Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Meccanica e Scienze Avanzate dell'Ingegneria
Ciclo 30

Settore Concorsuale: 09/A1 Ingegneria aeronautica, aerospaziale e navale

Settore Scientifico Disciplinare: ING-IND/03 Meccanica del volo

Development of a fight control architecture for rotary-wing UAVs with model-based design approach

Presentata da:

Gianluca Rossetti

Coordinatore Dottorato	**Supervisore**
Prof. Marco Carricato	Prof. Fabrizio Giulietti

Esame finale anno 2018

Declaration of Authorship

I, Gianluca Rossetti, declare that this thesis titled, 'Development of a flight control architecture for rotary-wing UAVs with model-based design approach' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Ph.D. degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

"The best design method is the one with which the designer is most comfortable."

D.J. Moorhouse, Flight Control Design Best Practices, 2000

ALMA MATER STUDIORUM UNIVERSITÁ DI BOLOGNA

Abstract

University of Bologna
Department of Industrial Engineering

Doctorate in Mechanics and Advanced Engineering Sciences

by Gianluca Rossetti

This thesis describes the design and implementation of various autopilot software architectures for mini/micro rotary-wing unmanned aerial vehicles by exploiting the model-based design approach. Nowadays in fact, the tendency for software development is changing from manual coding to automatic code generation, in other words, it is becoming model-based. In general, models can be described as abstractions of systems, they are created to serve particular purposes, for example, to present a user-understandable description of the system or to present information in a more intuitive form. Model-based techniques for software design enables the engineer to reduce drastically development time required for software corrections or modifications. Under the various chapters, different flight control techniques are presented with theoretical background and tested via simulations and experimental campaigns. All the navigation and control problems presented below arise in development of embedded software that exploits the innovative model-based design technology. In order to provide validations of the proposed solutions, software for simulation and implementation is specialized for the case of multirotor vehicles, which are becoming very helpful systems for many and varied civil operations. This is the reason why part of the text is devoted to multirotor vehicle dynamics.

Acknowledgements

This thesis is submitted in partial fulfillment of the requirements for the Doctor of Philosophy in 'Mechanics and Advanced Engineering Sciences'. The work has been carried out during the period from November 2014 to March 2017 under the supervision of Prof. Fabrizio Giulietti. I would like to thank Prof. Fabrizio Giulietti for his support and guidance, Matteo Turci and Emanuele Luigi de Angelis for their fundamental help, it has been really inspiring working with all of you. I would also like to thank Prof. Goele Pipeleers, Ruben Van Parys and the whole MECO research crew for the good time spent in KU Leuven. Finally, thanks to my dear family and my friends for having them always by my side during these years.

Contents

List of Figures

List of Tables

Abbreviations

RPAS	**R**emotely **P**iloted **A**ircraft **S**ystem
UAV	**U**nmanned **A**erial **V**ehicle
VLOS	**V**isual **L**ine **O**f **S**ight
BVLOS	**B**eyond **V**isual **L**ine **O**f **S**ight
GCS	**G**round **C**ontrol **S**tation
FMU	**F**light **M**anagement **U**nit
GNC	**G**uidance **N**avigation **C**ontrol
GNU	**GNU's** Not Unix
GPL	**G**eneral **P**ubblic **L**icense
LGPL	**L**esser **G**eneral **P**ubblic **L**icense
MPC	**M**odel **P**redictive **C**ontrol
NMPC	**N**on-linear **M**odel **P**redictive **C**ontrol
DC	**D**irect **C**urrent
BLDC	**B**rush-**L**ess **D**irect **C**urrent
ESC	**E**lectronic **S**peed **C**ontroller
PSP	**P**ilot **S**upport **P**ackage
EKF	**E**xtended **K**alman **F**ilter
GPS	**G**lobal **P**osition **S**ystem
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
DCM	**D**irection **C**osine **M**atrix
OS	**O**perative **S**ystem
PID	**P**roportional **I**ntegral **D**erivative
PWM	**P**ulse-**W**idth **M**odulation
OMG	**O**ptimal **M**otion **G**eneration
MIMO	**M**ultiple-**I**nput **M**ultiple-**O**utput

FGC	**F**ormation **G**eometry **C**enter
SD	**S**ecure **D**igital

Symbols

t	Time	s
m	Mass	kg
R	Transformation matrix	
g	Gravitational acceleration	m/s^2
F^i	Inertial frame	
F^v	Vehicle frame	
F^b	Body frame	
$\xi = [x, y, z]^T$	Position vector	m
$V = [u, v, w]^T$	Velocity vector	m/s
$\alpha = [\phi, \theta, \psi]^T$	Euler's angle vector	rad
$\Omega_{b/i} = [p, q, r]^T$	Angular rates vector	rad/s
$F = [F_x, F_y, F_z]^T$	External force vector	N
h	Angular momentum	kg$\cdot m^2/s$,
$M = [\tau_\phi, \tau_\theta, \tau_\psi]^T$	Torque vector	N$\cdot m$
J	Moment of inertia matrix	m^4
K	Motor constant parameter	N/(s$\cdot10^{-6}$)
δ_{pwm}	Motor command signal	s$\cdot10^{-6}$
l	distance between motor and center of mass	m
C_D	Drag coefficient	
ρ	Air density	kg/m^3
A	Equivalent flat plate areas	m^2
ϵ	Error signal,	
	Soft-formation deviation	
$K_{P/I/D}$	Controller gains	m
a	Acceleration	m/s^2,

	Normal vector to the hyperplane,	
	Potential variable	
T	Final time	s
q	Trajectory	
$h(q,t)$	Inequality constraints	
$J(q)$	Cost function	
q	Spline control points	
b	Spline basis	m
f	Total acceleration	m/s^2
ω	Angular rate	rad/s
r	Vehicle influence radius,	
	Position vector	m
w	polyhedron vertex	m
b	Hyperplane's offset	
$x(t)$	System state	
$\hat{x}(t)$	Estimate system state	
ΔT	Update time	s
d	Distance vector	m
u	Models' input	
P_i	Actual position point	m
D_i	Desired position point	m
w	Soft-formation parameter	

Dedicated to my grandpas.

Chapter 1

Introduction

We live in a time of great technological innovation and Remotely Piloted Aircraft Systems (RPASs), also known as Unmanned Aerial Vehicles (UAVs), or popularly as drones, can certainly be considered part of this change. UAVs basically are aircrafts with no human pilot on-board; they are flown through a ground control station but they are generally capable to fly autonomously with pre-programmed flight plans, which is crucial, for example, in Beyond-Visual-Line-Of-Sight (BVLOS) operations. The control system of an UAV is always composed by two major sub-systems: the control station at ground and the flight management unit on-board. A Ground Control Station (GCS) is a set of ground-based systems which provide human control and telemetry feedback to the operator. They can have different forms and dimensions depending on the particular type of UAV; smaller UAVs can be operated with a traditional transmitter as used for radio-controlled models. The extension of this setup with data and video telemetry by laptop or tablet computer creates what is effectively considered a Ground Control Station. On the other hand, an on-board Flight Management Unit (FMU) is an embedded system that handles at different levels of automation the UAV during flight. The FMU primary task is performing Guidance, Navigation and Control (GNC) algorithms, which means using sensors data for operating at different levels on the flight behaviour of the UAV. For these reasons, the term ”autopilot” is also widely used when referring to both the hardware and the software of the FMU. There are many UAV autopilots out on the market but two main types are distinguished. Some companies sell commercial autopilots with proprietary design; although they are often very good products, they appear to be ”black boxes” in the eyes of a customer. In this case, vendors need to establish an efficient support service because customers are dependent for any technical issue or modification into the autopilot. On the other hand, many other companies sell autopilots derived from open-source projects. In an open-solution in fact, companies can reduce cost and time to market and customers have the possibility to know exactly what is happening inside the autopilots, becoming developers if they want to. For example, this fact gives great advantages during research projects, in which open-systems are always favoured. As regards open-software proprietary information, today the two most common open-source licenses are GNU General Public License (GPL) and GNU Lesser General Public License (LGPL), for both of them any modifications made to the source code must become public domain [1]. Anyway, in both open source and commercial worlds, traditional methods used for synthesizing, implementing and validating a flight management software are still the standard. Traditional coding approach obliges engineers to use complex structures and extensive work for converting engineering principles and control theories into software code, which consumes time and resources for companies and researchers. For this reason, another approach, called model-based design, is taking off quickly in software engineering. Model-based design approach is significantly different from traditional design methodology and, for various reasons, it results to be much more appropriate for small-UAV software development. Today, flight software designers can define plant models with advanced functional features, use building blocks and tools which can lead to rapid software prototyping, testing and deploying more quickly and much more efficiently. In favour of what has been said, this thesis wants to give clear example of the potential and advantages which model-based design can provide in software development, applying that methodology for the flight software project of multirotor UAVs.

1.1 Multirotor vehicles

Drone technology has developed and prospered in the last few years both in military and civil field, especially as regards the category of small multirotor vehicles, or multirotors, which nowadays is the most common. Recent civil applications highlighted the capability of multirotor platforms to perform unimaginable mission tasks ranging from environment monitoring and remote sensing to surveillance and rescue operations [2], whereas theoretical studies provided novel design tools for optimal performance [3, 4]. A multirotor is an aerial vehicle with very simple mechanics, motion is controlled by speeding or slowing multiple propeller motors installed on a rigid frame. The multirotor vehicle is intrinsically unstable, that is way it requires a flight controller for performing stable flight. Respect to helicopters or fixed-wing UAVs, a real manual-controlled flight is not possible. Many multirotor configurations exist, based on the number and position of rotors they are called tri-rotors, quad-rotors, hexa-rotors and so on. Although the invention of this type of aerial vehicle is dated almost ninety years ago, when the four-rotor helicopter of Louis Breguet succeeded to lift itself off the ground for a few seconds, the large-scale dissemination of the multirotor architecture happened only some years ago. This fact is definitely connected with the great progress made in the field of information technology, in terms of miniaturization, reduction in cost of electronic components and computing performance. In relation to other types of UAVs, multirotors have become much more popular to the point that, nowadays, the vast majority of people uses the newly coined word "drones" for referring only to the multirotor class. Their fame came from user-friendly features, mechanical simplicity and flying capabilities. With respect to fixed-wing configurations, in fact, the hovering and vertical take-off capabilities represent crucial aspects for many aerial operations[5]. Furthermore, with respect to conventional remotely piloted helicopters, multirotor platforms show great advantages in large-scale sales in terms of safety, reliability and controllability.

1.2 Outline

The aim of this thesis is to present potentialities of the model-based design approach via the development of a flight control software architecture conceived for multirotor UAVs. That approach is intended for combining theoretical design tools and experimental procedures, so that become easy to synthesize, implement and test flight controllers of small UAVs in a safer, cost effective and time efficient way. This thesis proposes different GNC algorithms and strategies which try to address different tasks in operations with single multirotor or a fleet of them. In particular these tasks are trajectory planning, trajectory tracking, collision avoiding and motion controlling. The overall control design process is covered, including modelling, control development and validation with simulations and experimental tests.
The Thesis is organized as follows:

In Chapter 2 the non-linear simulation model of a multirotor vehicle is derived starting from the definition of the rigid body equations of motion. Model, developed in Matlab/Simulink® environment, is designed for testing synthesized control architectures before they are put into an embedded hardware for flight tests.

Chapter 3 presents the general architecture of the flight control software developed with the model-based methodology. This approach uses the Matlab/Simulink® tool suite for developing the architecture, design and modeling the software parts. The software code is then auto-generated by embedded coder. The motion control architecture is also explained in detail and supported by simulations and flight tests.

Chapter 4 explains an innovative methodology for computing optimal autonomous navigation for multirotor vehicles in obstructed environments. The navigation algorithm, formulated in a Model Predictive Control (MPC) fashion, guarantees capability of trajectory planning and tracking with obstacle avoidance. After a simulation campaign, the strategy is incorporated in the flight control software and tested in outdoor environment.

Chapter 5 focuses the attention into the UAV formation flight control problem. Two alternative strategies are proposed: the first one is a potential method that ensures trajectory tracking, formation keeping and collision avoidance tasks. The second method follows the work explained in Chapter 4, extending it for formations of UAVs. For both methodologies, results of numerical simulations are provided to confirm their validity. Potential method is also tested in outdoor environment, as part of the flight control software, for validation in presence of external disturbances and unmodeled dynamics.

