to the other one:

$$s_{k_{\hat{R}}} = s_{k_l} = \frac{\tanh\left(100(r - r_{lim}/2)\right) + 1}{2} \tag{3.87a}$$

$$s_d = \frac{\tanh\left(R - R_{lim} - 2\right) + 1}{2} \tag{3.87b}$$

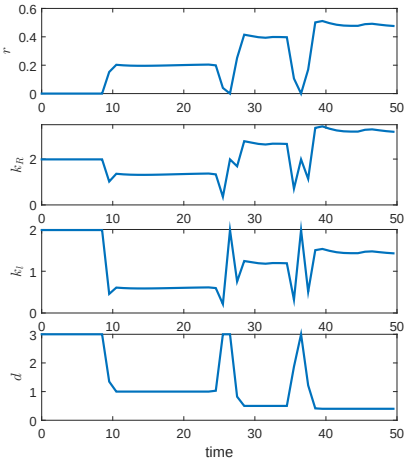
In Figure 3.21 the adapted laws of the parameters, defined above, has been represented in red. Note that the adapted functions are equal to the laws designed to follow a curved trajectory until r_{lim} and R_{lim}, when they change rapidly to reach the value corresponding to the linear trajectory.

The only remaining discussion is how the required yaw rate r will be calculated. We could take the curvature of the path every instant at the virtual point P, being then $r = v_Q/R(P)$. However, the change in the curvature along the path might introduce a continuous change in the parameters and therefore, we would have a new kinematics of the movement, since the parameters were considered as constant in the analysis carried out in Sections 3.5.2 and 3.5.3. We could rather calculate the mean value of the curvature in advance of the point P, which is the section of the path the quadcopter is about to travel along. The parameters will be updated every second, a sample time high enough to be sure that in the kinematics analysis the parameters can be considered as constant. The length along which the mean value will be calculated is then:

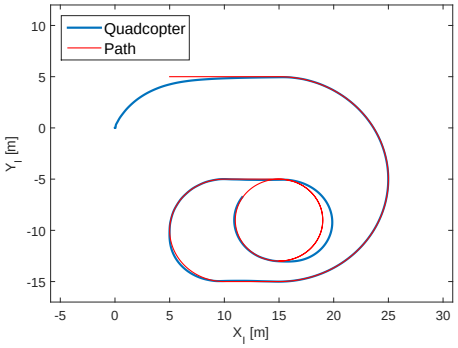
$$l_r = \frac{t_{rm}}{v_Q} \tag{3.88}$$

where t_{rm} is taken equal to 1.2 s, considering a 20% more of time than the sample time.

To test the law of parameters of Equations 3.86, the trajectory in Figure 3.22b, colored in red, has been considered. It is made up with 3 different curves, connected by straight lines, with three curvature radius, $R = 10$, 5 and 4 m. The velocity is $v_Q = 2$ m/s, so that r =0.2, 0.4 and 0.5 rad/s. The trajectory has been represented in blue. As one can see, the quadcopter is able to follow the path without any problems until it reaches the sections with high values of r. It must be said that the quadcopter goes slightly off from the path when the path changes suddenly its curvature, i.e. in the transitions between the circles and the straight lines; afterwards, it comes back on track and does not leave



(a) Change of the needed yaw rate r and the parameters value.



(b) Trajectory following, with changes in the curvature.

Figure 3.22: Example of following a trajectory with changes in the curvature, up to $r = 0.5$ rad/s, using the parameters law of Equations 3.22a.

it. In any case, we can hardly expect a better behavior of the controller than the one presented in Figure 3.19b, in which it was shown that the error starts increasing rapidly from values of r higher than 0.45 rad/s.

In Figure 3.23, a similar path has been generated by using spline functions, which means that the curvature along the path is a smooth function. Although along this path a maximum value of $r = 0.59$ rad/s is reached, the quadcopter is able to follow the path better than in the path of Figure 3.21. We can then conclude that if the paths generated have not abrupt changes in the curvature, the algorithm performance will be better.

In Figure 3.22a, the evolution along the time of the variable r and the parameters of the algorithm have been represented, when following the path of Figure 3.22b. Note the four levels of the required yaw rate r, corresponding to the straight lines and three different curvatures of the circles, and how the parameters adapt depending on its value.

3.6.1 Comparison of the two parameter's law

In Section 3.4, a fuzzy law to set the parameters of the algorithm was proposed and tested. It was based on a general knowledge about the functioning of the algorithm and the expertise acquired after some time working with it. It is a simple solution, easy and fast to develop and implement, and it turned out to be adequate for many paths.

Afterwards, a deep analysis of the kinematics of the movement and the yaw control was carried out, obtaining new and more accurate conclusions. A new and general law to set the parameters was proposed and tested, being the performance even better.

In Figure 3.24b, the trajectory of the quadcopter, following the same path as in Figure 3.23, has been shown. In blue, the general law, and in yellow, the

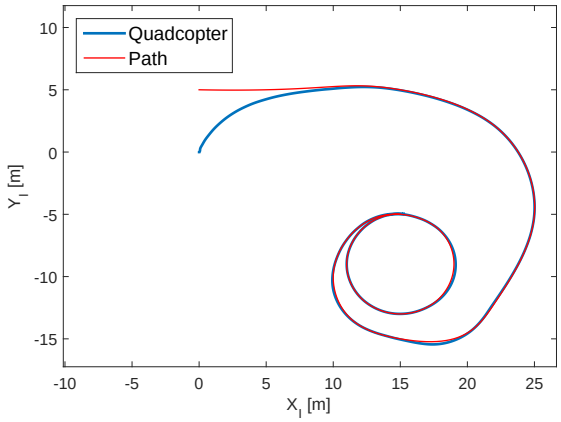


Figure 3.23: If the path is generated by spline functions, the path-following is more accurate, since the curvature along the path is smoother.

fuzzy law. To compare only the performance of the algorithm when following the desired path, and to omit the transient movement to approach the path, the quadcopter starts in the beginning of it, with an initial velocity already properly oriented. As it can be seen, both parameter's selection perform well, and only differences are visible when the curvature increases. In Figure 3.24a, the value of the parameters have been represented for both cases. Note that both laws set the parameters completely different, although at the end both are able to select the parameters so that the algorithm does well. The cross-track error of both laws are:

$$E_{general} = 3.8 \text{ m}^2$$
$$E_{fuzzy} = 7.7 \text{ m}^2$$

The error is double with the fuzzy law, although it is still good. With these two values, we can confirm what it seemed having a look at the trajectories: the general law better guides the quadcopter.

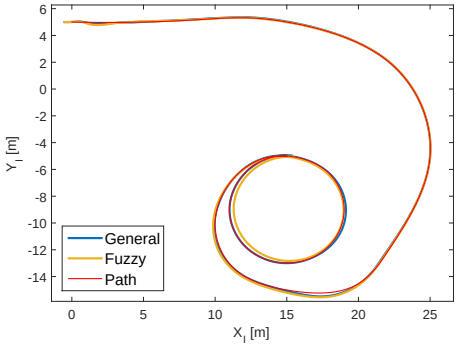
3.7 Modification of the algorithm

3.7.1 Creation of a virtual point

The result in Figure 3.2 shows that the algorithm works properly; the quadcopter converges rapidly and it keeps in track during the whole path. However, the algorithm tends to shorten the path if the quadcopter is located closer to middle points than to the starting point. This situation might no be desirable, since the paths will be given by the necessity of exploring a certain area.

To solve it, the algorithm may be modified so that the quadcopter directs to the starting point before starting following the path. We must keep in mind that once the quadcopter reaches the beginning of the path, it will need to turn to head the direction of the path at its beginning, i.e. vector $\mathbf{t}$. A virtual point P_{v_0} will be created, which is located in a circumference of radius R_{min} and tangent to the vector $\mathbf{t}$ at P_0 (beginning of the path). The vector $\overrightarrow{QP}_{v_0}$ is tangent to

(a) Comparison of the parameters' value.



(b) Comparison of the trajectories.

Figure 3.24: Comparison of the value of the parameters, using the general law presented in Section 3.6 and the fuzzy law, Section 3.4. In the right figure, the trajectories have been represented.

that circumference as well. In Figure 3.25a the geometry described has been represented. Note that there can be more than one solution, even 4 depending on how the problem is posed, and the proper one must be chosen. The minimum radius of curvature, R_{min}, which can be achieved by the quadcopter, is defined by:

$$R_{min} = \frac{v_Q}{r_{max}} \tag{3.90}$$

where r_{max} is the maximum yaw rotation velocity and it can be equal to 0.5 rad/s. Once the virtual point P_{v_0} has been calculated, the algorithm is used to head it by doing:

$$\mathbf{t} = \frac{\overrightarrow{QP}_{v_0}}{|\overrightarrow{QP}_{v_0}|} \tag{3.91a}$$

$$\mathbf{n_2} = (t_y, -t_x, 0) \tag{3.91b}$$

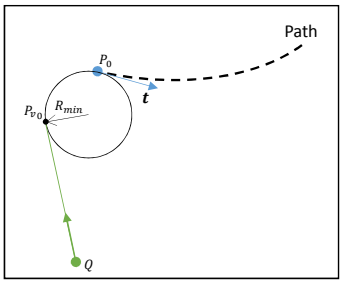
$$\mathbf{n_1} = (0, 0, 1) \tag{3.91c}$$

When the quadcopter gets closer than 1 m to the virtual starting point, P_{v_0}, the standard algorithm is used from then on. In Figure 3.25b it has been shown the trajectories of the quadcopter following a path, the original one and with the modification. The original algorithm shortens the path and reaches it 40 m from the starting point, while the modified one reaches it at its beginning, marked with a black circle.

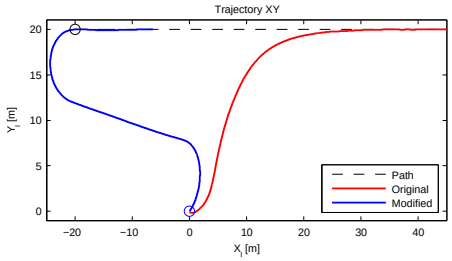
If the path to be followed has a curvature higher than the one that the quadcopter can follow, it will go off from it; this situation is expressed by the following condition at certain point:

$$R_{path} = \frac{1}{\rho} \tag{3.92a}$$

$$v_{Q_{max}} = r_{max} \cdot R_{path} \tag{3.92b}$$



(a) Virtual point P_{v_0} created to direct the quadcopter to the start of the path.



(b) Comparison between the trajectories with using the modified (blue) and the original (red) algorithm.

Figure 3.25: Modification of the algorithm to reach the beginning of the path.

where ρ is the first curvature of the path, see Equation 3.3, R_{path} is the radius of the path, r_{max} is the maximum yaw rotation rate and $v_{Q_{max}}$ is the maximum velocity of the quadcopter in order to be able to turn with that R_{path}.

3.7.2 Change in the parameters

Depending on the value of the parameters, when the quadcopter approaches the path, it might do it too aggressively. This leads to unnecessary oscillations until the quadcopter converges to the path. Recall that the parameters were chosen supposing that the quadcopter was close to the virtual point, so that those values might not be the best if there is a remarkable distance between then.

$$\text{if } d_{QP} > 1 \text{ m} \Rightarrow \begin{cases} k_{\hat{R}} = 2 \\ d = 3 \\ k_l = 0.15 \end{cases} \tag{3.93}$$

In Figure 3.26 the difference between the original configuration, i.e. with no parameter change when the quadcopter is far from the path, and Conf. 2, with the above rule. Note that if the change in the parameters are used, the approach is softer.

3.8 Operation modes

The algorithm already described needs a complete information of the path to be followed, i.e. discrete position of the points P along it, length l, basis vectors of the frame $\{\mathcal{F}\}$ (see Figure 3.1) and the two curvatures k_1 and k_2, see Equation 3.3. As it has been implemented, all the information is inside a single matrix,

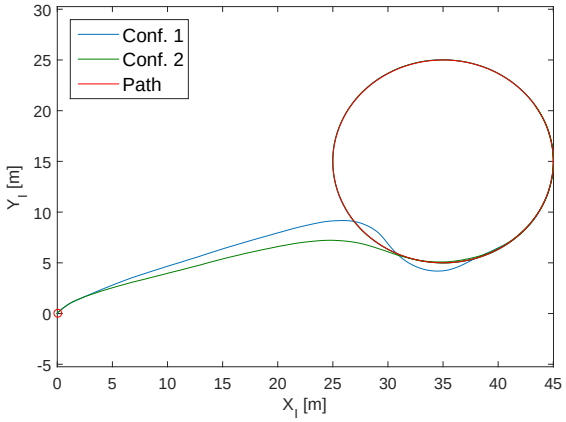


Figure 3.26: Differences in the trajectory when approaching the path. The Conf.2 reaches the path softer than Conf.1.

whose structure is:

$$path = \begin{bmatrix} P_{x_1} & P_{x_2} & ... & P_{x_n} \\ P_{y_1} & P_{y_2} & ... & P_{y_n} \\ P_{z_1} & P_{z_2} & ... & P_{z_n} \\ 0 & l_1 & ... & l_{n-1} \\ t_{x_1} & t_{x_2} & ... & t_{x_n} \\ t_{y_1} & t_{y_2} & ... & t_{y_n} \\ t_{z_1} & t_{z_2} & ... & t_{z_n} \\ n_{1_{x_1}} & n_{1_{x_2}} & ... & n_{1_{x_n}} \\ n_{1_{y_1}} & n_{1_{y_2}} & ... & n_{1_{y_n}} \\ n_{1_{z_1}} & n_{1_{z_2}} & ... & n_{1_{z_n}} \\ n_{2_{x_1}} & n_{2_{x_2}} & ... & n_{2_{x_n}} \\ n_{2_{y_1}} & n_{2_{y_2}} & ... & n_{2_{y_n}} \\ n_{2_{z_1}} & n_{2_{z_2}} & ... & n_{2_{z_n}} \\ k_{1_1} & k_{1_2} & ... & k_{1_n} \\ k_{2_1} & k_{2_2} & ... & k_{2_n} \end{bmatrix} \tag{3.94}$$

The first three rows correspond to the coordinates of the points P_i of the path in the $\{I\}$ frame; the forth row is the length l_i along the path, measured from the beginning to each point; the next three sets of three rows are the basis vectors of Frame $\{\mathcal{F}\}$, $\mathbf{t}_i$, $\mathbf{n_1}_i$ and $\mathbf{n_2}_i$, at each point P_i; finally, the last two rows are the two bishop curvatures.

Therefore, the matrix consists of with 15 rows and an undetermined number of columns, typically around 500. All this information must be sent to the quadcopter from a centralized controller, which might be an issue because of the big amount of data. To avoid this, different modes of operation have been considered so that the data transmitted is much lower. Another algorithm has been implemented to create the matrix given by 3.94 with the reduced information received.

3.8.1 Complete path

The information sent to the quadcopter is complete and ready to be used by the algorithm. This means that the whole matrix 3.94 is already generated. On one hand, the calculation load by the quadcopter is lower, since no extra computation is needed; on the other hand, the amount of data sent is larger. In this mode, the virtual point P_{v_0} (see Figure 3.25a) is activated, meaning that the quadcopter will go first to it before starting following the path. Once the end of the path is reached, the quadcopter will hold its position if $path_{4,1} = 0$, waiting for a new order.The path will be repeatedly follow if the element $path_{4,1} = 1$.

3.8.2 Only path points

In this mode, only the points P_i are sent (i.e. only the three first rows of matrix) and the path control algorithm must calculate the rest of terms of the path matrix:

$$path = \begin{bmatrix} 1 & P_{x_1} & P_{x_2} & ... & P_{x_n} \\ repeat & P_{y_1} & P_{y_2} & ... & P_{y_n} \\ 0 & P_{z_1} & P_{z_2} & ... & P_{z_n} \end{bmatrix} \tag{3.95}$$

The element $path_{1,1} = 1$ indicates to the algorithm that this mode is to be used and if $path_{2,1} = 1$ the path will be repeatedly followed. This mode lowers the transmission load, but requires extra computation on board to fill out the path matrix. With this mode, the algorithm does not add extra interpolating points, which does happen when using waypoints. In Figure 3.27a two paths have been represented; note that the quadcopter reaches the very beginning of the paths because of the activation of the virtual point P_{v_0}. The Path 1 is followed consinously until $t = 150$ s, when it is commanded to follow the second path. Once the quadcopter reaches the end of Path 2, it keeps the position.

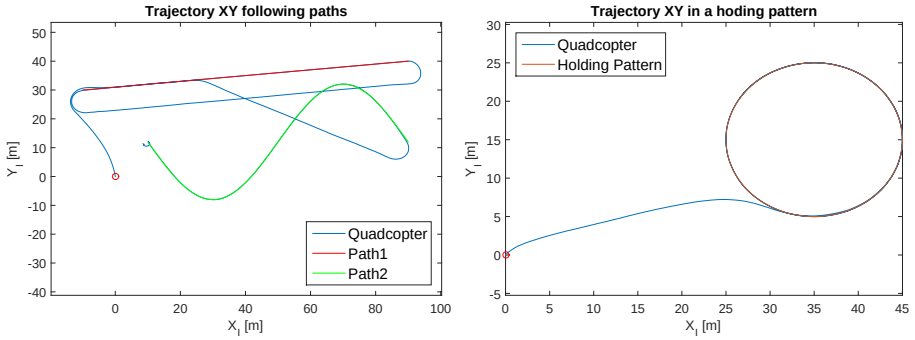
3.8.3 Holding pattern

In this case, the quadcopter is required to follow a circular trajectory repeatedly, a so called holding pattern. It is normally used to monitor a small area or to simply wait for a new command[2]. In this mode, the information needed is drastically reduced:

$$path = \begin{bmatrix} 2 & R & C_x & C_y & C_z \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.96}$$

The mode is activated if $path_{1,1} = 2$; R is the radius of the circumference and $C = [C_x, C_y, C_z]$ are the coordinates of its center. The last row of zeros is needed for memory reasons, since the path matrix is a variable-size matrix. In Figure 3.27b an example of trajectory approaching and following a holding pattern has been represented. Note that in this case the virtual point P_{v_0} is not used since it does not matter where the trajectory starts. The pattern can be followed indefinitely thanks to the algorithm, which restarts the path when the quadcopter is close to the end of it. The direction of spin around the circle depends on the initial position and velocity of the quadcopter so that it follows the pattern without doing unnecessary turns.

[2]In general, rotorcraft consume less energy when they fly at certain velocities than when simply hover. A holding pattern may be then also used for long waiting periods.



(a) Example of trajectories following paths.

(b) Trajectory following a circular holding pattern.

Figure 3.27: Different trajectories for two operational modes.