Chapter 2

Multirotors kinematics and dynamics

An important stage of model-based flight control design approach is represented by the derivation of a suitable flight dynamics model of the aerial vehicle which can be used in different steps of controller development. In this chapter, the non-linear mathematical model of multirotor is defined starting from general expressions for the kinematics and dynamics of a rigid body. The derived mathematical model is just an approximation of a real multirotor dynamics, in particular because some aerodynamic effects are still less well understood and hard to model, but either way it gives fundamental support for controllers development. In order to ensure a more clear explanation, the quad-rotor case is considered as example of the equation specifications part.

2.1 Reference frames

First, three right-handed orthogonal reference frames are introduced, they are used to derive the mathematical model of multirotor.

1. NED Frame F^i (North-East-Down): this is an inertial frame under the assumption of flat and non-rotating Earth.
2. Vehicle Frame F^v: this is a Local Vertical / Local Horizontal frame, the origin is located at the center of gravity of the multirotor. It has axes parallel to the inertial frame.

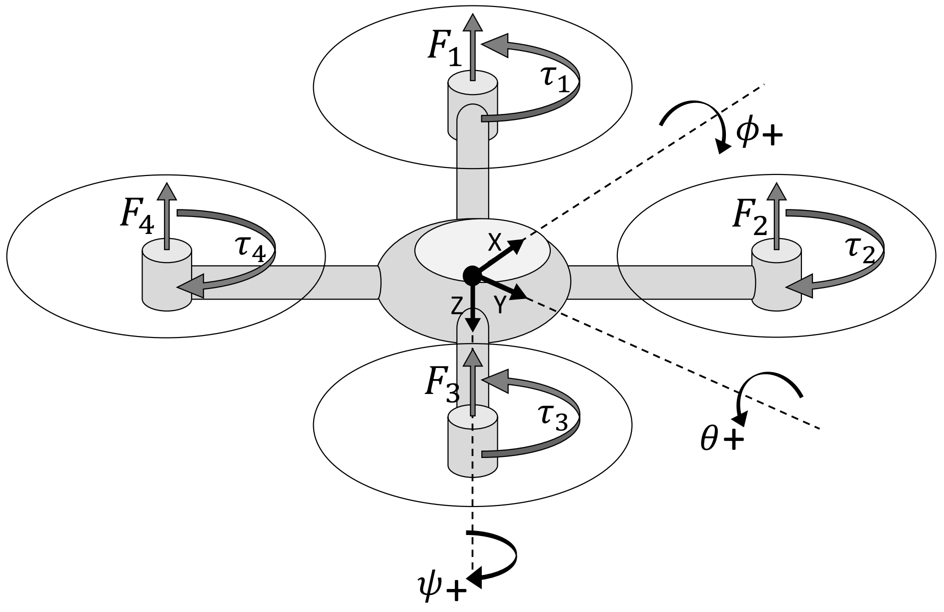


Figure 2.1: Representation of a three-dimensional quadrotor.

3. Body Fixed Frame F^b: As illustrated in Figure 2.2, the origin of this frame is located at the center of gravity of the multirotor. The X axis points in forward direction, generally defined by the inertial measurement unit orientation or by vehicle geometry; the Y axis points to the right; the Z axis points downwards.

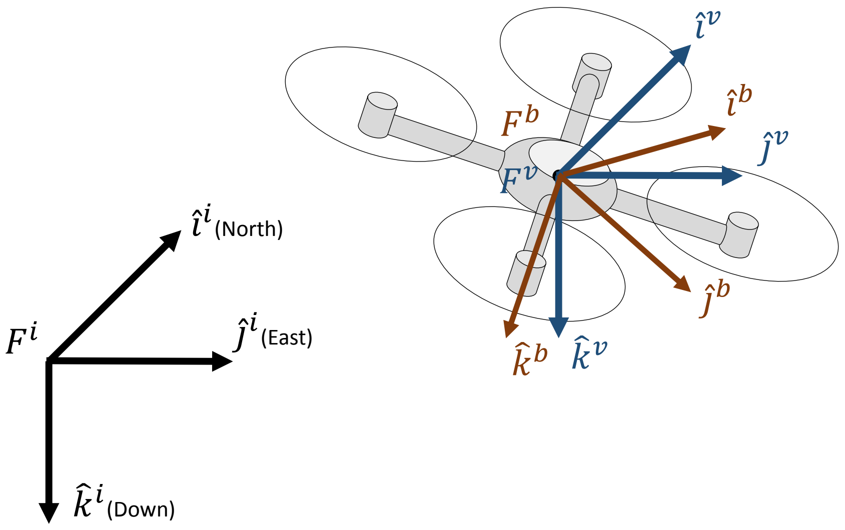


Figure 2.2: Representation of the three reference frames

The multirotor state is composed by twelve variables:

$$\begin{bmatrix} x & y & z \end{bmatrix}^T : \text{Position} \tag{2.1}$$

$$\begin{bmatrix} u & v & w \end{bmatrix}^T : \text{Velocity} \tag{2.2}$$

$$\begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T : \text{Euler angles} \tag{2.3}$$

$$\begin{bmatrix} p & q & r \end{bmatrix}^T : \text{Angular rates} \tag{2.4}$$

The position vector of the multirotor is given with respect to the NED frame. Velocity and angular velocity vectors are given in the body fixed frame.

2.2 Multirotor kinematics

The relationship between the position defined in the vehicle frame and the velocity defined in the body fixed frame is given by:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.5}$$

where R_b^v represents the rotation matrix [6] for vector transformation between body fixed frame and vehicle frame:

$$R_b^v = \begin{bmatrix} c\theta\, c\psi & s\phi\, s\theta\, c\psi - c\phi\, s\psi & c\phi\, s\theta\, c\psi + s\phi\, s\psi \\ c\theta\, s\psi & s\phi\, s\theta\, s\psi + c\phi\, c\psi & c\phi\, s\theta\, s\psi - s\phi\, c\psi \\ -s\theta & s\phi\, c\theta & c\phi\, c\theta \end{bmatrix} \tag{2.6}$$

the relationship between absolute angular rates ϕ , θ and ψ and the angular rates p , q and r defined in the body fixed frame is given by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi\, t\theta & c\phi\, t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} , \tag{2.7}$$

in which c, s and t symbolize respectively cos, sin and tan.

2.3 Multirotor dynamics

The Newton's law applied to the translational motion is:

$$m\frac{dV}{dt_i} = F \tag{2.8}$$

in which m is the mass of the multirotor, v is the velocity vector, $\frac{d}{dt_i}$ is the time derivative in the NED frame, F is the total force applied to the multirotor. From the equation of Coriolis, equation (2.8) becomes:

$$m\frac{dV}{dt_i} = m(\frac{dV}{dt_b} + \Omega_{b/i} \times V) = F \tag{2.9}$$

where $\frac{d}{dt_b}$ is the time derivative in the body fixed frame (airframe), $\Omega_{b/i} = \begin{bmatrix} p & q & r \end{bmatrix}^T$ is the angular velocity of the airframe with respect to the NED frame. Equation (2.9) can be also written as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{2.10}$$

The Newton's second law for the rotational motion is:

$$\frac{dh}{dt_i} = M \tag{2.11}$$

where h is the angular momentum and M is the applied torque. From the equation of Coriolis, equation (2.11) becomes:

$$\frac{dh}{dt_i} = \frac{dh}{dt_b} + \Omega_{b/i} \times h = M \tag{2.12}$$

writing $h = J\Omega_{b/i}$ as follows:

$$J\frac{d\Omega_{b/i}}{dt_b} + \Omega_{b/i} \times (J\Omega_{b/i}) = M \tag{2.13}$$

$$\frac{d\Omega_{b/i}}{dt_b} = J^{-1}(M - \Omega_{b/i} \times (J\Omega_{b/i})) \tag{2.14}$$

Assuming the multirotor as a symmetric body for all three axes, the constant inertia matrix J can be written as:

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{2.15}$$

Defining $M = \begin{bmatrix} \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T$, equation (2.14) can be formulated in body coordinates as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{bmatrix} + \begin{bmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\theta \\ \frac{1}{J_z} \tau_\psi \end{bmatrix} \tag{2.16}$$

2.4 Forces and moments

From a mechanical point of view, the multirotor is a quite simple aerial vehicle, probably the most simple after balloons and gliders. With the exception of the tri-rotor configuration, that needs a yaw-tilt servo to balance torques, it consists of an even number of rotors attached to a rigid airframe. Airframes are built with the aim of placing rotors equidistant from the centre of gravity, which generally matches the power cell position. Except some very rare cases, the propulsion system is electric and formed by brushless DC electric motors (BLDC), electronic speed control units (ESCs) and batteries. Forces and moments in multirotors are primarily due to gravity and propellers but, in a navigation context, reliable simulation results can only be obtained taking into account also drag forces during motion. That said, equation (2.10) can be written as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\left(\begin{bmatrix} F_g \end{bmatrix} + \begin{bmatrix} F_p \end{bmatrix} + \begin{bmatrix} F_d \end{bmatrix} \right) \tag{2.17}$$

In the vehicle frame, the gravity force acting on the center of mass is given by:

$$F_{g|i} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{2.18}$$

Transforming to the body fixed frame gives:

$$F_g = R_v^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -mg \sin\theta \\ mg \cos\theta \sin\phi \\ mg \cos\theta \cos\phi \end{bmatrix} \tag{2.19}$$

Multirotor control is achieved by controlling the power generated by each motor. So each motor produces a force F and a torque τ defined as:

$$F_{(i)} = K_1 * \delta_{pwm(i)} \tag{2.20}$$

$$\tau_{(i)} = K_2 * \delta_{pwm(i)} \tag{2.21}$$

In which K_1 and K_2 are constants that have to be determined experimentally, δ_{pwm} is the motor command signal. In order to write forces and torques that act on the multirotor, it is necessary to define the number of motors and the configuration of the frame. As illustrated in Figure 2.1, a four-motor cross configuration (sometimes called X-quadrotor) is chosen as example; the following quantities can be defined:

Total force:

$$F_{tot} = F_1 + F_2 + F_3 + F_4 \tag{2.22}$$

Roll and pitch control are obtained by modifying the speed of the rotors in pairs, while yaw control is obtained by modifying the average speed of the clockwise and anticlockwise rotating rotors.
Roll torque:

$$\tau_\phi = l\frac{\sqrt{2}}{2}(F_1 - F_2 - F_3 + F_4) \tag{2.23}$$

Pitch torque:

$$\tau_\theta = l\frac{\sqrt{2}}{2}(F_1 + F_2 - F_3 - F_4) \tag{2.24}$$

Yaw torque:

$$\tau_\psi = \tau_1 - \tau_2 + \tau_3 - \tau_4 \tag{2.25}$$

In which l stands for the distance in the X-Y plane between the center of gravity and the point of application of the i-th motor force. The force produced by propellers in the body fixed frame is:

$$\begin{bmatrix} F_p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -(F_1 + F_2 + F_3 + F_4) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -F_{tot} \end{bmatrix} \tag{2.26}$$

The drag force in the body fixed frame is given by:

$$\begin{bmatrix} F_d \end{bmatrix} = -\frac{1}{2}\rho C_D \begin{bmatrix} A_x u|u| \\ A_y v|v| \\ A_z w|w| \end{bmatrix} \tag{2.27}$$

where ρ is the air density, C_D is the drag coefficient and A_x, A_y and A_z are the projections of the quadrotor surfaces in the body frame.

Finally equation (2.16) can be formulated as follows:

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}rv-qw\\ pw-ru\\ qu-pv\end{bmatrix}+\frac{1}{m}(\begin{bmatrix}-g\sin\theta\\ g\cos\theta\sin\phi\\ g\cos\theta\cos\phi\end{bmatrix}+\begin{bmatrix}0\\ 0\\ -F_{tot}\end{bmatrix}+\frac{1}{2}C_D\rho\begin{bmatrix}-A_xu|u|\\ -A_yv|v|\\ -A_zw|w|\end{bmatrix}) \tag{2.28}$$

The six-degree of freedom quadrotor model is now defined for the twelve state variables and it is formulated by equations (2.5),(2.7),(2.16) and (2.17). As any other multirotor model, this is an under-actuated model; so the translational speed in the horizontal plane of the NED frame must be managed through the system dynamics, by controlling the attitude.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}=\begin{bmatrix}c\theta c\psi & s\phi s\theta c\psi-c\phi s\psi & c\phi s\theta c\psi+s\phi s\psi\\ c\theta s\psi & s\phi s\theta s\psi+c\phi c\psi & c\phi s\theta s\psi-s\phi c\psi\\ -s\theta & s\phi c\theta & c\phi c\theta\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix} \tag{2.29}$$

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}rv-qw\\ pw-ru\\ qu-pv\end{bmatrix}+\begin{bmatrix}-mg\sin\theta\\ mg\cos\theta\sin\phi\\ mg\cos\theta\cos\phi\end{bmatrix}-\frac{1}{m}\begin{bmatrix}0\\ 0\\ F_{tot}\end{bmatrix}-\frac{1}{m}\frac{1}{2}C_D\rho\begin{bmatrix}A_xu|u|\\ A_yv|v|\\ A_zw|w|\end{bmatrix} \tag{2.30}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & sin\phi tan\theta & cos\phi tan\theta\\ 0 & cos\phi & -sin\phi\\ 0 & \frac{sin\phi}{cos\theta} & \frac{cos\phi}{cos\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{2.31}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}\frac{J_y-J_z}{J_x}qr\\ \frac{J_z-J_x}{J_y}pr\\ \frac{J_x-J_y}{J_z}pq\end{bmatrix}+\begin{bmatrix}\frac{1}{J_x}l\frac{\sqrt{2}}{2}(F_1-F_2-F_3+F_4)\\ \frac{1}{J_y}l\frac{\sqrt{2}}{2}(F_1+F_2-F_3-F_4)\\ \frac{1}{J_z}(\tau_1-\tau_2+\tau_3-\tau_4)\end{bmatrix} \tag{2.32}$$

2.5 Simplified multirotor models

The six degree-of-freedom model represented by equations (5.5), (2.30), (2.31) and (2.32) can be used to perform accurate simulations. However, the model turns out to be not appropriate or too complex for other purposes like control design [6] or motion planning methodologies, reasons of that will be more clear along the following sections. In this section some approximations are made to the equations in order to obtain two simplified derived models.

2.5.1 Simplified inertial model

Let is start differentiating equation (5.5), neglecting $\dot{R}_b^v$ gives:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} \tag{2.33}$$

In (2.30) all the Coriolis terms are considered small and can be neglected, moreover drag effects are removed from equation. Plugging that equation into (2.33) gives:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} -c\phi s\theta c\psi - s\phi s\psi \\ -c\phi s\theta s\psi + s\phi c\psi \\ -c\phi c\theta \end{bmatrix} \frac{F_{tot}}{m} \tag{2.34}$$

Angles ϕ, θ and ψ are assumed small in order to write:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.35}$$

In equation (2.32), Coriolis terms qr, pr and pq are considered small and can be neglected:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x}\tau_\phi \\ \frac{1}{J_y}\tau_\theta \\ \frac{1}{J_z}\tau_\psi \end{bmatrix} \tag{2.36}$$

Writing $f = \frac{F_{tot}}{m}$, finally the following simplified inertial model can be written:

$$\begin{cases} \ddot{x} = f(-\cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi)\,, \\ \ddot{y} = f(-\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi)\,, \\ \ddot{z} = -f\ \cos\phi\cos\theta + g\,, \\ \ddot{\phi} = \dfrac{1}{J_x}\tau_\phi\,, \\ \ddot{\theta} = \dfrac{1}{J_y}\tau_\theta\,, \end{cases} \tag{2.37}$$

2.5.2 Simplified model for control design

The mechanical simplicity of multirotors comes with a price: they are under-actuated vehicles. In particular, it is possible to control only four of the six degrees-of-freedom. Multirotor vehicles are capable of tracking desired attitudes, headings and accelerations in the body-fixed vertical direction, but they cannot achieve accelerations in the body-fixed horizontal plane. This results in a coupling between multirotor attitude and acceleration. For control design purposes, in order to find relations between angles ϕ/θ

and forward/right accelerations, the heading angle ψ becomes irrelevant and can be set equal to zero. This is equivalent to rotate the vehicle frame F^{v} by the heading angle, equations (2.37) become:

$$\begin{cases} \ddot{x} = -f\ \cos\phi\sin\theta\,, \\ \ddot{y} = f\ \sin\phi\,, \\ \ddot{z} = -f\ \cos\phi\cos\theta + g\,, \\ \ddot{\phi} = \dfrac{1}{J_x}\tau_\phi\,, \\ \ddot{\theta} = \dfrac{1}{J_y}\tau_\theta\,, \end{cases} \tag{2.38}$$

Section 3.4.4.4 describes how relations between angles and accelerations can be derived starting from the model above, while Appendix A tackles the same problem considering drag forces in the solutions.

2.5.3 Simplified inertial model with small angle approximation

The model represented by equations (2.38) can be further reduced by introducing small angle approximation, $\sin\alpha \approx \alpha$ and $\cos\alpha \approx 1$. This approximation is considered valid for attitude angles smaller than 15°.

$$\begin{cases} \ddot{x} = -f\,\theta\,, \\ \ddot{y} = f\,\phi\,, \\ \ddot{z} = -f + g\,, \\ \ddot{\phi} = \dfrac{1}{J_x}\tau_\phi\,, \\ \ddot{\theta} = \dfrac{1}{J_y}\tau_\theta\,, \end{cases} \tag{2.39}$$

Chapter 3

Flight control software development

This chapter presents the architecture of the flight control software developed during the doctorate. The model-based approach helps the software development as well as the presentation of all its sub-systems. First of all, a quick presentation of the hardware used for software deployment is necessary, following a summary of flight modes are commonly implemented and used for UAV control. The chapter continues with the description of the software parts, especially those relating to architecture of the flight control systems.

3.1 Model-based design approach

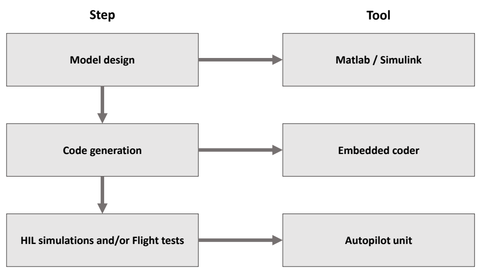


FIGURE 3.1: The model-base approach for UAV Autopilot Software developing

Model-Based Design approach is a method for developing dynamic processes by using visual and mathematical tools. For example, problems which are often associated with model-based design are complex control systems developing, signal processing and communication systems design. The main reason to use model-based design approach is

improving the product quality and reducing development time, in fact today this approach is increasingly used in the industry [7]. Regarding the topic of this thesis, the model-based design approach is used with the aim of developing flight management code suitable for UAV vehicles. Hardware and software implementation requirements can be included during model development and then code can be automatically generated for embedded deployment. In comparison with the traditional programming approach, some of the advantages offered by model-based design are summarized below:
1 - Model-based design provides a common design environment that is much more intuitive for an engineer, the graphical approach facilitates general communication, data analysis and system verification.
2 - Engineers can apply modifications or locate and correct errors early, minimizing time spent in system design.
3 - Design reuse for upgrades and for derivative systems with expanded capabilities is facilitated.

3.2 Pixhawk® Flight Management Unit

The term "Pixhawk Flight Management Unit (FMU)" refers to an open-hardware onboard management unit suitable for a wide variety of micro air vehicles [8]. It combines in an embedded design a high-performance microcontroller, based on Cortex-M4F processor, an inertial measurement unit (IMU) which integrates 3-axis gyroscopes and accelerometers, a barometer and a set of external sensors including GNSS and magnetometers. Regarding the project of this thesis, Pixhawk® FMU was selected especially for its open-source features. Moreover, nowadays this system is an industry standard autopilot which also provides high-end functionalities to the academic community, at low costs and high availability. Table 3.1 contains some important specifications of Pixhawk® FMU and sensors can be implemented.

The flight software will follow has been developed using Matlab-Simulink® together with the toolbox Pixhawk® Pilot Support Package (PSP) [9]. The PSP feature allows users to use Simulink models to generate code targeted for the Pixhawk® flight management unit. The PSP provides the ability to incorporate the Pixhawk® toolchain for complete firmware build and download to the Pixhawk® hardware. The user needs to use blocks from the base Simulink® or possibly the Aerospace block-set for simulating flight control system models. Once the flight control system has been successfully modeled, simulated and verified, the Embedded Coder can be used to deploy it into the autopilot hardware. NuttX is the Operative System delivered with the Pixhawk® toolchain and it is used for running the code generated from the Simulink® model. The flight control software can

Figure 3.2: Pixhawk® FMU main module.

be easily adapted for various configurations of micro-RPAS and represents the vehicle's main brain. Different flight modes have been implemented inside, which use more or less complex control chains depending on the degree of automation required. Each flight mode is selectable by the pilot in real time using, for example, a switch on the transmitter.

3.3 Flight modes

Flight modes are the way the autopilot responds to pilot inputs and controls vehicle motion. Every UAV autopilot software allows the user to control the vehicle by using different flight modes, which provide different levels of autopilot-assisted flight. The number and types of flight modes programmed in an autopilot software must be related with vehicle configuration and some modes can behave differently on different flying machines. The pilot can change between flight modes by using a switch on the remote control or a ground control station. Based on the level of automation provided by the autopilot, flight modes can generally be grouped into manual, assisted and auto modes; this section provide an overview of flight modes commonly used for fixed-wing and rotary-wing class of UAVs.

3.3.1 Manual flight modes

By using manual modes, the pilot has direct control of the vehicle actuators via joystick, yoke or stick movements. In this case the autopilot provide direct pass-through of user

Processor
32bit STM32F427 Cortex M4 core with FPU
168 MHz
256 KB RAM
2 MB Flash
32 bit STM32F103 failsafe co-processor
Sensors
ST Micro L3GD20H 16 bit gyroscope
ST Micro LSM303D 14 bit accelerometer / magnetometer
Invensense MPU 6000 3-axis accelerometer / gyroscope
MEAS MS5611 barometer
Interfaces
5x UART (serial ports), one high-power capable, 2x with HW flow control
2x CAN (one with internal 3.3V transceiver, one on expansion connector)
Spektrum DSM / DSM2 / DSM-X Satellite compatible input
Futaba S.BUS compatible input and output
PPM sum signal input
RSSI (PWM or voltage) input
I2C
SPI
3.3 and 6.6V ADC inputs
Internal microUSB port and external microUSB port extension
Power system and protection
Ideal diode controller with automatic failover
Servo rail high-power (max. 10V) and high-current (10A+) ready
All peripheral outputs over-current protected, all inputs ESD protected
Peripherals
Digital airspeed sensor PX4AIRSPEED
u-Blox GPS Module
External USB port
External multicolor LED
I2C splitter

Table 3.1: Pixhawk® specifications

inputs to actuators or to an output mixer if the user input is associated to a combination of actuators. In this mode the automation level is minimum and the autopilot is also described as "transparent"; at best it can modify the type of response of vehicle movement respect to stick movement or limit it between maximum and minimum values. For these reasons this mode is generally used for Visual-Line-Of-Sight (VLOS) operations. In fixed-wing and rotary-wing UAVs, manual flight mode is used by expert pilots because it requires more skills and effort. For consumer multirotor class, a real manual flight mode is generally not available for most of them because of the piloting complexity.

3.3.2 Assisted flight modes

Assisted modes are also pilot-controlled but offer some level of automatic assistance. User inputs are not directly linked with actuators but they are interpreted by the autopilot as physical reference quantities. Depending on the automation level, a wide range of sensors is used; the autopilot triggers different feedback control chains in order to govern actuators and chase user inputs. Assisted modes often make it much easier to gain or restore controlled flight, especially during BVLOS operations. For the multirotor class, examples of assisted modes can be:
- Attitude rate flight mode:
User controls roll, pitch and yaw angle rates, throttle control is manual.
- Attitude flight mode:
User controls roll and pitch angles and yaw angle rate, throttle control is manual.
- Attitude + Altitude flight mode:
User controls roll and pitch angles and yaw angle rate, throttle controls climb/descent speed at a predetermined maximum rate.
- Speed flight mode:
User controls left-right and front-back speed over ground. Yaw input controls yaw angle rate. Throttle controls climb/descent speed at a predetermined maximum rate.