3.8.4 Waypoints

The coordinates of an undetermined number of waypoints are sent to the quadcopter, placed in the order to be followed. The information matrix for the mode is:

$$path = \begin{bmatrix} 3 & Wp_{x_1} & Wp_{x_2} & ... & Wp_{x_n} \\ interp. & Wp_{y_1} & Wp_{y_2} & ... & Wp_{y_n} \\ repeat & Wp_{z_1} & Wp_{z_2} & ... & Wp_{z_n} \end{bmatrix} \tag{3.97}$$

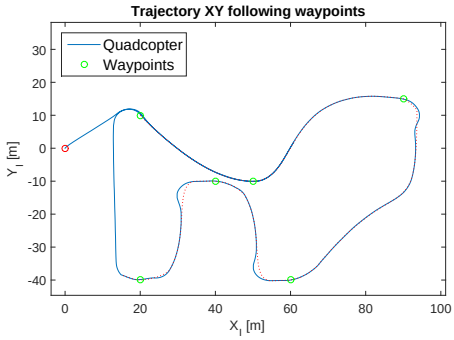
The difference with the only-path-points mode is that interpolating points between the waypoints will be created. The mode is activated when $path_{1,1} = 3$. The element *interp.* indicates which interpolation function must be used when creating the complete path (linear interpolations or piecewise cubic interpolation curves), and *repeat* means whether the path must be repeatedly followed. In Figure 3.28a an example of an UAV following waypoints has been represented; in this case, the complete path has been created using piecewise cubic interpolation curves passing through the waypoints (green circles) and it is repeatedly followed. In this mode, the virtual point is activated so that the first waypoint is also reached.

3.8.5 Heading

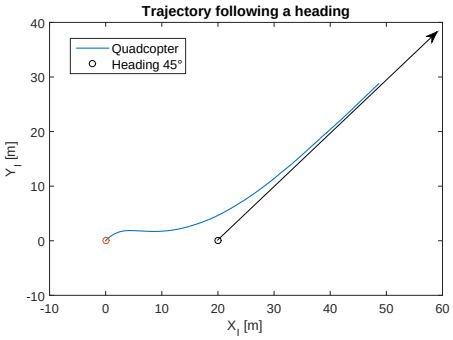
The last operating mode consists of flying in one direction, defined by the angle with the north direction ($\mathbf{e_1}|_{\mathcal{I}}$ or $\mathbf{e_2}|_{I}$), positive clockwise. The information sent is:

$$path = \begin{bmatrix} 4 & P_{0_x} & P_{0_y} \\ start & \Psi & h \end{bmatrix} \tag{3.98}$$

When $path_{1,1} = 4$, this mode is used. P_0 is the initial point of the heading, only used if $start = 1$. If this variable takes another value, the initial point will be taken as the current position of the quadcopter. Ψ is the heading angle and h is the desired altitude. The virtual point is not used in this mode and the configuration parameters are taken beside the law described at Section 3.6 to



(a) Example of trajectories following waypoints, generated by piecewise cubic interpolation curves.



(b) Trajectory heading 45° direction.

Figure 3.28: Different trajectories for the other two operational modes.

make the algorithm shorten the path to reach the heading trajectory:

$$d = 10$$
$$k_l = 0.15$$
$$k_R = 2$$

In Figure 3.28b an example of the heading mode has been shown. In this case, $\Psi = 45°$ and the starting point has been taken as $P_i = [20, 0]$.

Chapter 4

UAV ship-tracking algorithm

4.1 Introduction

The complete system (quadcopters, unmanned ships, contention barriers, etc.) will be boarded in a ship capable of carrying it to the proximity of the spill to be monitored. If a quadcopter needs to return to the ship, an algorithm must guide it, being known the present position, the velocity and the heading of the ship. This controller will only guide the aircraft to the proximity of the landing platform, since the GPS signal might not be accurate enough to carry out a safety landing on a small surface. When the quadcopter hovers over the landing platform, another controller might take the control to effectuate the actual landing, using more accurate sensors, such as a vertical camera and an ultrasonic or laser rangefinder.

This new mission, usually called target tracking, differs considerably from the previously studied path-following. The path is not known in advance, since the ship is allow to travel freely on the water; the point to which the quadcopter must converge to, called point P in the path-following algorithm, is now the ship and cannot be controlled.

In the literature, we can distinguish different algorithms based basically on three strategies or methods [6]:

- Line of sight (LOS): the algorithm tries to align the velocity vector of the UAV with the line of sight between the target (in our case, the ship) and another third point. It has been widely used to guide surface-air missiles [1], being the LOS defined between a radar located on the earth and the flying target; the missile tries to align with the LOS. Two examples of their use applied to autonomous system navigation are [9] and [3].

- Pure Pursuit (PP) [8]: the UAV is steered directly to the target, i.e. the velocity is controlled to be aligned with the vector between them. It has been used to guide the air-surface missiles [1]. Normally, this strategy leads to an excessive tail-chase (equivalent to a predator chasing a prey in the animal world). Note that in our case, the LOS and the PP strategy

are the same, since the LOS is defined between the quadcopter and the ship.

- Constant Bearing Guidance (CBG): the relative velocity between the UAV and the target is oriented to the LOS between both. This method, which has been used for the air-air missiles [1] and by the mariners during centuries, is normally implemented making the yaw velocity proportional to the yaw rate of the LOS [7]. Another possibility would be directly command the velocity so that the relative velocity is oriented to the target, and its magnitude depends on the relative distance.

As can be seen, all the methods were originally used in missile guidance and then adapted to civil applications. As it happens in the path-following algorithm, the tracking controllers are normally a high level ones, i.e. there exists a low level controller which is in charge of stabilizing the dynamics of the system and of making the UAV have a certain velocity. Again, it is necessary that both controllers are decoupled and to ensure that the time constant of the high level controller is typically 2 times longer that the low level one.

To tackle the ship-following mission, we will choose the third method above presented, the constant bearing guidance approach. The performance of this method has been proved for instance in [5] and [6]. In this last paper, the velocity reference was defined by:

$$\mathbf{v}_c(t) = \mathbf{v}_S(t) + \mathbf{v}_a(t) \tag{4.1}$$

where $\mathbf{v}_S$ is the velocity vector of the target and $\mathbf{v}_a$ is the relative velocity between the target and the UAV or, in other words, the velocity with which the UAV approaches the target. This velocity is defined as:

$$\mathbf{v}_a = K \frac{\mathbf{r}_S}{|\mathbf{r}_S|} \tag{4.2}$$

The relative velocity has the direction of the relative position vector, $\mathbf{r}_S$, and to be proportional to a factor K, whose value depends on the relative distance:

$$K(t) = U_{a_{max}} \cdot \frac{|\mathbf{r}_S|}{\sqrt{|\mathbf{r}_S|^2 + \Delta_p^2}} \tag{4.3}$$

being $U_{a_{max}}$ the maximum relative speed and Δ_p a factor which controls the transient interceptor-target behavior. The maximum relative speed is finally defined by the law:

$$U_{a_{max}} = |\mathbf{v}_S| \cdot \zeta \tag{4.4}$$

where ζ is a positive factor lower than 1. A similar algorithm will be proposed in this work and they both will be compared.

It must be pointed out that, in the early beginning of this work, willing to use the path-following algorithm presented in Chapter 3 due to its good results, it was modified and tested to fulfill this new tracking task. However, after several modifications and tests, it was clear that the algorithm was not appropriate for this purpose and was discarded. For example, if the ship is not moving, the algorithm is unable to converge if the quadcopter is not located behind the ship.

4.2 Ship-following algorithm

4.2.1 Velocity control

It is assumed that a low level controller is already implemented, which stabilizes the quadcopter and leads it so that its velocity converges to the reference one. This is the same assumption that it was made for the path-following algorithm.

We first define the relative position vector:

$$\mathbf{r}_S = \mathbf{S} - \mathbf{Q} \tag{4.5}$$

where $\mathbf{S} = [S_x, S_y]$ is the position of the ship and $\mathbf{Q} = [Q_x, Q_y]$ the position of the quadcopter, both in the $\{I\}$ frame, and $\mathbf{r}_S$ is the vector between the quadcopter and the ship, that is, the error in the position of the quadcopter with respect to the ship. We propose a PID controller to control the velocity of the quadcopter in both coordinates:

$$\mathbf{v}_c = \mathbf{v}_S + k_P \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} = \mathbf{v}_S + \mathbf{v}_b \tag{4.6}$$

being

$$\mathbf{v}_b = k_P \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} \tag{4.7}$$

In every moment, the position of the ship, its velocity and its heading will be assumed as known (either this information is transmitted from the ship or by the on-board sensors when the quadcopter is close enough). Note that this algorithm is also a constant bearing one, being the relative position between the quadcopter and the ship controlled by a PID controller.

Since the magnitude of the velocity is an input of the controller here exposed, Equation 4.6 will be finally bounded as following:

$$\mathbf{v}_c|_{bounded} = \begin{cases} v_Q \dfrac{\mathbf{v}_c}{|\mathbf{v}_c|} & \text{if } |\mathbf{v}_c| > v_Q \\ \mathbf{v}_c & \text{if } |\mathbf{v}_c| \leq v_Q \end{cases} \tag{4.8}$$

This way the direction of the velocity is kept and its magnitude adjusted to the required one, v_Q, if $|\mathbf{v}_c| > v_Q$.