3.3.3 Auto flight modes

Auto modes are those where the autopilot doesn't require constant user inputs. In this case, paths or trajectories' data must be uploaded or generated into the embedded flight controller.
- Mission flight mode
The vehicle obeys to a pre-programmed mission sent by user using GCS, or created by a path/trajectory planner implemented in the flight control software.
- Return to home flight mode
This mode is often used for emergency. Many autopilots, equipped with GNSS, can automatically steer a vehicle to the take-off position if there is a need. This mode can be triggered automatically or by user command, technically it represents a special case of the Mission flight mode.

3.4 Software architectural overview

This section explains the most interesting parts of the flight control software. Figure 3.3 shows a graphical view of the proposed software architecture, as it has been designed in Matlab/Simulink®. The whole software is intended to be stored in the Pixhawk® FMU with the exception of the trajectory planning task, to which a separated processor is dedicated.

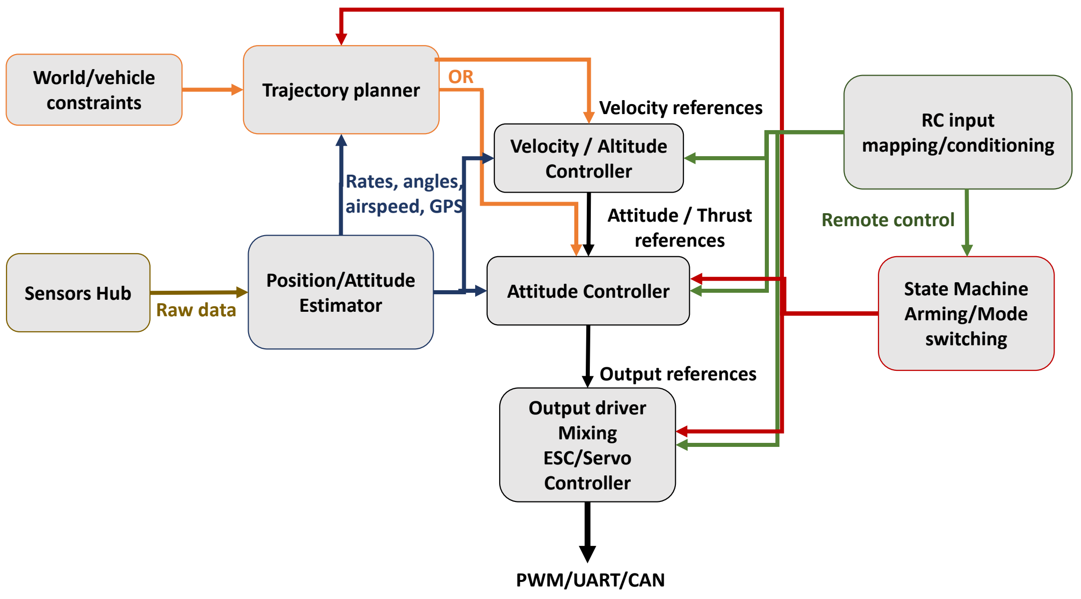

World/vehicle constraints
Trajectory planner
OR
Velocity references
RC input mapping/conditioning
Velocity / Altitude Controller
Rates, angles, airspeed, GPS
Attitude / Thrust references
Remote control
Sensors Hub
Raw data
Position/Attitude Estimator
Attitude Controller
State Machine Arming/Mode switching
Output references
Output driver Mixing ESC/Servo Controller
PWM/UART/CAN

FIGURE 3.3: The diagram provides an overview of the Flight control software architecture, it can be adapted for various UAV classes.

3.4.1 Signal conditioning

Signal conditioning is an important portion of the software that elaborates input commands sourced by remote control station. This stage is required to make user inputs suitable for processing through flight controllers, so that input signals can be compared with sensor measurements. In this software signal conditioning includes processes of range matching with saturation and converting. Signals are re-mapped in range $[-1, 1]$ by using the following general relation:

$$out = out_{min} + (in - in_{min}) * \frac{(out_{max} - out_{min})}{(in_{max} - in_{min})} \tag{3.1}$$

3.4.2 Position and Attitude Estimator

Generally one of the most crucial part of a flight control software is the algorithm for sensor fusion. This is the reason why many more or less complex estimation algorithms were tested and adopted in the UAV field over the years [10, 11]. Here an Extended Kalman Filter (EKF) is proposed to perform accurate estimations. The algorithm is the non-linear version of the Kalman filter, it uses rate gyroscopes, accelerometer, compass, GPS, airspeed and barometric pressure measurements to estimate the position, velocity and angular orientation of the flight vehicle. This algorithm was not developed by the candidate, but is based on initial work documented in [12]. The advantage of the EKF over simpler algorithms like complementary filter or Direction Cosine Matrix (DCM) algorithm, is the increased ability to reject measurements with significant errors by fusing all available measurements. This makes the vehicle less susceptible to faults which can affect a single sensor. The EKF algorithm implemented is available as a built-in App in the Nuttx OS of Pixhawk®, it can estimate a total of 22 states with the underlying equations derived in [13].

3.4.3 Trajectory planning

Trajectory planning consists of generating time sequences of reference inputs $q(t)$ to the inner motion control system which ensures that vehicle performs the planned trajectory. Trajectory planning is often referred to as path planning, but actually these are two different concepts. A ”path” is a geometric description of motion, it denotes a locus of points in space which the vehicle has to follow. Trajectory, however, refers to a path on which a time law is specified at each point in terms of different variables like, for instance, velocities, accelerations or angles. The vehicle motion can be restricted by several limitations that must be considered and imposed as constraints into the definition of the planning problem; typically, these can be initial and final conditions, mechanical constraints and obstacle constraints. In this thesis, Chapter 4 presents an efficient methodology of trajectory generation which can be applied to multirotor vehicles and Chapter5.4 shows how the same approach can be extended to multirotor fleets. As will be seen, the strategy used for trajectory planning requires a not-negligible computing power and for this reason it has been implemented and tested separately in a different hardware connected with Pixhawk® autopilot; it can be nevertheless considered full-fledged part of the software architecture.

3.4.4 Flight control

Flight controllers represent the main processing stage in the software. Proportional-Integral-Derivative (PID) control technique dominates in multirotor application in particular for its feasibility and simplicity. The tuning of PID gains can be tackled based upon the system parameters if they can be achieved or estimated precisely. On the other hand, if the system parameters are unknown, appropriate PID gains can be designed just based on the system tracking error and treating the system like a black-box.

This fact gives chance to easily adapt a given PID control architecture to vehicles having widely different dynamics. This Section presents one by one the structure of the software's flight controllers, which are organized as a cascade control system [14]. In this architecture there are generally a primary controller and a primary dynamics which are components of the outer loop. A secondary controller loop is designed as part of the outer loop. Set-points of the inner loop are calculated by the outer loop, hence the name "cascade control". For good performances the inner loop should represent a significantly faster dynamics with respect to the outer loop; this assumption allows restraining interaction that can occur between them and improve stability characteristics. Therefore, a higher gain in the inner loop can be adopted. An additional advantage is that the non-linear features of the plant are generally handled by the controller in the inner loop without having meaningful influence on the outer loop [5]. Thanks to the application of a cascade control structure, the PID control strategy can be adapted for controlling complex dynamics like rotary-wing aerial vehicles. In sections below each flight controller implemented into the software is graphically and analytically explained. Finally, control performances are evaluated setting up a simulation environment which uses the dynamics of a DJI® F550 hexa-rotor. The multirotor dynamics is designed in line with the six degree-of-freedom model, represented by equations (5.5), (2.30), (2.31) and (2.32) derived in Chapter 2. During simulations the model state feeds back to the controller without any additional noise.

3.4.4.1 Attitude controller

The attitude controller represents the innermost control chain in the software and gives the basic piloting level of the multirotor vehicle. This controller is always active into the software since no real manual flight modes are permitted for steering multirotors, it would require too much effort for the user. Following the same philosophy as the others, the attitude controller is implemented as a cascade control system, figure 3.4 shows the design in detail.

The outer loop is based on the Euler angles ϕ and θ and exploits attitude estimations calculated by the extended Kalman filter, here just a simple proportional gain is used. The inner loop deal with angular velocities in a PI-D scheme in which the D-term processes only the derivative of the feed-back signal (output) instead the error signal. The PI-D solution is more suitable in practical implementation because it prevents violent controller reactions in case of step change in the input signal.

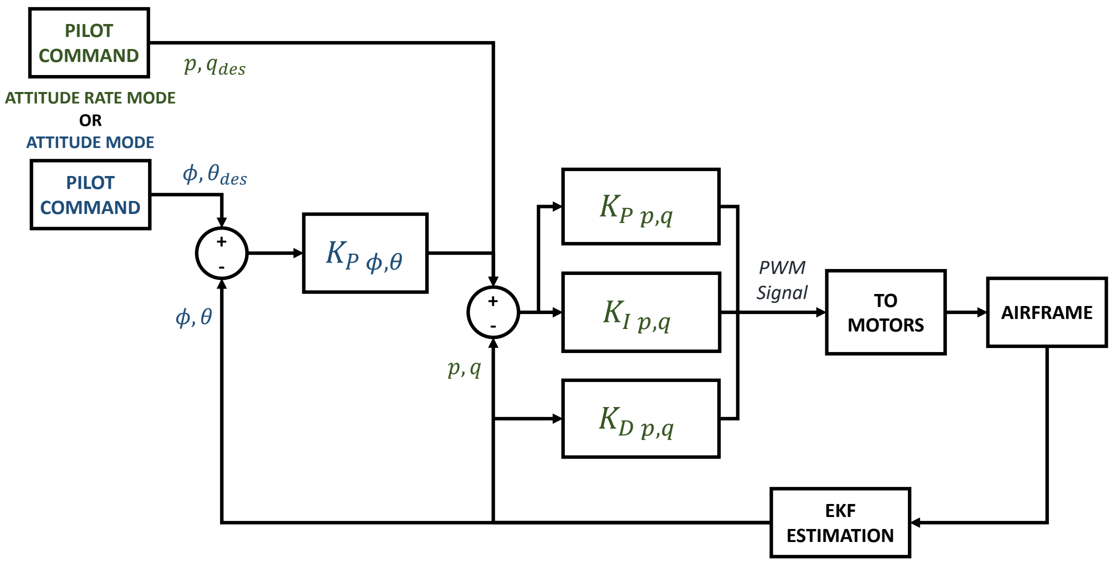


FIGURE 3.4: Attitude controller architecture.

By defining the following error signals:

$$\begin{cases} \epsilon_\phi = \phi_{des} - \phi \,, \\ \epsilon_\theta = \theta_{des} - \theta \,, \\ \epsilon_p = p_{des} - p \,, \\ \epsilon_q = q_{des} - q \,, \end{cases} \tag{3.2}$$

The cascade P-PID control output can be resumed as follows:

$$\begin{cases} p_{des} = K_{P\phi} \epsilon_\phi \\ q_{des} = K_{P\,\theta}\, \epsilon_\theta \\ \tau_{\phi(pwm)} = K_{P\,p}\, \epsilon_p - K_{D\,p}\, \dot{p} + K_{I\,p} \int_0^t \epsilon_p \, dt \\ \tau_{\theta(pwm)} = K_{P\,q}\, \epsilon_q - K_{D\,q}\, \dot{q} + K_{I\,q} \int_0^t \epsilon_q \, dt \end{cases} \tag{3.3}$$

The attitude controller allows users to select two different flight modes depending on whether the aim is to control angles or angular velocities. These modes are called ATTITUDE MODE or ATTITUDE RATE MODE.

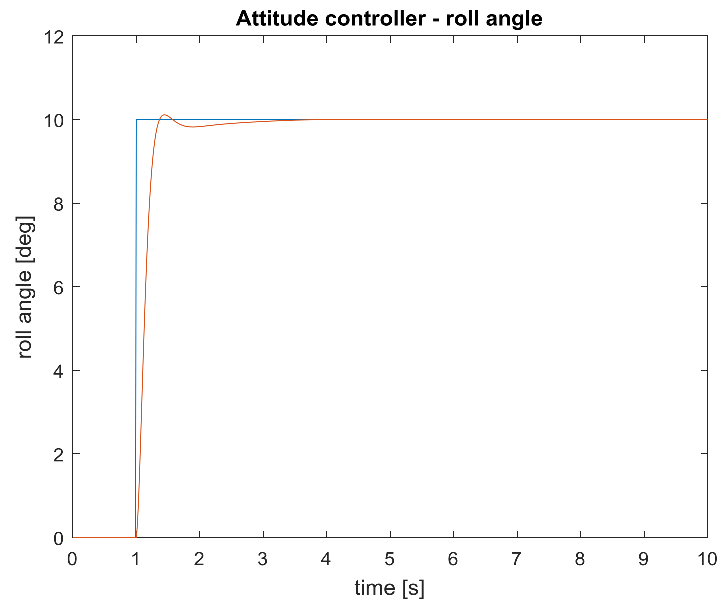


FIGURE 3.5: Response of roll angle to step change of roll angle set-point vs time.

3.4.4.2 Heading hold controller

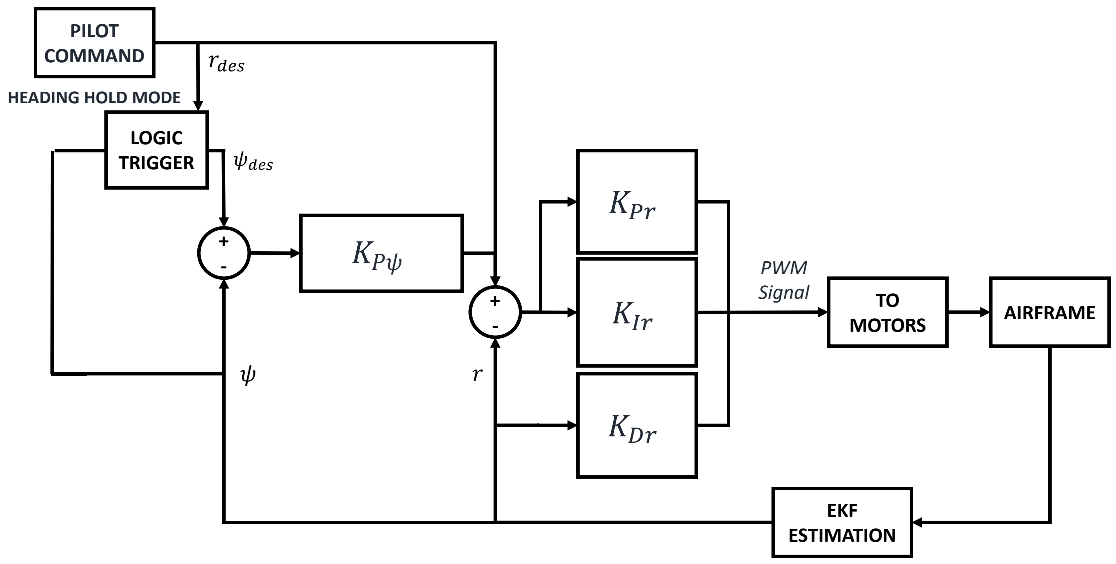


FIGURE 3.6: Heading hold controller architecture.

Figure 3.6 shows the design of the heading hold controller. The scheme is very similar to that seen for attitude control, the only difference appears in dealing the user's input signal. In fact heading hold controller does not provide direct control of the real heading angle ψ, that would be inappropriate for piloting. r_{des} represents the real input variable controlled by the user, the ψ_{des} reference for the outer loop is obtained by the current value of ψ_{des} whenever the user's command returns to zero. The strategy of triggering the outer loop only for as long as user's command is null is used to ensure a stable and robust heading control in presence of disturbances on the airframe. By defining the following error signal:

$$\begin{cases} \epsilon_\psi = \psi_{des} - \psi \\ \epsilon_r = r_{des} - r \end{cases} \tag{3.4}$$

The cascade P-PID control output can be resumed as follows:

$$\begin{cases} r_{des} = K_{P\,\psi}\,\epsilon_\psi + r_{in} \\ \tau_{\psi(pwm)} = K_{P\,r}\,\epsilon_r - K_{D\,r}\,\dot{r} + K_{I\,r}\displaystyle\int_0^t \epsilon_r\,dt \end{cases} \tag{3.5}$$

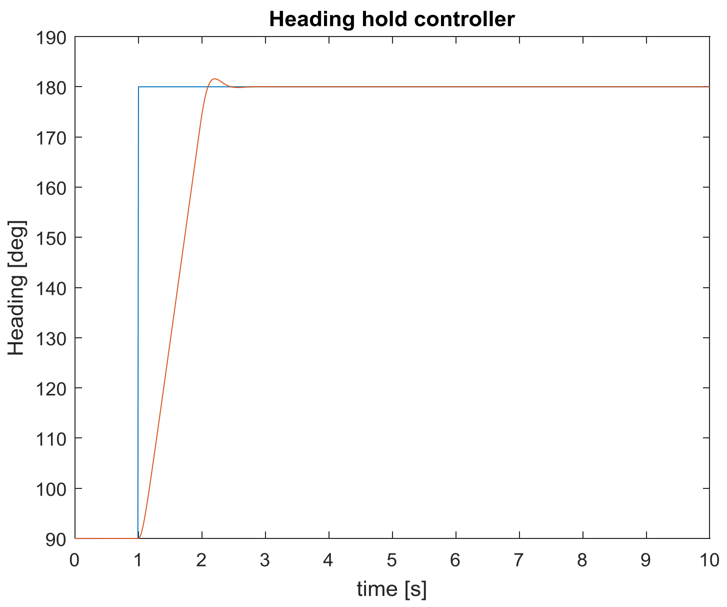


FIGURE 3.7: Response of heading angle to step change of heading set-point vs time.

3.4.4.3 Altitude hold controller

Altitude hold controller allows the user to control multirotor vertical speed and/or vertical acceleration acting on the total force applied by motors. Two possible modes are available for this controller, the THROTTLE MODE and the ALTITUDE MODE. Using the first one gives the user direct control of the multirotor total thrust, this mode is considered manual because there is no presence of feedback signal from any autopilot sensors. By selecting the ALTITUDE MODE, the user input is interpreted as vertical speed reference; figure 3.8 shows how three cascade control loops are used forming a P-P-PID control chain. In order to maintain a fixed altitude and ensure good performance in presence of disturbance, the same strategy of the heading hold controller is used, in

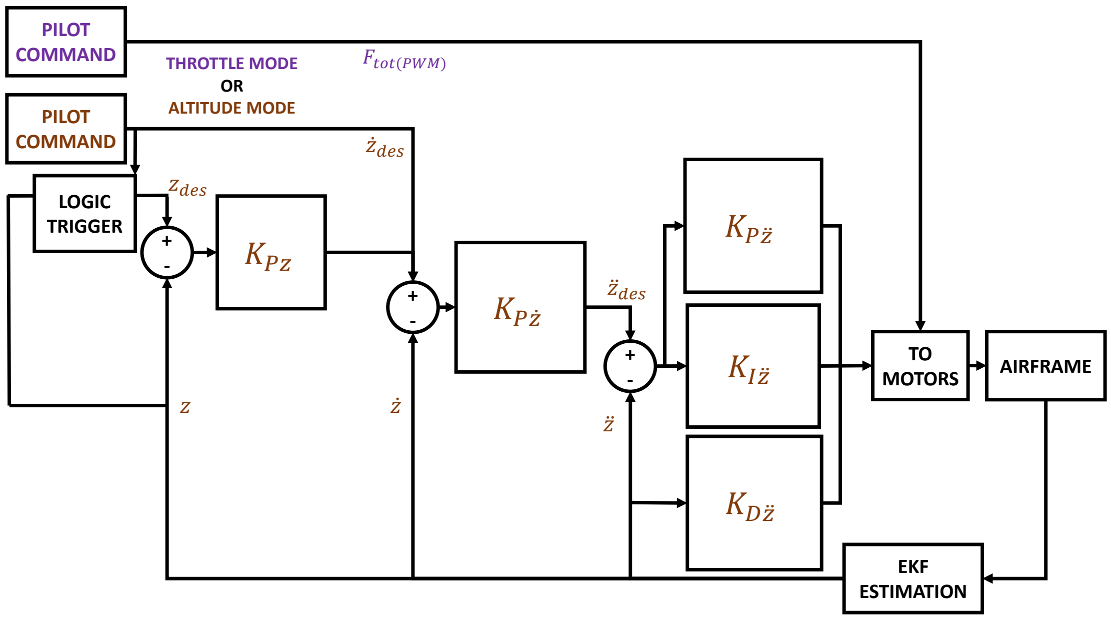


FIGURE 3.8: Altitude hold controller architecture.

which a new reference altitude is stored and triggered whenever the user's command drop to zero. By defining the following error signal:

$$\begin{cases} \epsilon_z = z_{des} - z \\ \epsilon_{\dot{z}} = \dot{z}_{des} - \dot{z} \\ \epsilon_{\ddot{z}} = \ddot{z}_{des} - \ddot{z} \end{cases} \tag{3.6}$$

The cascade P-P-PID control output can be resumed as follows:

$$\begin{cases} \dot{z}_{des} = K_{P\,z}\,\epsilon_z + \dot{z}_{in} \\ \ddot{z}_{des} = K_{P\,\dot{z}}\,\epsilon_{\dot{z}} \\ F_{tot(pwm)} = K_{P\,\ddot{z}}\,\epsilon_{\ddot{z}} - K_{D\,\ddot{z}}\,\dddot{z} + K_{I\,\ddot{z}} \int_0^t \epsilon_{\ddot{z}}\,dt \end{cases} \tag{3.7}$$

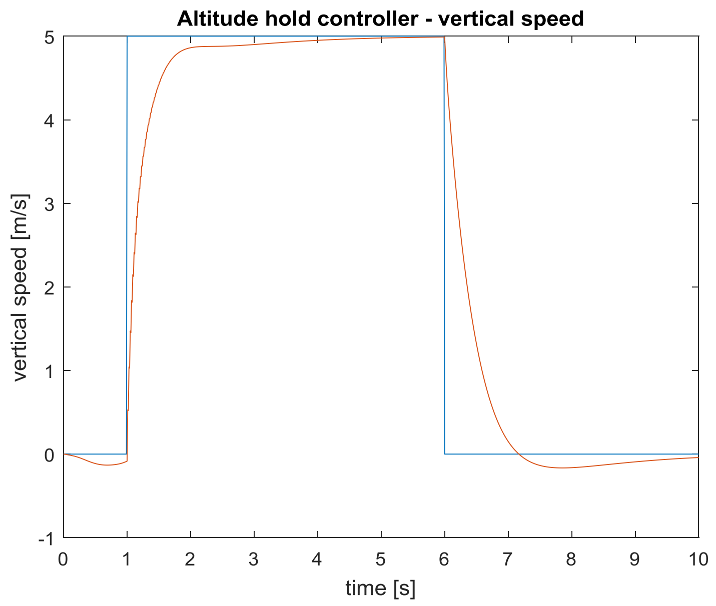


FIGURE 3.9: Response of vertical speed to step change of vertical speed set-point vs time.

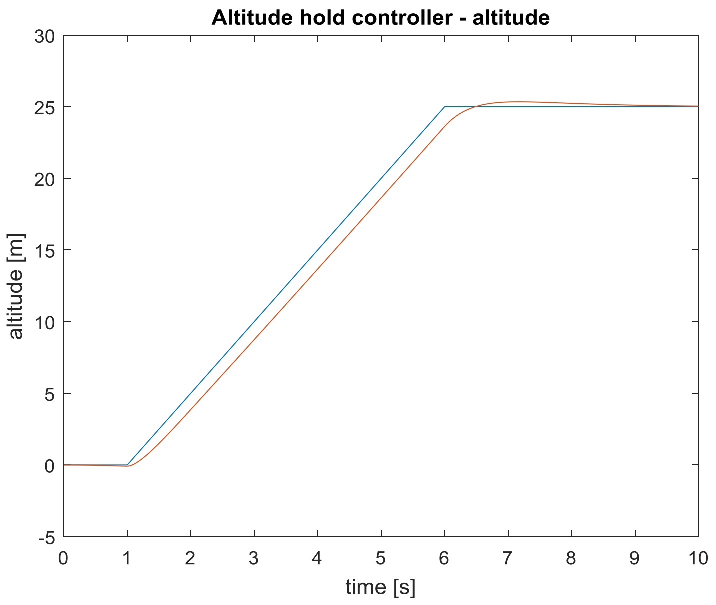


FIGURE 3.10: Response of altitude to step change of altitude set-point vs time.