Since the integral term of the controller accumulates the distance between the quadcopter and the ship, if they both are initially far away one from the other, by the time the quadcopter reaches the ship, this term might be very high. This would imply an unnecessary overshooting and a longer time to converge. To limit its influence, the parameter k_I will be taken as zero when the quadcopter is further than 2 m from the ship:

$$k_I = \begin{cases} k_I & \text{if } |\mathbf{r}_s| \leq 2 \text{ m} \\ 0 & \text{if } |\mathbf{r}_s| > 2 \text{ m} \end{cases} \tag{4.9}$$

4.2.2 Altitude control

The altitude of the quadcopter must be controlled as well. The next law has been chosen:

$$h_c = \begin{cases} 10 & \text{if} \quad d_{QS} > 10 \\ d_{QS} & \text{if} \quad 10 \geq d_{QS} > 2 \\ 2 & \text{if} \quad d_{QS} < 2 \end{cases} \tag{4.10}$$

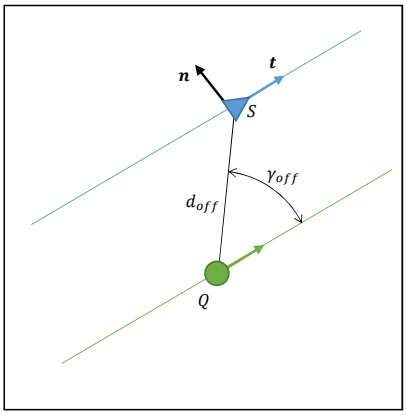


Figure 4.1: Fly in formation mode, defined by the parameters γ_{off} and d_{off}.

.

where $d_{QS} = \sqrt{(S_x - Q_x)^2 + (S_y - Q_y)^2}$ is the 2-dimensional distance between the ship and the quadcopter. If the quadcopter is far from the quadcopter, its altitude is kept constant at a sufficient altitude from the sea level. When it gets closer than 10 m, it starts descending linearly with the distance to the ship until a minimum altitude is reached, equal to 2 m.

4.2.3 Fly in formation mode

Beside returning to the ship and land on it, this operational mode is also required. The quadcopter will follow the ship movement at a certain position, defined by the line of sight γ_{off} and the distance d_{off}, see Figure 4.1. A point $\mathbf{S_{off}}$ is then defined as:

$$\mathbf{S}_{off} = \mathbf{S} - d_{off} \cdot (\mathbf{t} \cdot cos(\gamma_{off}) + \mathbf{n} \cdot sin(\gamma_{off})) \tag{4.11}$$

Then, the quadcopter will converge to this point and follow it. In this operational mode, the altitude will not be controlled by Equation 4.10, but kept constant and equal to a nominal value sent from the centralized controller.

4.3 Comparison of controllers

If we compare the controller proposed in Equation (4.6) with the controller presented in [5] and [6], Equations (4.1), (4.2) and (4.3), we notice that they both have the first term $\mathbf{v}_S$ in common in both controllers, but the second one differs:

$$\mathbf{v}_a = |\mathbf{v}_S| \cdot \zeta \cdot \frac{\mathbf{r}_S}{\sqrt{|\mathbf{r}_S|^2 + \Delta_p^2}} \tag{4.12a}$$

$$\mathbf{v}_b = k_p \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} \tag{4.12b}$$

Having a look at both expression, one can find two main differences:

- The controller here proposed has an integer and a derivative term.
- In the Breivik et al. controller, the relative velocity $\mathbf{v}_a$ is directly oriented to the target, but its modulus is the same order of magnitude than the velocity of the target itself. In the controller here proposed, before limiting the velocity by Equations (4.8), the order of magnitude of the relative position vector, $\mathbf{r}_S$, will be higher than the ship velocity if the quadcopter and the ship are far enough, so the predominant term will be this one.

To illustrate and analyze this second point, let's assume this situation:

$$\mathbf{Q} = [0, 0],\ \mathbf{S} = [100, 0] \Rightarrow \mathbf{r}_S = [100, 0] \tag{4.13a}$$

$$\mathbf{v}_S = [0, 1] \tag{4.13b}$$

and the next configuration for both controllers:

$$k_P = 1,\ k_I = 0.01,\ k_D = 0 \tag{4.14a}$$

$$\zeta = 0.5,\ \Delta_p = 1 \tag{4.14b}$$

The previous values are standard and have not been optimized. For the sake of this comparison, their values are not important, since we only want to compare the global response of the controllers by having a look at their trajectories, without taking into consideration any performance measurement in detail.

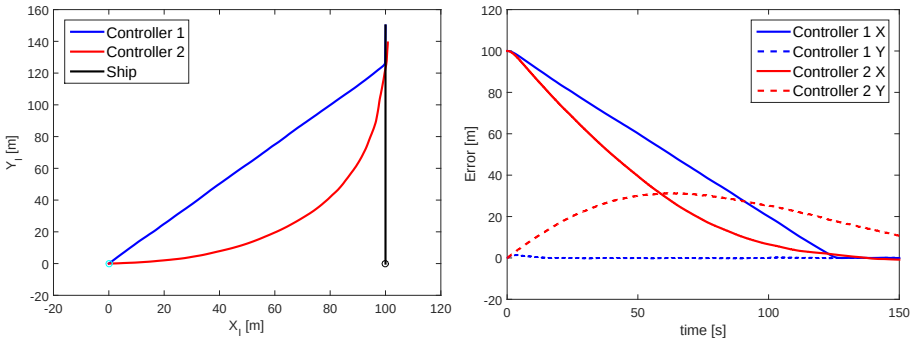
Naming the controller developed in [5] and [6] as *controller 1*, and the one shown here as *controller 2*, the value of the velocity for each controller at $t = 0$ s will be:

$$\mathbf{v}_c(t = 0)|_1 \simeq [0.8, 1] \tag{4.15a}$$

$$\mathbf{v}_c(t = 0)|_2 \simeq [1.3, 0] \tag{4.15b}$$

For the second controller, the modulus of the velocity has been chosen so that both have the same modulus. With the above result one can realize what the main difference between both is: controller 2 heads the quadcopter directly to the ship when both are far from each other, while the controller 1 only directs the relative velocity, independently of how big the distance between them is. In Figure 4.2a the trajectory of the quadcopter pursuing the ship (black line) using both controllers has been represented, when the both controllers set the same velocity modulus. The controller 1 guides the quadcopter straight to the future position of the ship, while the controller 2 pursues too much the present position of the ship in every moment. In the Figure 4.2b the distance between the ship and the quadcopter, in both coordinates X_I and Y_I, has been shown. Here we can also see that the controller 1 converges to the ship faster than the controller 2.

This characteristic of the controller 1 is a clear advantage over the controller 2, which will pursue directly the ship when the distance is big, making the quadcopter travel along longer distances. When the quadcopter is closer to the ship, both controllers will behave in a similar way. We will try now to modify the controller 2 to improve this behavior.



(a) Trajectories following the ship with both controllers.

(b) Distance from the quadcopter to the ship.

Figure 4.2: Example of the quadcopter tracking the ship with both controllers (left figure) and the distance to the ship in both coordinates (right figure). The modulus of the velocity is similar for both controllers.

4.4 Modification of the algorithm

As we have just seen in Figures 4.2a and 4.2b, in the case the ship is far from the quadcopter, it will always try to reach the ship position at every moment. Since it may take some time until the ship is reached, the ship position will have already changed, resulting in that the quadcopter goes along a path longer than it should. To try to shorten the length of the path, instead of using the point $\mathbf{S}$, a virtual point $\mathbf{S}'$ will be used, defined as the foreseen position in which the ship and the quadcopter would meet if they kept their velocities and headings. In Figure 4.3 the virtual point S' has been represented, which has to be calculated firstly estimating ρ, time to reach the ship in a straight line. To calculate ρ, the next system equation must be solved:

$$x^2 + y^2 - v_Q^2 \cdot \rho^2 = 0 \tag{4.16a}$$

$$x = r_{S_x} + v_S \cdot \rho \cdot t_x \tag{4.16b}$$

$$y = r_{S_y} + v_S \cdot \rho \cdot t_y \tag{4.16c}$$

where $\mathbf{t} = (t_x, t_y)$ is the direction of the ship movement. Equations 4.16 lead to a second order polynomial which gives two results for ρ, one positive and another negative (to be dismissed), or two positive, in which case the lowest will be taken. Once ρ has been determined, the foresee position can be calculated as:

$$\mathbf{S}' = \mathbf{S} + d_S \cdot \mathbf{t} \tag{4.17}$$

where $d_S = v_S \cdot \rho$. S' will then be replaced with S in Equations 4.5 and 4.6. The use of the virtual point S' will be deactivated when the quadcopter is closer than 5 meters to the ship.

In Figure 4.4a the trajectory of both controllers has been represented, the controller 1 and the modified controller 2, named as *controller 2.2*. Both trajectories are now much more similar; they lead the quadcopter directly to the future position of the ship, which is basically the shortest path to reach it, given

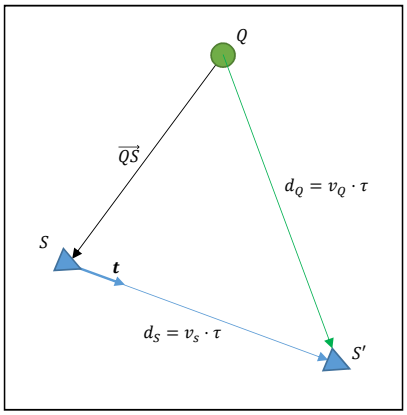


Figure 4.3: Modification of the distance d in the algorithm to anticipate the motion of the ship.

.

a maximum velocity modulus. In Figure 4.4b the distance in both coordinates has been presented, showing again how similar both trajectories are.

Having checked that both controller's performances are very similar, one might think that there is no point of using controller 2.2 instead of controller 1, which is easier to implement and will probably require less computation costs.

However, the controller 1 specifies the relative velocity $\mathbf{v}_a$ between the ship and the quadcopter through Equation (4.12a). This means that whatever the ship velocity is, the quadcopter will approach the ship with a previously configured velocity. The total absolute velocity will be then the vector addition of both velocities:

$$\mathbf{v}_Q = \mathbf{v}_S + \mathbf{v}_a \tag{4.18}$$

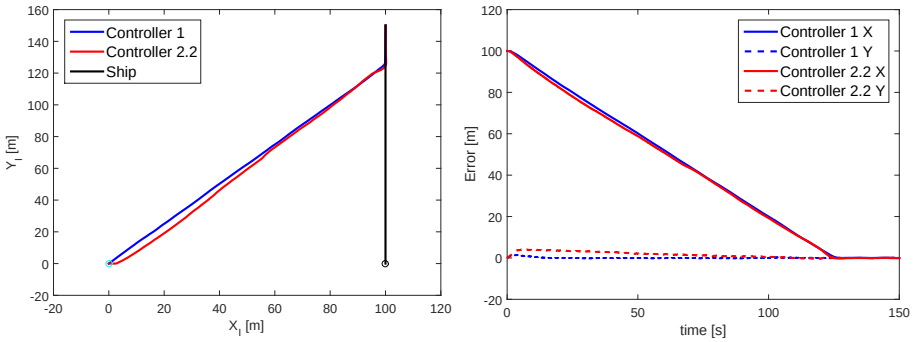
Quadcopters have an absolute velocity limitation because of the aerodynamic forces on it. The above result shows that if both velocities have opposite direction, the absolute velocity will be lower and, in fact, the maximum velocity of the quadcopter will not be used and the approach will be slower.

To illustrate this reasoning let's simulate both controllers, with the parameters previously chosen, but this time the initial position of the quadcopter will be $\mathbf{Q}_0 = [50, 80]$. The maximum velocity for the controller 2.2 will be chosen as previously, $v_Q = 1.3$ m/s. In Figure 4.5a both trajectories have been represented. As can be seen, the controller 2.2 leads the quadcopter much faster to the ship, cutting short the ship path, while with the controller 1 the quadcopter converges to the ship traveling in the direction of its movement. In Figure 4.5b the distance between the quadcoter and the ship has been represented for each coordinate.

Initially the velocity calculated by controller 1 is:

$$\mathbf{v}_c|_1 = \mathbf{v}_S + \mathbf{v}_a = [0, 1] + 0.8 \cdot 1 \cdot \frac{[50, -80]}{94.3} = [0.42, 0.32] \tag{4.19}$$

Notice that this vector is pointed firstly to the positive direction of the Y_I axis, while the ship is located in the other direction. Secondly, its modulus



(a) Trajectories following the ship with both controllers.

(b) Distance from the quadcopter to the ship.

Figure 4.4: Example of the ship tracking the ship with controller 1 and the modified controller 2, and the distance to the ship in both coordinates. The modulus of the velocity is similar for both controllers.

is only $|\mathbf{v}_c| = 0.53$ m/s, lower than the velocity we had when the quadcopter initially was located in the origin. Indeed, what is happening is that since the ship is heading approximately the initial position of the quadcopter, the controller 1 fixes the relative velocity and, therefore, the total velocity turns out to be much lower. In other words, in case the ship is traveling in the direction of the quadcopter, the controller 1 tends to make the quadcopter wait for the ship to come, instead of heading the aircraft to it.

The controller 2.2 uses the absolute velocity possible (or externally indicated) to directly lead the quadcopter to the position in which the ship is expected. As it can be seen in Figure 4.5b, the error in both coordinates converges linearly as fast as the velocity modulus restriction allows it, being in this case the most optimum path possible. Another advantage of the controller 2.2 is that there is not an extra parameter, ζ, to be configured for the far-field approximation, which ease its implementation.

4.5 Determination of the parameters

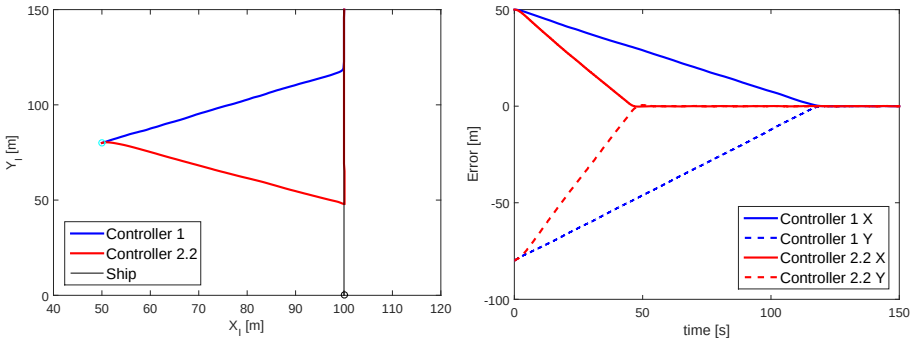
As have been shown, the modified controller 2 performs successfully when approaching the ship. Now, the study of the values of the parameters must be carried out. Recall that the control Equations 4.6 and 4.8 to steer the quadcopter were:

$$\mathbf{v}_c = \mathbf{v}_S + k_P \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} \tag{4.20a}$$

$$\mathbf{v}_c|_{bounded} = \begin{cases} v_Q \dfrac{\mathbf{v}_c}{|\mathbf{v}_c|} & \text{if } |\mathbf{v}_c| > v_Q \\ \mathbf{v}_c & \text{if } |\mathbf{v}_c| \leq v_Q \end{cases} \tag{4.20b}$$

The velocity $\mathbf{v}_c$ is the input commanded to the low level controller, which must ensure that it is achieved successfully.

To simplify the above equations, we will first assume that the quadcopter is close enough to the ship, so that the terms proportional to $\mathbf{r}_S$ and $\mathbf{v}_S$ are small



(a) Trajectories of the quadcopter tracking the ship with both controllers.

(b) Distance from the quadcopter to the ship.

Figure 4.5: Example of ship following with controller 1 and the modified controller 2, and the distance to the ship in both coordinates, when the ship is heading approximately the initial position of the quadcopter.

enough and $|\mathbf{v}_c| \leq v_Q$ is satisfied. This implies that the velocity will not be bounded. Remember also that the integral term of the controller will be zero if the distance $|\mathbf{r}_S| > 2$ m, see Equation 4.9. Moreover, let's suppose that the high level controller is again much faster that the low level one, as it was also supposed when developing the path-following controller in Chapter 3, so that we can assume:

$$\mathbf{v}_Q \simeq \mathbf{v}_c \tag{4.21}$$

which means that the absolute velocity of the quadcopter is instantaneously lead to the reference one, and therefore:

$$\mathbf{v}_Q \simeq \mathbf{v}_S + k_p \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} \tag{4.22}$$

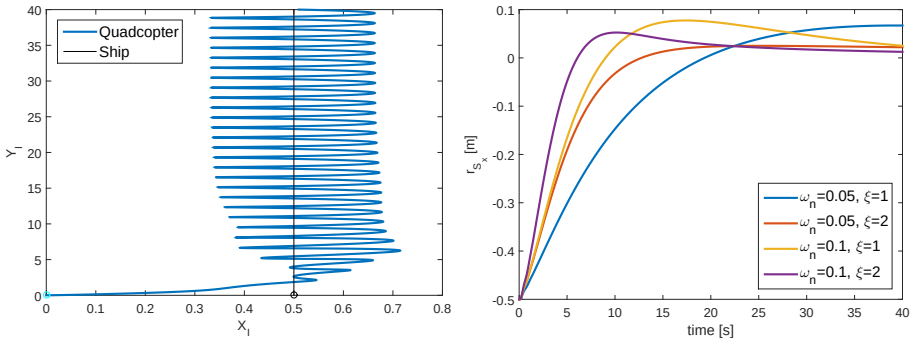
Note that in this case, we have another variable which was not present previously, $\mathbf{v}_S$, on which the commanded velocity directly depends. Therefore, if there were fast changes in $\mathbf{v}_S$, we could not be able to ensure that both controllers are decoupled. Luckily, the dynamics of a ship are much slower than the quadcopter's ones because of the nature of this system. The ship moves on a viscous liquid, has a big mass and is not able to shift laterally. Knowing this, we can ensure that the changes in its velocity will be negligible compared to changes in $\mathbf{r}_S$. Then, the equation that describes the kinematics of the motion is:

$$\mathbf{v}_Q = \mathbf{v}_S + k_p \cdot \mathbf{r}_S + k_I \cdot \int_0^t \mathbf{r}_S dt + k_D \cdot \frac{d\mathbf{r}_S}{dt} \tag{4.23}$$

Knowing that $\dot{\mathbf{r}}_S = \mathbf{v}_S - \mathbf{v}_Q$, and deriving the previous equation, we finally get:

$$(1 + k_D) \cdot \ddot{\mathbf{r}}_S + k_p \cdot \dot{\mathbf{r}}_S + k_I \cdot \mathbf{r}_S = 0 \tag{4.24}$$

We must point out that the consideration taken, Equation 4.21, to obtain the kinematics equation, might not be always fulfilled. For instance, let's assume



(a) Trajectory with $k_D = -1$, $k_P = 2$ and $k_I = 0.1$.

(b) X_I coordinate trajectories with $k_D = 0$.