3.4.4.4 Velocity controller

A very useful way to control an UAV is by speed commands. This section shows how the velocity controller is modeled inside the flight control software for a rotary-wing UAV, figure 3.11 shows the design in detail. The controller is part of the main cascade control scheme and is designed as a feed-back proportional controller that accepts forward/right velocity references $v_{f/r\,des}$ and provides desired roll/pitch angles to the inner control loop.

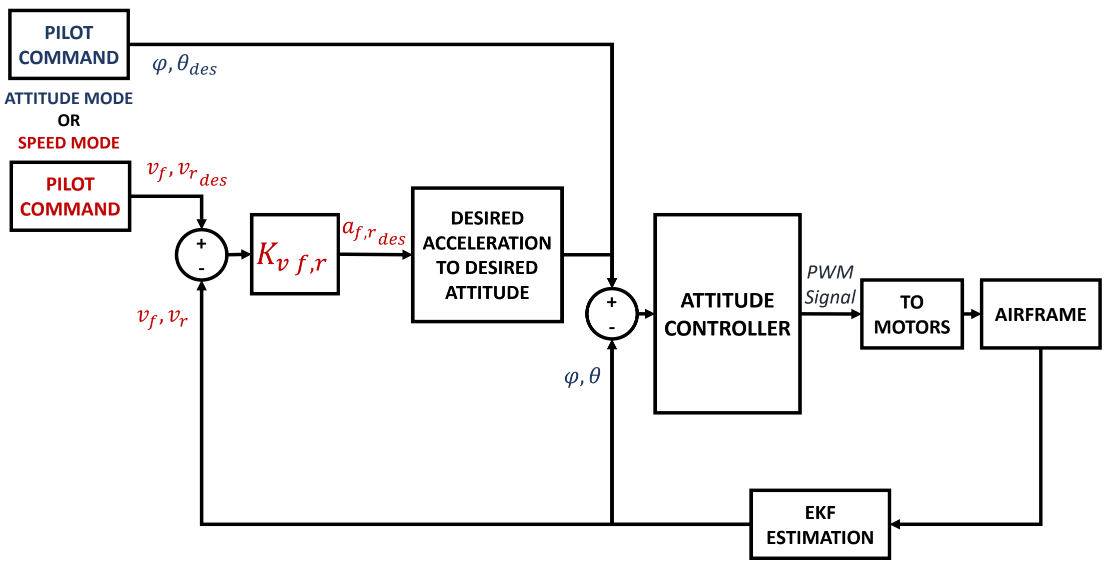


FIGURE 3.11: Velocity controller architecture.

Controller inputs and outputs are intended in the forward-right-downward coordinate frame $F^{\,c}$, which is equivalent to the vehicle frame $F^{\,v}$ after rotating by the heading angle.
The transformation matrix $R^c_v(\psi)$ is defined as follows:

$$R^c_v(\psi) = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \tag{3.8}$$

By introducing the speed errors ϵ_v:

$$\begin{bmatrix} \epsilon_{v\,f} \\ \epsilon_{v\,r} \end{bmatrix} = R^c_v(\psi) \begin{bmatrix} u_{des} - u \\ v_{des} - v \end{bmatrix} = \begin{bmatrix} v_{f\,des} - v_f \\ v_{r\,des} - v_r \end{bmatrix} \tag{3.9}$$

a simple proportional controller can be implemented inside the software with this form:

$$\begin{cases} a_{f\,des} = K_{vf}\,\epsilon_{v\,f} \\ a_{r\,des} = K_{vr}\,\epsilon_{v\,r} \end{cases} \tag{3.10}$$

In which $K_{v1,2}$ are gains. Since multirotors are under-actuated vehicles, controlling speeds or accelerations in body-fixed horizontal plane implies to impose references in total thrust and attitude angles or angle rates. Desired accelerations $a_{f\,des}$ and $a_{r\,des}$ must be expressed as function of the desired attitude angles needed for increasing airspeed.

Starting from the simplified equations presented in (2.38), it is possible to write:

$$\begin{cases} a_{f\,des} = -f\ \cos\phi\sin\theta\,, \\ a_{r\,des} = f\ \sin\phi\,, \\ a_{d\,des} = -f\ \cos\phi\cos\theta + g\,, \end{cases} \tag{3.11}$$

By imposing $a_{d\,des} = 0$, the following relations can be found for ϕ_{des} and θ_{des}:

$$\begin{cases} \phi_{des} = \arctan\dfrac{a_{r\,des}\cos\theta_{des}}{g} \\ \\ \theta_{des} = -\arctan\dfrac{a_{f\,des}}{g} \end{cases} \tag{3.12}$$

By referring to figure 3.11, equations (3.12) are stored inside the ACC. TO ATTITUDE block. The velocity controller is active only if the appropriate flight mode, here called SPEED MODE, is selected by the user.

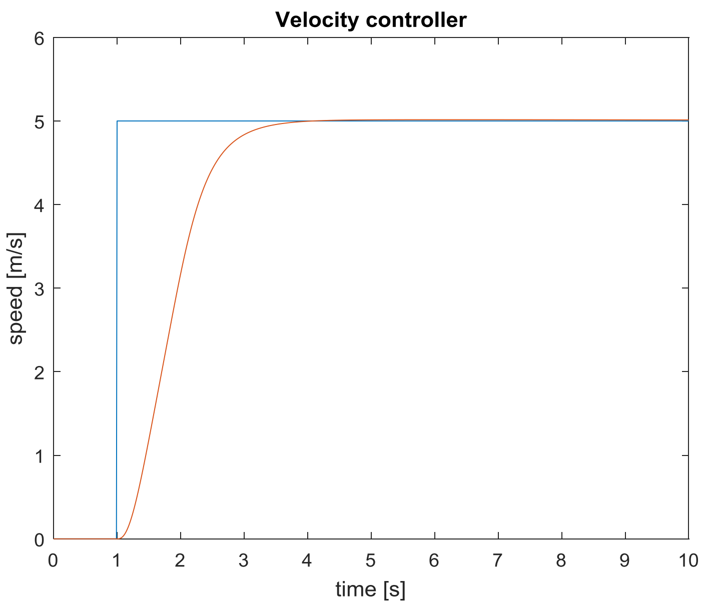


FIGURE 3.12: Response of translational speed (x-axis) to step change of speed set-point vs time.

3.4.5 Motor mixing

Motor mixer, also called output mixer, represents the final stage of the processed signals inside the software. This part depends on multirotor configuration you have to control, which means the number of motors and the geometrical layout. It is basically a table which specifies the magnitude of forces that should be applied to each motor to control multirotor's movements (pitch, roll, yaw and total thrust). It is easy to understand that an inappropriate motor mix can severally affect or undermine completely the upstream control chain and PID tuning. The design of a proper motor mixing is simple once you know the location of each motor in relation to the center of mass. Recalling equations (2.20) and (2.21) for motor thrust and torque, it is possible to write the following relation:

$$\begin{bmatrix} \delta_{pwm(1)} \\ \delta_{pwm(2)} \\ .. \\ \delta_{pwm(n)} \end{bmatrix} = M_{mix} \begin{bmatrix} \tau_{\phi(pwm)} \\ \tau_{\theta(pwm)} \\ \tau_{\psi(pwm)} \\ F_{tot(pwm)} \end{bmatrix} \tag{3.13}$$

in which n is the number of motors and M_{mix} is the motor mix matrix. For example, a proper motor mix matrix for a quad-rotor symmetrical cross configuration (Figure 2.1) is defined as follows:

$$M_{mix} = \begin{bmatrix} l\frac{\sqrt{2}}{2} & l\frac{\sqrt{2}}{2} & 1 & 1 \\ -l\frac{\sqrt{2}}{2} & l\frac{\sqrt{2}}{2} & -1 & 1 \\ -l\frac{\sqrt{2}}{2} & -l\frac{\sqrt{2}}{2} & 1 & 1 \\ l\frac{\sqrt{2}}{2} & -l\frac{\sqrt{2}}{2} & -1 & 1 \end{bmatrix} \tag{3.14}$$

3.4.5.1 Output saturation

Pratically, motor command signals δ_{pwm} are expressed as values of the duty cycle of the PWM motor drivers. Before sending signals to motor regulators (ESCs), it is necessary programming a saturation step that prevents certain side-effects caused by physical features of the actuators:

1. Each PWM output cannot be more than a maximum value corresponding to 100% of duty cycle.
2. Each PWM output cannot be less than the minimum value that stops the propeller; this is required to avoid multirotor flip.
3. The constraints above must not affect the displacement imposed by motor mixer, otherwise control could not work well or not work at all.

3.5 Flight test

In order to confirm the performance of the flight control architecture, some experimental tests are performed by using a DJI® F550 hexa-rotor with take-off mass $m = 1.47$ kg and arm length $l = 0.46$ m. This is the same vehicle modeled for simulations. The vehicle is equipped with the Pixhawk® autopilot in which the proposed software is directly deployed by using the Pixhawk® Pilot Support Package toolbox. The autopilot unit performs estimation and control tasks in real-time with a frequency set to 250 Hz, the data is captured and stored in microSD at a sampling rate of 10 Hz. Control gains are trimmed in empirical way starting from values used in simulations. During flight tests simple maneuvers are performed by the pilot in different flight modes, so that it is possible to engage all the controllers by enforcing various reference commands. Figures below show a comparison among reference and actual values for the most significant quantities.

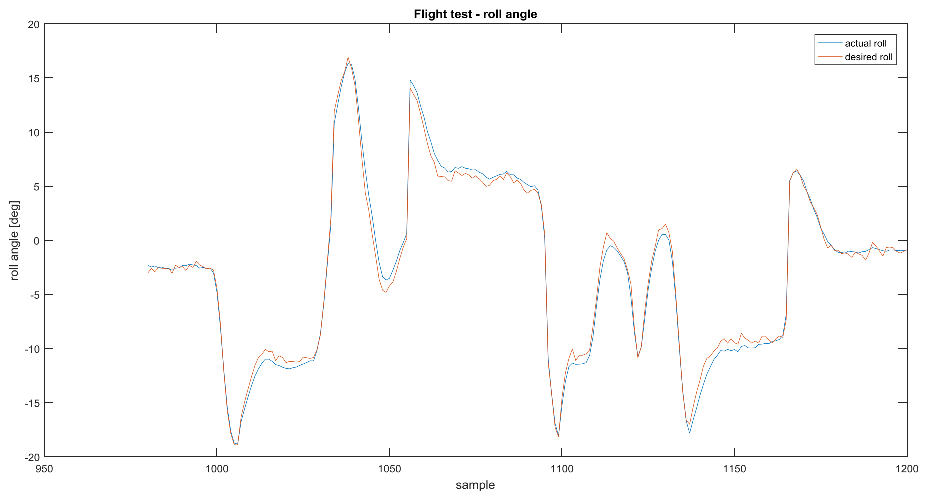


FIGURE 3.13: Flight test - Performance of attitude controller.

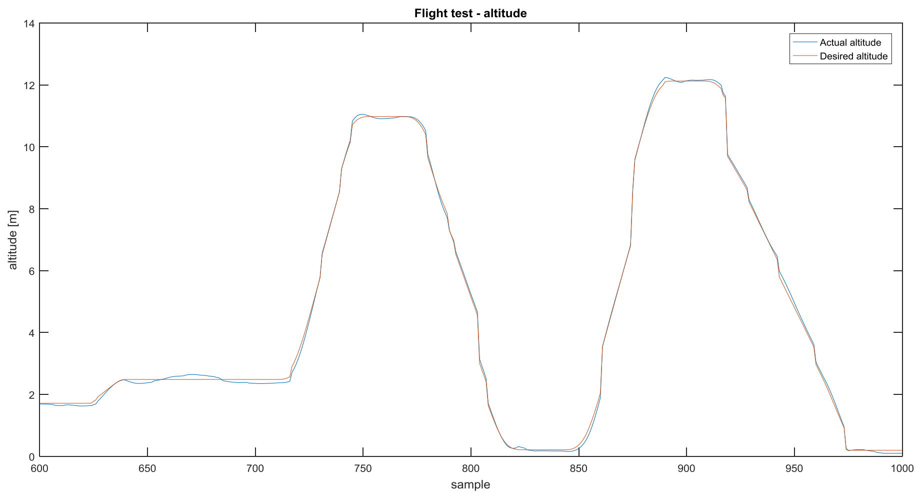


FIGURE 3.14: Flight test - Performance of altitude hold controller.