Figure 4.6: Examples showing the discrepancies between Equation 4.24 and the system response.

that we would like to reduce the order of the Equation 4.24 by assigning $k_D = -1$, obtaining:

$$k_p \cdot \dot{\mathbf{r}}_S + k_I \cdot \mathbf{r}_S = 0 \tag{4.25}$$

Theoretically, the system response would be a first order one, converging to the ship asymptotically; however the system presents a complete different behavior, presented in Figure 4.6a. In that example it has been taken $k_P = 2$ and $k_I = 0.1$ and therefore, the response should an exponential function with a time constant $\tau = k_P/k_I = 20$ s; however, the response does not converge and oscillates without damping.

This is not the only discrepancy between the simulations and Equations 4.24. Let's suppose that we take $k_D = 0$; Equation 4.24 becomes:

$$\ddot{\mathbf{r}}_S + k_p \cdot \dot{\mathbf{r}}_S + k_I \cdot \mathbf{r}_S = 0 \tag{4.26}$$

The above equation is a second order system, whose natural frequency and damping ratio are:

$$\omega_n^2 = k_I \tag{4.27a}$$
$$2\xi\omega_n = k_P \tag{4.27b}$$

In Figure 4.6a, the X_I coordinate of 4 trajectories with different values of ω_n and ξ have been shown. Although the four configurations have been chosen so that $\xi > 1$, the system responses are underdamped; on the other hand, the system is more damped when ξ increases and the response is faster if ω_n increases, as expected. Then, we can conclude that although the system does not behave as a second order system, it does present some of its characteristics.

As it was also done in the path-following algorithm, we must identify a time constant for which the high and low level controller start coupling. The time constant of the high level controller, τ_h must be higher enough than the low level one, τ_l:

$$\tau_h = \frac{1}{\xi\omega_n} \gg \tau_l \tag{4.28}$$

where τ_l and τ_h are the time constants of the low and high level controllers, respectively.

To tune the values of the PID controller, we will firstly assume $k_D = 0$ and choose the best values of k_P and k_I. Afterwards, we will choose k_D to fine tune the system controller.

The performance of the system response will be measured by two variables, the ship-following error and the controller effort, defined as follows:

$$E_S = \int_0^t |\mathbf{r}_S| dt \tag{4.29a}$$

$$\sigma_S = \int_0^t |r| dt \tag{4.29b}$$

being r the yaw rate, which can be calculated by:

$$r = \frac{d\gamma}{dt} \tag{4.30a}$$

$$\gamma = \arctan\left(\frac{v_{cx}}{v_{cy}}\right) \tag{4.30b}$$

To check the performance of the algorithm with the above expressions, the test shown in Figure 4.7a will be carried out. The quadcopter starts at position $[0, 0]$ and the ship at $[2, 0]$ and moves in the y_I direction at 1 m/s. To better see the differences, depending on the parameters' value, only a distance between them of 2 m is considered; higher distances implied less differences and make difficult the analysis. The quadcopter flies with a nominal velocity of 2 m/s and suffers a perturbation that tries to move it off the ship trajectory.

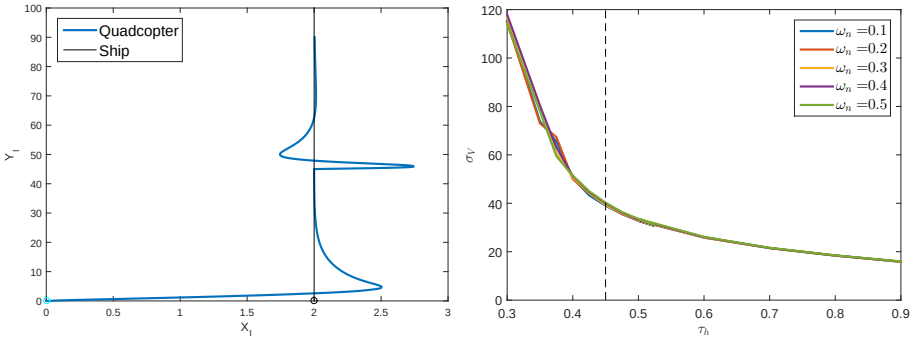
We must first identify for which time constants of the high level controller, both level controller start coupling. To do that, we will simulate the system, swapping a range of values of τ_h, with different values of ω_n, being then the parameter ξ determined by Equation 4.28. In Figure 4.7b, the effort has been represented depending on the time constant τ_h and for different values of ω_n. As it can be seen, all the curves are approximately constant for high values of τ_h and at $\tau_h \simeq 0.4$ s the effort increases rapidly. This might be assumed as the time constant in which both controllers start coupling, although as it happened in the path-following algorithm, it would have been expected for values around $2 \cdot \tau_l$. Note that in this figure, the time constant $\tau_l \simeq 0.45$ s has also been represented with a dashed line. We must point out that the difference in the effort when the values of ω_n change, is small.

Although from Figure 4.7b we identify the coupling frontier at $\tau_h \simeq 0.4$, taking a look at the trajectories of the quadcopter, one realizes that there exist oscillations up to $\tau_h \simeq 0.5$. This oscillations were not identify using Equation 4.29b because they appear only in the peaks of the trajectory and they are small. We can then select as the coupling frontier:

$$\tau_h = 0.5 \text{ s} \tag{4.31}$$

which is actually the value we obtained analyzing the path-following algorithm, see Figure 3.12.

Knowing now that the minimum time constant, at which the high and low controllers do not couple, is $\tau_h \geq 0.5$ s, we can represent the error defined by



(a) Example of trajectory following the ship, with a perturbation.

(b) Effort for different ω_n values and depending on τ_h.

Figure 4.7: Example of ship following trajectory; the quadcopter suffers a perturbation in the direction of Y_I. Effort of the controller, defined by Equation 4.29b, depending of τ_h and for different ω_n values; the time constant of the low level controller has been represented with a dotted line.

Equation 4.29a for different values of τ_h and ω_n, see Figure 4.8a. The error decreases when ω_n decreases, coming to a halt for values lower than 0.7 s approximately. Its dependency with τ_h is not clear, but anyway is not big either. In Figure 4.8b the effort has been shown, which increases when the time constant τ_h decreases, and is practically constant with the natural frequency ω_n. Note that between $\tau_h = 0.5$ s and $\tau_h = 0.6$ s the difference in the effort is higher than between $\tau_h = 0.6$ s and $\tau_h = 0.7$ s, meaning that between both there might be still some coupling. We can conclude saying that the error depends basically on the natural frequency ω_n, and the effort on the time constant τ_h.

Having seen the results, we will finally choose:

$$\tau_h = 0.7 \text{ s} \tag{4.32a}$$

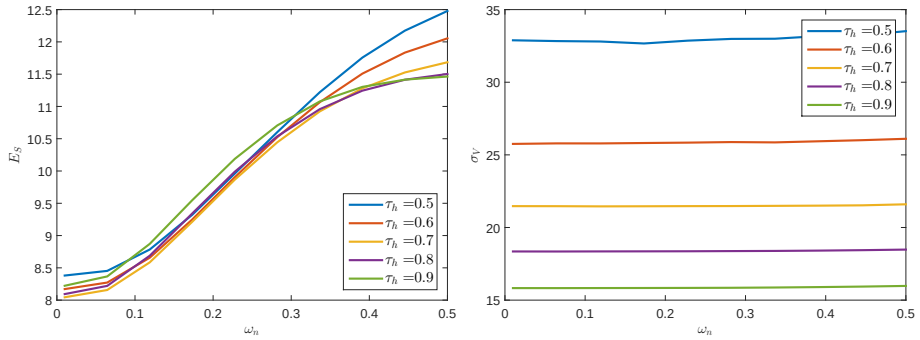
$$\omega_n = 0.05 \text{ s}^{-1} \Rightarrow \xi = 2.86 \tag{4.32b}$$

These values show the lowest cross-track error and medium value of effort. The parameters of the controller are then:

$$k_I = 0.0025 \text{ s}^{-2} \tag{4.33a}$$

$$k_P = 2.86 \text{ s}^{-1} \tag{4.33b}$$

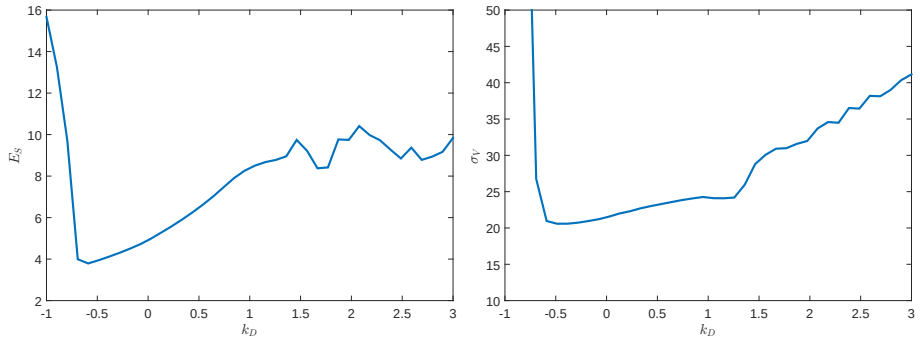
Now that these two parameters have been chosen, we can study whether a change in k_D can improve the performance of the control. In the same way that it was previously done, we represent the cross-track error (Figure 4.9a) and the effort (Figure 4.9b). As it can be seen, for lower values of k_D than -0.5, the error and the effort increase abruptly, indicating the coupling between both controllers. Also, for higher values than 1.3, the error becomes unstable and the effort starts increasing faster; observing the system response, one can see that again the response oscillates for those values, meaning that there is coupling as well. We could set k_D with values close to -0.5, since the cross-track error and the effort would be lower; however, choosing a higher value, but still in the



(a) Error tracking the ship.

(b) Effort tracking the ship.

Figure 4.8: Error (left figure) and effort (right figure) of the controller tracking the movement of the ship, depending on the time constant τ_h and the natural frequency ω_n.