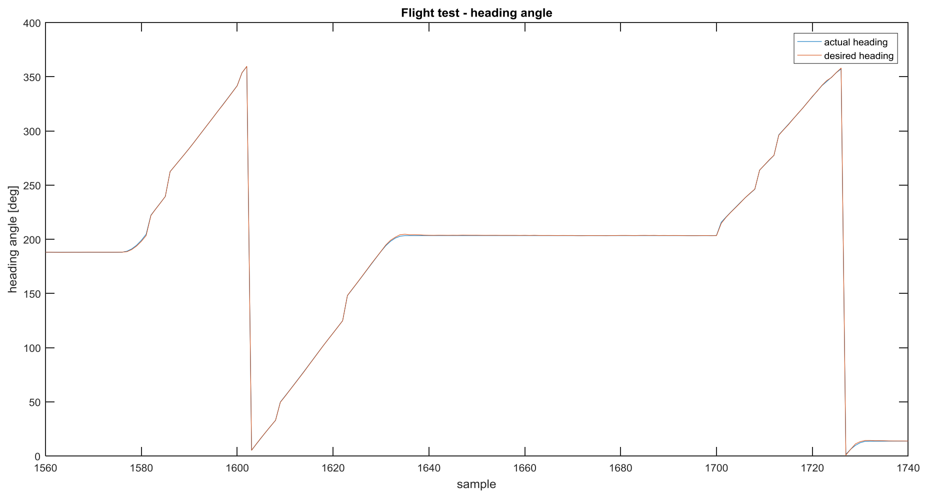


FIGURE 3.15: Flight test - Performance of heading hold controller.

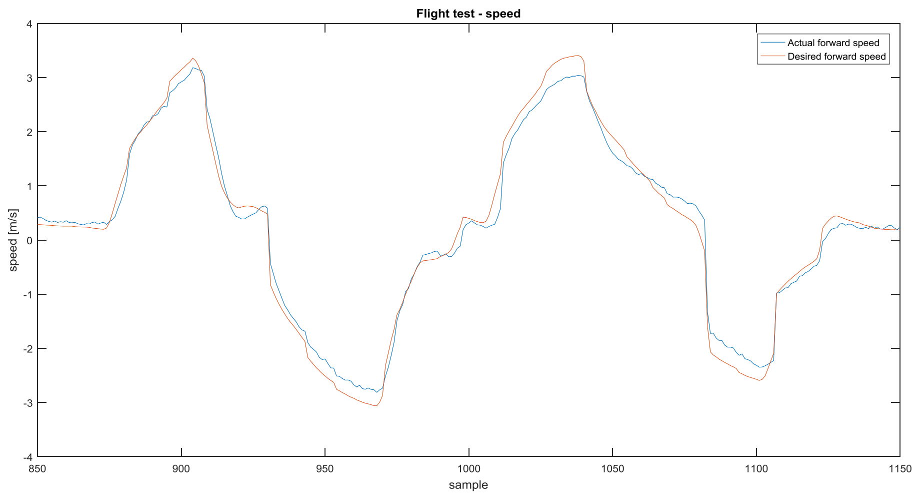


FIGURE 3.16: Flight test - Performance of velocity controller.

Chapter 4

Optimal multirotor navigation

This section explains an innovative methodology for computing optimal autonomous navigation for multirotor vehicles in obstructed environments. This approach is adopted from early work described in [15, 16]. The research work presented in this chapter has been made in collaboration with MECO (Motion Estimation, Control Optimization) Research Team of KU Leuven University, Division PMA (Production engineering, Machine design and Automation) [17]. First the general methodology is presented. Afterwards it is applied to the multirotor navigation case.

4.1 General methodology

The presented methodology aims to solve a navigation problem in the form of optimization problem, in which constraint satisfaction is guaranteed over the entire time horizon. The approach suits for systems that are differentially flat and admit a polynomial representation of flat outputs and, below right assumptions, can be adapted for the case of multirotor vehicle. The optimization problem with parameterization of the flat output will be presented in a general form in subsection 4.1.1. After that, in subsection 4.1.2 the concept of differential flatness for the multirotor case is explained in a bit more detail.

4.1.1 Optimal motion problem

The considered motion problem searches for trajectories $q(\cdot)$ which steer a system from an initial condition, at $t = 0$, to a terminal condition, at $t = T$. Both conditions are expressed as conditions on q and its derivatives $q^{(j)}$. Optimal trajectories are obtained by minimizing an objective J while respecting constraints h over the considered time horizon $[0, T]$. These represent constraints on input and states and include actuator limitations and obstacle avoidance constraints. If the system is flat, initial conditions, final conditions, input and state constraints can be expressed as conditions on q and its derivatives $q^{(j)}$. The motion planning problem generally can be translated in an optimization problem of the following form:

$$
\begin{aligned}
\underset{q(\cdot)}{\text{minimize}} \quad & J(q) \\
\text{subject to} \quad & q^{(j)}(0) = q_0^{(j)}\,, \quad j \in \{0,\dots,r\}\,, \\
& q^{(j)}(T) = q_T^{(j)}\,, \quad j \in \{0,\dots,r\}\,, \\
& h(q,t) \geq 0\,, \quad \forall t \in [0,T]\,.
\end{aligned}
\tag{4.1}
$$

In order to solve the optimization problem above, two challenges have to be faced: Firstly, the problem (4.1) is infinite dimensional because $q(\cdot)$ is a function representing an infinite set of optimization variables. Secondly, constraints on $q(\cdot)$ have to be enforced and guaranteed at all time instances. In order to tackle both issues, the trajectories $q(\cdot)$ are approximated as piece-wise polynomials and are parameterized in B-spline basis [18]:

$$
\hat{q}(t) = \sum_{l=1}^{n} \mathsf{q}_l b_l(t) = \mathsf{q}^T b(t)\,, \tag{4.2}
$$

with B-spline basis $b = [b_1, \dots, b_n]^T$ and B-spline coefficients, also called control points, $\mathsf{q} = [\mathsf{q}_1, \dots, \mathsf{q}_n]^T$, which become the new optimization variables.

The main reason for adopting the B-spline basis is the so-called convex hull property: as the B-splines are positive and sum up to 1, a spline is always contained in the convex hull of its B-spline coefficients. For completeness, such property is reported below as written in [18]:

Convex hull property *Let q be a polynomial spline of order k with knot vector t. From the non-negativity, partition of unity and local support property of the B-spline basis it follows immediately that the segment $q(t)$, $t[t_i, t_{i+1}]$ lies within the convex hull of its control points $c_{ik+1}, ..., c_i$.*

This way, bounds on a spline function can be enforced by imposing them on the coefficients:

$$
\mathsf{q} \geq 0 \Rightarrow \hat{q}(t) \geq 0\,, \forall t \in [0,T]\ . \tag{4.3}
$$

Because derivatives, anti-derivatives and any polynomial function of a spline are splines as well, also polynomial constraints on spline trajectories and their derivatives and anti-derivatives can be relaxed in the same way.

Using the spline parameterization (4.2) and constraint relaxation (4.3) allows to translate problem (4.1) into a non-linear program which generates trajectories with guaranteed satisfaction of constraint h at all time instances. It requires however to find a set of trajectories $q(\cdot)$ that characterizes the motion of the system and from which state and input trajectories can be determined. Furthermore, it should be possible to reformulate constraints on state and inputs as polynomial constraints in q, its derivatives and anti-derivatives. This is however possible for many vehicle systems including considered multirotor vehicles.

4.1.2 Differential flatness of multirotor model

Differential flatness is a property of systems in which the state and control inputs can be expressed as functions of the flat output and its time derivatives. Aside from all linear and controllable systems, also many nonlinear systems are differentially flat, as the one shown below related to a tri-dimensional multirotor. Writing the output q as following:

$$q = F(x, u, \dot{u}, \ddot{u}, ...) \tag{4.4}$$

q is a flat output if there exist smooth functions F_x and F_u such that:

$$x = F_x(q, \dot{q}, \ddot{q}, ...) \tag{4.5}$$

and:

$$u = F_u(q, \dot{q}, \ddot{q}, ...) \tag{4.6}$$

Considering the following multirotor model already shown in Section 2:

$$\text{Model:} \begin{cases} \ddot{x} &= -f \;\cos\phi \sin\theta \\ \ddot{y} &= f \;\sin\phi\,, \\ \ddot{z} &= -f \;\cos\phi \cos\theta + g \\ \dot{\phi} &= \omega_r \\ \dot{\theta} &= \omega_p \end{cases} \tag{4.7}$$

Defining inputs and outputs as shown below:

$$\text{Inputs:} \begin{cases} u_1 &= f \\ u_2 &= \omega_r \\ u_2 &= \omega_p \end{cases} \tag{4.8}$$

$$\text{Flat outputs:} \begin{cases} q_1 &= x \\ q_2 &= y \\ q_3 &= z \end{cases} \tag{4.9}$$

It can be demonstrated that the system is flat [19]. This means that both state and input equations can be expressed as functions of the specified flat output and its time derivatives:

$$\begin{cases} \ddot{x} = \ddot{q}_1 \\ \ddot{y} = \ddot{q}_2 \\ \ddot{z} = \ddot{q}_3 \\ \dot{\phi} = \arctan \dfrac{\ddot{q}_2}{\sqrt{\ddot{q}_1^2 + (\ddot{q}_3 - g)^2}} \\ \dot{\theta} = \arctan \dfrac{\ddot{q}_1}{\ddot{q}_3 - g} \end{cases} \tag{4.10}$$

$$\begin{cases} f = \sqrt{\ddot{q}_1^2 + \ddot{q}_2^2 + (\ddot{q}_3 - g)^2} \\ \omega_r = \dfrac{\dddot{q}_2(\ddot{q}_1^2 + \ddot{q}_2^2 + (\ddot{q}_3 - g)^2) - \ddot{q}_2(\ddot{q}_1 \dddot{q}_1 + \ddot{q}_2 \dddot{q}_2 + (\ddot{q}_3 - g)\dddot{q}_3)}{(\ddot{q}_1^2 + \ddot{q}_2^2 + (\ddot{q}_3 - g)^2)\sqrt{\ddot{q}_1^2 + (\ddot{q}_3 - g)^2}} \\ \omega_p = \dfrac{\ddot{q}_1 \dddot{q}_3 - (\ddot{q}_3 - g)\dddot{q}_1}{\ddot{q}_1^2 + (\ddot{q}_3 - g)^2} \end{cases} \tag{4.11}$$

4.2 Point-to-point multirotor navigation

In the navigation problem considered in this section, a multirotor is steered from an initial condition towards a terminal condition which are expressed as equality constraints on the initial and terminal positions $\xi_0 = [x_0, y_0, z_0]^T$ and $\xi_T = [x_T, y_T, z_T]^T$, roll and pitch angles ϕ, θ and their derivatives $\dot{\phi}$, $\dot{\theta}$. The objective is formulated as

$$J = \int_0^T \|\xi(t) - \xi_T\|_1 dt\,, \tag{4.12}$$

where $\xi = [x, y, z]^T$ represents the quad-rotor's position. This objective function will steer the multirotor as close as possible to the destination throughout the control horizon.