(a) Error tracking the ship.

(b) Effort tracking the ship.

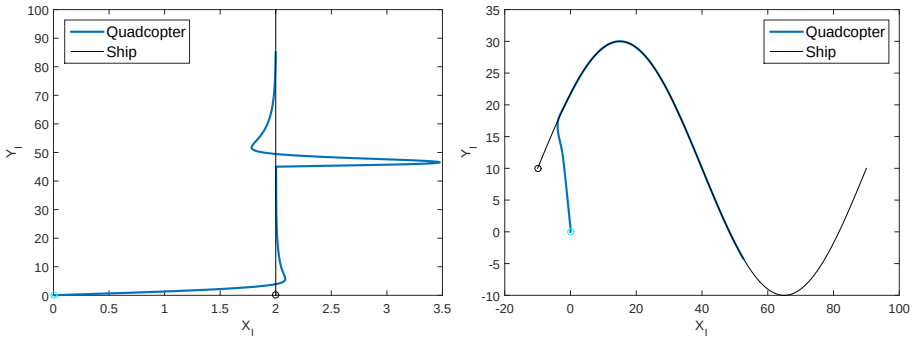
Figure 4.9: Error and effort of the controller tracking the movement of the ship, depending on the parameter k_D, with k_P =2.86 and k_I =0.0025.

non-coupling range, the performance of the algorithm would be also good and the approximation to the ship would be smoother. This last characteristic is specially visible if the quadcopter reaches the ship from its bow, i.e. when they have contrary velocity directions. Having said that, k_D will be set to:

$$k_D = 1 \tag{4.34}$$

In Figure 4.10a, the same test previously carried out has been represented. The trajectory presents an overshooting of 2% approximately and there is no oscillations due to the coupling between the high and the low level controllers. Moreover, in Figure 4.10b another ship trajectory has been tested. The ship travels along a sinusoidal curve at 1 m/s, being the quadcopter perfectly able to track it.

Finally, we must point out that the characteristics here taken into account to tune the controller have been the cross-track error and the effort. This lead to aggressive approaches to the ship and might not be wanted. If this were the case, one could just increases the value of τ_h, for example up to 5 s, and then



(a) Linear trajectory with a perpendicular perturbation.

(b) Sinusoidal trajectory without perturbation.

Figure 4.10: Examples of the quadcopter tracking the ship with the parameters chosen, $k_P = 2.86$, $k_I = 0.0025$ and $k_D = 1$.

tune again k_D, probably decreasing it to a value close to $k'_D = k_D \cdot \tau_h/\tau'_h = 1 \cdot 0.7/5 = 0.14$. The response would be then slower and smoother.

Chapter 5

Conclusions

5.1 Complete mission

In the Figure 5.1 a complete mission has been simulated. Initially, the quadcopter (red circle) is landed on the ship (black circle) and both are located at the position $[0, 0]$. The ship moves firstly to the south and when it reaches the position $[-150, 0]$ it turns to the East. It velocity is always 0.5 m/s and its trajectory has been represented in black. Close to the initial position, the position of the ship (black circle) and the quadcopter (red circle) has been represented after 20 s of mission.

The first command to the quadcopter is to follow a 90°heading, which starts at position $[0, 60]$, represented with a doted line. After 70 s, the quadcopter is ordered to follow a sinusoidal path, in red. At $t = 175$ s, it starts following a set of waypoints, marked with green circles; the path, which connects the waypoings, is generated by piecewise cubic functions. The waypoints are repeatedly followed. At $t = 425$ s, it flies to a holding pattern, with a radius of 20 m and centered in the point $[30, -150]$ (cyan colored). At $t = 570$ s, the quadcopter is commanded to fly to the ship, flying in formation mode with $d_{off} = 20$ m and $\gamma_{off} = -\pi/4$. At $t = 615$ s, the position of the quadcopter and the ship has been represented, with a red-filled and black-filled circles, respectively (S - *formation* and Q - *formation* in the legend). One can see that indeed the quadcopter is flying in formation with those values of d_{off} and γ_{off}. At $t = 640$ s, the quadcopter finally flies to land on the ship. Two more pair of circles, quadcopter and ship, have been drawn to show the convergence of the UAV to the boat. Note that when the quadcopter flies to the ship, or to the formation position, it does it following a straight line. Also, the transition to follow the path, the waypoints and the holding pattern, are straight lines.

5.2 Conclusions

In this master thesis an algorithm, originally developed for fixed-wing UAVs, has been analyzed and adapted so that it can be used in rotorcraft, and more specifically in quadcopters. The algorithm has three parameters, in the original work set as constant. As a first approximation, the algorithm was improved by setting its parameters with a fuzzy law depending on the velocity and the path

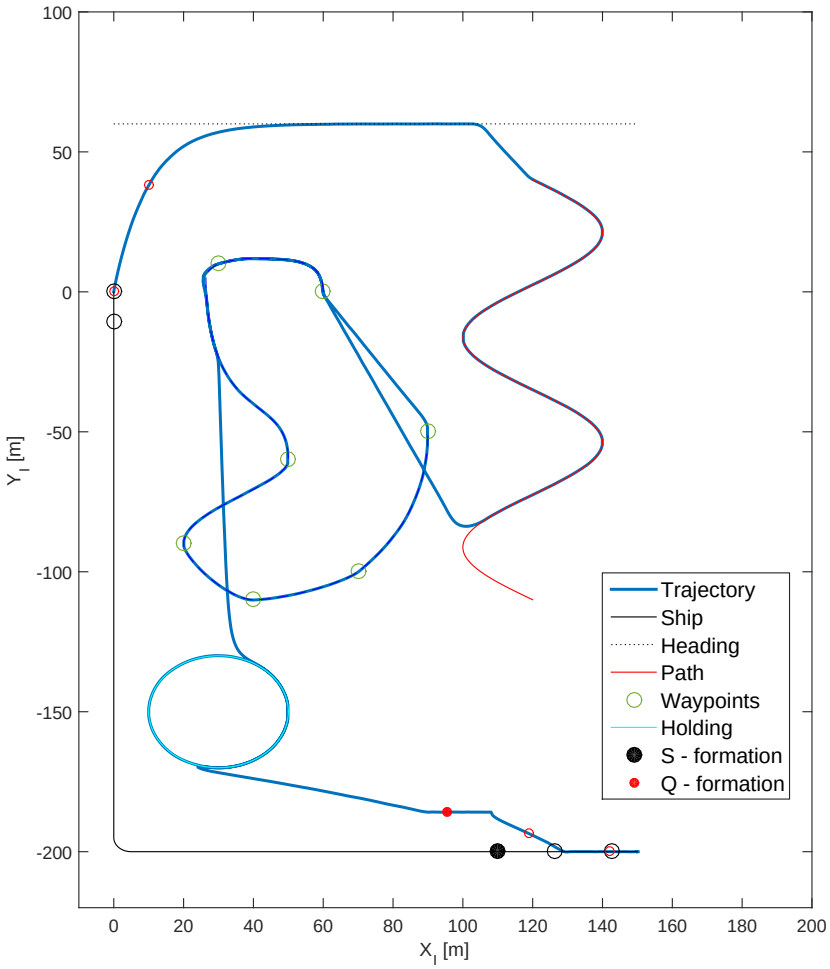


Figure 5.1: Trajectory of the quadcopter during a complete mission example.

curvature. Further, a deep analysis of the kinematics of the movement of the quadcopter was carried out. Linear and circular trajectories were considered, testing the robustness of the algorithm by applying an external perturbation and measuring the performance with the cross-track error and the required effort. The conclusions leaded to the proposal of a general law, which applies for all types of trajectories. This general law was compared with the fuzzy one and it was shown that, despite both were good solutions, the general law achieves better results. Also, different working modes have been implemented, such as holding a position, path generated by waypoints and fly with a determined heading.

A tracking algorithm was also needed to lead the quadcopter to the ship and land on it. The main purpose of this algorithm was to control the convergence until a close position; in that moment, another algorithm, which uses more accurate positioning equipment, would take the control. The algorithm here proposed is similar to the ones already studied in the literature and can be considered as a constant bearing guidance algorithm. The main contribution is the modification implemented to foresee the position of the ship so that it is reached faster, following a shorter path. The algorithm was compared with the one presented in [6], and the improvement was shown. The tuning of the parameters was done by analyzing the kinematics, in a similar way as with the path-following algorithm. The robustness of the algorithm was tested by applying an external perturbation and the performance was measured by integrating the distance to the ship and the needed effort.

The quadcopter proposed, although cannot be used in real situations because of its short endurance, low maximum endurance and pay-load, is useful as a testbed to prove the convenience of these algorithms. The algorithms have been implemented in a Simulink model, being possible to simulate all types of missions and to export them to the real hardware.

As can be seen, the objectives of this work have been successfully fulfilled, not only by using techniques already developed by other authors, but also by contributing with new detailed analysis and improvements, which showed to contribute with *SALACOM* project.

5.3 Future works

We can identify some improvements and lines of work to further develop this algorithms and test them in a real system. Although the project is way more complex, only possible future development, impacting the work here exposed, will be now presented.

On one hand, robustness of the algorithms should be tested exhaustively. In this work, only perpendicular perturbations, which move the quadcopter off its path, were considered. This perturbations were only peaks of forces and their values were taken so that the UAV was displaced far enough from its track. However, a quadcopter in the real environment will be subject to winds and gusts, which will affect its performance. These forces must be estimated and included in the simulations, and probably the parameter selection will need to be repeated to maximize the performance of the algorithms. Also, Monte Carlo simulations can be used to cover a wide range of situations.

Once the upper level controller is developed, it might be possible that time

Figure 5.2: Example of quadcopter with its payload covered with a nacelle. The direction of flight must be aligned with it.

constraints are needed, and therefore, the path-following algorithm might be converted into a path-tracking one. Recall that the original work [41] had also the ability to track the path, having decoupled the variables space and time. This task will be then carried out using the work already done.

If other algorithms were available and tuned for this quadcopter, it would be also interesting to compare them with the ones developed. This comparison was out of the scope of this work, since the implementation and tuning would have needed a lot of work. It is true that another algorithm to track the ship was compared, but it was only to compare the general strategy of both methods.

The yaw control of the quadcopter itself has been omitted in the whole work. This is because the algorithms command the velocity direction, and because of the quadcopter nature, it can be achieved without changing the yaw of the UAV. However, if the payload and/or the frame are covered by a nacelle, it might be needed to orientate the quadcopter in a preferred direction. In Figure 5.2, an example of quadcopter with its payload covered with a nacelles is shown. The UAV must flight so that the air comes to it with the proper incidence. Also, if the vision system needs to capture the images with an orientation, the yaw control must controlled as well.

An algorithm is needed to land the quadcopter on the ship, or, at least, a fine tuning of the present algorithm when the quadcopter is close to the ship. This operation is delicate and would need more accurate positioning information from equipements such as a vertical camera and a laser rangefinder. A deep robustness study, considering the *ground effect*, must be carried out. The altitude control here used was simple, and when landing, the altitude becomes critical.

It is well known that endurance and range rotorcraft in general, and quadcopters in particular, depend on the airspeed [40]. Actually, the velocity to achieve the longest range is faster than the velocity to reach the highest endurance. To maximize the utility of the quadcopter, depending on the requirements of the mission, a study should be carried out to identify these velocities.

The quadcopter to be used in the final system will be another one. Therefore, an adaptation of the algorithms must be performed, taking into account the new

quadcopter characteristics. To do that, this work might be used as a guide for that configuration. Finally, the software must be boarded in a real system and tested.

Bibliography

[1] Adler, F. P. (1956). Missile Guidance by Three-Dimensional Proportional Navigation. Journal of Applied Physics, 27(5), 500-507.

[2] Aznar, F., Sempere, M., Pujol, M., Rizo, R., & Pujol, M. J. (2014). Modelling Oil-Spill Detection with Swarm Drones. In Abstract and Applied Analysis. Hindawi Publishing Corporation.

[3] Belkhouche, F., Belkhouche, B., & Rastgoufard, P. (2006). Line of sight robot navigation toward a moving goal. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 36(2), 255-267.

[4] Bishop, R. L. (1975). There is more than one way to frame a curve. American Mathematical Monthly, 246-251.

[5] Breivik, M., & Fossen, T. I. (2007). Applying missile guidance concepts to motion control of marine craft. In Control Applications in Marine Systems, 7, No. 1, pp. 349-354.

[6] Breivik, M., Hovstein, V. E., & Fossen, T. I. (2008). Straight-line target tracking for unmanned surface vehicles. Modeling, Identification and Control, 29(4), 131-149.

[7] Breivik, M., Strand, J. P., & Fossen, T. I. (2006). Guided dynamic positioning for fully actuated marine surface vessels. Proceedings of the 6th IFAC MCMC, Lisbon, Portugal.

[8] Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm (No. CMU-RI-TR-92-01). Carnegie-Mellon Univ PITTSBURGH Pa Robotics Ins.

[9] Dobrokhodov, V. N., Kaminer, I. I., Jones, K. D., & Ghabcheloo, R. (2006). Vision-based tracking and motion estimation for moving targets using small UAVs. In American Control Conference, 2006. IEEE.

[10] Dongxuan, L., Ting, Z., Weijie, L., Jinku, L., Yina, R., & Di, M. (2014, August). Robust trajectory control of a four-axis dual rotor aircraft. In Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese (pp. 927-931). IEEE.

[11] Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics, 497-516.

[12] Educate, Q. I. Quanser Qball-x4 user manual.

[13] Conte, G., Duranti, S., & Merz, T. (2004). Dynamic 3D path following for an autonomous helicopter. In Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles.

[14] de la Cruz, J. M., Lopez-Orozco J. A., Besada-Portas, E., Aranda-Almansa, J (2015).A Streamlined Nonlinear Path Following Kinematic Controller. IEEE International Conference on Robotics and Automation.

[15] Fossen, T. I., Breivik, M., & Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. Proceedings of the 6th IFAC MCMC, Girona, Spain, 244-249.

[16] García-Auñón, P. & Santos, M. (2014). Use of genetic algorithms for unmanned aerial systems path planning. Decision Making and Soft Computing **9** 430–435

[17] García-Auñón, P. & Santos, M. (2015). A First Intelligent Approach to Path Following Algorithm for Quadcopters. International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO) **10**

[18] Hehn, M., & D'Andrea, R. (2011, September). Quadrocopter trajectory generation and control. In IFAC World Congress (Vol. 18, No. 1, pp. 1485-1491).

[19] Hoffmann, G. M., Waslander, S. L., & Tomlin, C. J. (2008, August). Quadrotor helicopter trajectory tracking control. In AIAA Guidance, Navigation and Control Conference and Exhibit (pp. 1-14).

[20] Kothari, M., Postlethwaite, I., & Gu, D. W. (2010). A suboptimal path planning algorithm using rapidly-exploring random trees. International Journal of Aerospace Innovations, 2(1), 93-104.

[21] Kuo, B. C. (1981). Automatic control systems. Prentice Hall PTR.

[22] Martinez, S. E., & Tomas-Rodriguez, M. (2014). Three-dimensional trajectory tracking of a quadrotor through PVA control. REVISTA IBEROAMERICANA DE AUTOMATICA E INFORMATICA INDUSTRIAL, 11(1), 54-67.

[23] Mejari, M., Gupta, A., Singh, N. M., & Kazi, F. (2013). Trajectory Tracking of Quadrotor with Bounded Thrust using Model Predictive Control. In Proceedings of Conference on Advances In Robotics (pp. 1-6). ACM.

[24] Nelson, D. R., Barber, D. B., McLain, T. W., & Beard, R. W. (2007). Vector field path following for miniature air vehicles. Robotics, IEEE Transactions on, 23(3), 519-529.

[25] Lapierre, L., & Soetanto, D. (2007). Nonlinear path-following control of an AUV. Ocean engineering, 34(11), 1734-1744.

[26] Lawrence, D., Frew, E., & Pisano, W. (2007). Lyapunov vector fields for autonomous UAV flight control. In AIAA Guidance, Navigation and Control Conference and Exhibit pp. 2007-6317.

[27] Lee, S., Cho, A., & Kee, C. (2010). Integrated waypoint path generation and following of an unmanned aerial vehicle. Aircraft Engineering and Aerospace Technology, 82(5), 296-304.

[28] Lee, T., Leok, M., & McClamroch, N. H. (2010). Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3). arXiv preprint arXiv:1003.2005.

[29] Papoulias, F. A. (1992). Guidance and control laws for vehicle pathkeeping along curved trajectories. Applied ocean research, 14(5), 291-302.

[30] Park, S., Deyst, J., & How, J. P. (2004, August). A new nonlinear guidance logic for trajectory tracking. In Proceedings of the AIAA Guidance, Navigation and Control Conference (pp. 1-16).

[31] Park, S., Deyst, J., & How, J. P. (2007). Performance and Lyapunov stability of a nonlinear path following guidance method. Journal of Guidance, Control, and Dynamics, 30(6), 1718-1728.

[32] Partington, K. (2014). An Assessment of Surface Surveillance Capabilities for Oil Spill Response using Airborne Remote Sensing. Polar Imaging Limited.

[33] Pereira, E., Bencatel, R., Correia, J., Félix, L., Gonçalves, G., Morgado, J., & Sousa, J. (2009). Unmanned Air Vehicles for coastal and environmental research. Journal of Coastal Research, 1557-1561.

[34] Pereira, E., da Silva, P. M., Krainer, C., Kirsch, C. M., Morgado, J., & Sengupta, R. A Networked Robotic System and its Use in an Oil Spill Monitoring Exercise.

[35] Rysdyk, R. (2003, September). UAV path following for constant line-of-sight. In 2th AIAA Unmanned Unlimited. Conf. and Workshop and Exhibit, San Diego, CA.

[36] Santos, M. (2011). Intelligent control: a practical approach. Revista Iberoamericana de Automática e Informática Industrial **8**, 283–296

[37] Salazar-Cruz, S., Palomino, A., & Lozano, R. (2005). Trajectory tracking for a four rotor mini-aircraft. In Decision and Control, European Control Conference. CDC-ECC'05. 44th IEEE Conference on (pp. 2505-2510). IEEE.

[38] Sujit, P. B., Saripalli, S., & Borges Sousa, J. (2014). Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicles. Control Systems, IEEE, 34(1), 42-59.

[39] Tan, L., Lu, L., & Jin, G. (2012). Attitude stabilization control of a quadrotor helicopter using integral backstepping. In Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on (pp. 573-577). IET.

[40] Torenbeek, E., & Wittenberg, H. (2009). Helicopter Flight Mechanics. Flight Physics: Essentials of Aeronautical Disciplines and Technology, with Historical Notes, 405-430.

[41] Xargay Mata, E. (2013). Time-critical cooperative path-following control of multiple unmanned aerial vehicles (Doctoral dissertation, University of Illinois at Urbana-Champaign).