The multirotor is subject to bounds on its thrust acceleration, roll and pitch angles and their derivatives:

$$\begin{aligned} &f_{\min} \leq f \leq f_{\max}\,, \\ &\phi_{\min} \leq \phi \leq \phi_{\max}\,, \quad \theta_{\min} \leq \theta \leq \theta_{\max}\,, \\ &\dot{\phi}_{\min} \leq \dot{\phi} \leq \dot{\phi}_{\max}\,, \quad \dot{\theta}_{\min} \leq \dot{\theta} \leq \dot{\theta}_{\max}\,. \end{aligned} \tag{4.13}$$

Since obstacles may arise in the multirotor's airspace, collision avoidance constraints are imposed. These are constructed by imposing the existence of a separating plane between multirotor and an obstacle [20]. Note that this construction can only separate convex shapes [21]. Suppose the multirotor's shape is represented by a sphere with radius r, while the obstacle is a convex polyhedron with vertices w_i. Demanding the separation of both shapes by a plane $\{x \in \mathbb{R}^3 | a^T x = b\}$ is achieved with the following set of constraints:

$$\begin{aligned} -a(t)^T \xi(t) + b(t) &\geq r\,, \\ a(t)^T w_i(t) - b(t) &\geq 0\,, \quad \forall i \in \{1, \ldots, n_{w_i}\}\,, \\ a_i(t)^T a_i(t) &\leq 1\,. \end{aligned} \tag{4.14}$$

In order to avoid collisions at all time, the separating line is allowed to change over time. Both $a(\cdot)$ and $b(\cdot)$, representing the normal vector to the hyperplane and the hyperplane's offset respectively, are introduced as time dependent optimization variables and are parameterized as splines. The problem of real-time obstacle detection, that is how sensing obstacles and transforming them in convex shapes which can be included into the navigation problem, is not addressed in this thesis, but it could be an interesting part of future research work. In order to use the spline parameterization and constraint relaxation described in Section 4.1.1, a set of trajectories $q(\cdot)$ has to be chosen, from which the inputs and states of the multirotor can be derived and such that constraints (4.13) and (4.14) can be reformulated as polynomial constraints in q and its (anti-)derivatives. Section 4.1.2 shows how this can be done exactly and in [16] is discussed how this can be achieved in a different manner by exploiting half angle identities. For both approaches, the obtained optimization problem is however complex and takes a rather long time to solve, because of the huge equations derived in (4.11). For this reason a different and approximating approach is proposed here by introducing the assumption of small attitude angles. Firstly the multirotor's motion is determined from the position trajectories $q = [x, y, z]^T$; this allows to formulate the thrust acceleration as:

$$f = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} - g)^2} \tag{4.15}$$

Expressions for the roll and pitch angles and their derivatives in (4.11) are elaborated using a small angle approximation, $\sin\phi \approx \phi$ and $\cos\phi \approx 1$:

$$\begin{aligned} \phi &= \frac{\ddot{y}}{\ddot{z} - g}\,, \quad \dot{\phi} = \frac{(\ddot{z} - g)\dddot{y} - \ddot{y}\dddot{z}}{(\ddot{z} - g)^2}\,, \\ \theta &= \frac{-\ddot{x}}{\ddot{z} - g}\,, \quad \dot{\theta} = \frac{-(\ddot{z} - g)\dddot{x} + \ddot{x}\dddot{z}}{(\ddot{z} - g)^2}\,. \end{aligned} \tag{4.16}$$

Constraints (4.13) are then formulated as polynomial functions in q and its derivatives by squaring the constraint on f and multiplying constraints on the roll and pitch angles and their rates by their (non-negative) denominators. Also collision avoidance constraints (4.14) are polynomial in q.

Using a small angle approximation is mainly valid in cases where smooth and gentle maneuvers are covered. This can be imposed by tightening the bounds on ϕ and θ. As will be shown in Section 4.5, deviations from computed trajectories due to modeling errors are however easily accounted for when the motion planning is performed repeatedly in receding horizon.

4.3 point-to-point navigation for multirotor considering holonomic trajectory planning

In robotics, a vehicle is considered holonomic if all the constraints that it is subjected to can be integrated to obtain positional constraints [22]. A multirotor with no tilting rotors is a non-holonomic vehicle since it is under-actuated. Anyway its model can be reduced to a holonomic vehicle model by doing three assumptions:
1 - Bandwidth of attitude dynamics is considered significantly larger than bandwidth of speed dynamics.
2 - Vehicle maintains a fixed orientation.
3 - Vehicle moves at low speed.
A holonomic vehicle in tri-dimensional space can be simply represented by a kinematics-only model:

$$\begin{cases} \dot{x} = u_1 \\ \dot{y} = u_2 \\ \dot{z} = u_3 \end{cases} \tag{4.17}$$

Therefore in this case trajectory is composed by path coordinates:

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \tag{4.18}$$

The general methodology for solving optimal motion problems can be successfully applied to the holonomic vehicle's case [20]. Due to the simplicity of the vehicle model, implemented in terms of constraints, the resulting MPC algorithm for trajectory planning can be solved at high frequency, proving to be much lighter and more numerically manageable. In conclusion, the strategy of solving the trajectory planning task for multirotors, imposing constraints derived from a holonomic vehicle model instead of multirotor model, can be considered as a good alternative for the implementation of the whole strategy into a software, especially if the computing power on-board is limited.

4.3.1 Minimum time

A variant of the optimization problem presented in (4.1) is where the final time T is a variable and the objective function is simply written as:

$$J = T \tag{4.19}$$

a classical time scaling can be applied to (4.1). The dimensionless time $\tau = \frac{t}{T}$ is used as free variable in the parameterization for the flat output instead of the time t. Consequently, the derivatives must be scaled by T and for free end-time problem (4.1) can be formulated as follows:

$$\begin{aligned} \underset{q(\cdot),T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & q^{(j)}(0) = q_0^{(j)}\,, \quad j \in \{0,\ldots,r\}\,, \\ & q^{(j)}(T) = q_T^{(j)}\,, \quad j \in \{0,\ldots,r\}\,, \\ & h(q,\tau) \geq 0\,, \quad \forall \tau \in [0,1]\,. \end{aligned} \tag{4.20}$$

Therefore, the proposed approach remains applicable to free end-time problems as well.

4.4 Motion planner

Problem presented in 4.2 and 4.3.1 can be numerically solved using OMG-Tools (Optimal Motion Generation-tools) [23], originally developed by MECO Research Team of KU Leuven. This toolbox facilitates the modeling of motion planning problems, the simulation of them and the embedding on real motion systems. The toolbox has been written in Python and uses the software CasADi as symbolic framework and interface to solvers. It also provides a library of different predefined system models and contains an extensive list of motion problems. Figures 4.1 and 4.2 show some illustrative examples of optimal trajectories solved by using OMG-Tools software, code examples are available in [23].

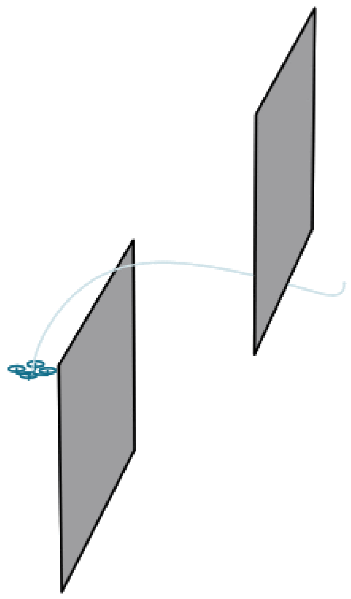


Figure 4.1: tri-dimensional quadrotor flying among obstacles.

4.5 Model predictive control strategy

Model Predictive Control (MPC) is an optimization based strategy for control of multiple-input/multiple-output (MIMO) systems. The technique uses a dynamic model of a system to solve a control problem in a receding horizon (online), without violating constraints, taking into account the most recent information about the environment. MPC combines the benefits from feed-forward and feed-back control by not only optimizing for the current state, but also for known events within the prediction horizon (e.g. changes in model parameters and reference signals). MPC is by now a well-established control approach with a vast theoretical basis. NMPC is an extension of MPC for the control nonlinear systems. It is implemented similarly as linear MPC techniques; it differs in that we obtain a non-convex control problem employing different solution strategies. NMPC solution strategies are generally computationally more challenging [24], for which often accuracy is traded for the solution time. For the case of real-time motion planning of autonomous vehicles indeed, generally vehicles operate in uncertain environments, in which obstacle positions and movements are not fully known a priori. The same is true for the motion of a real vehicle that cannot be exactly predicted using a mathematical model. This is why it is necessary to update the motion trajectory in real time, based on the most recent world information.

In order to account for disturbances, model-plant mismatches and changes in the environment, as first step it is necessary to eliminate the assumption of ideal motion control [25], which is present in the OMG-tools software. This can be done by coupling the

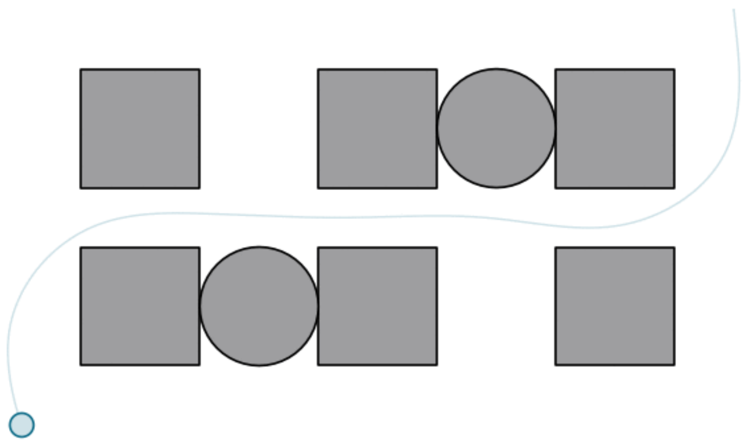


FIGURE 4.2: Two-dimensional holonomic vehicle moving in obstructed environment.

motion-planning problem with the more detailed multirotor model derived in Section 2. That model will be used in simulations to perform the calculated trajectories. A low-level control system is required to lead the vehicle model along the trajectory, the controller is explained in section 4.5.1. Figure 4.4 shows how the motion planner, the motion controller and the model are connected together to solve the navigation problem in a receding horizon. The navigation algorithm uses the update time $\Delta T : k = 0, 1, 2, ..$ and can be described by the following steps:

1. At time t_k, execute trajectory q_k.
2. At time $= t \in [t_k, t_{k+1}]$ update current state $x_k(t)$ from measurements and world information.
3. Estimate $\hat{x}_k(t_{k+1})$.
4. Solve optimization problem using $\hat{x}_k(t_{k+1})$ as initial data of vehicle dynamics and environment constraints. Obtain complete motion trajectory q_{k+1} for a granted time horizon.
5. At time $t_{k+1} = t_k + \Delta T$, vehicle's motion controllers receive trajectory q_{k+1}.
6. Repeat.

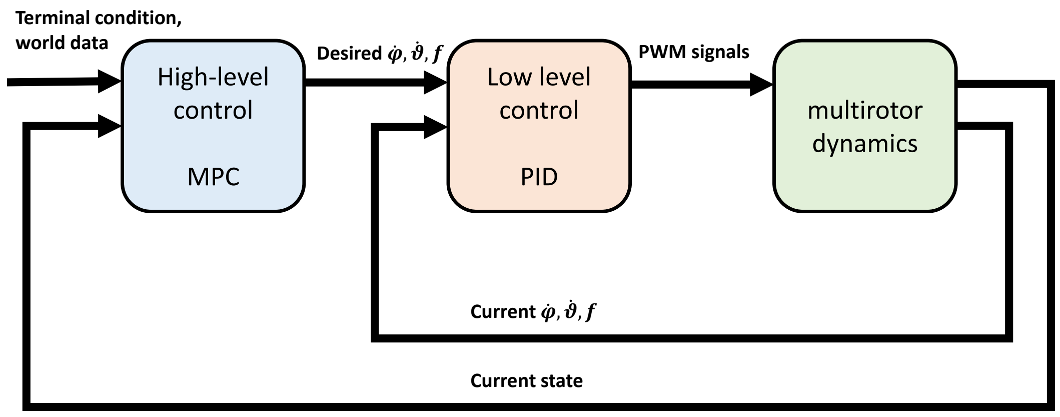

Terminal condition, world data
Desired $\dot{\varphi}, \dot{\vartheta}, f$
PWM signals
High-level control MPC
Low level control PID
multirotor dynamics
Current $\dot{\varphi}, \dot{\vartheta}, f$
Current state

FIGURE 4.3: Model predictive control scheme.

For a proper implementation of the algorithm, the update frequency has to be chosen high enough to compensate with updated data mismatches between the simplified dynamics used for motion planning and the more correct dynamics used for motion performing.

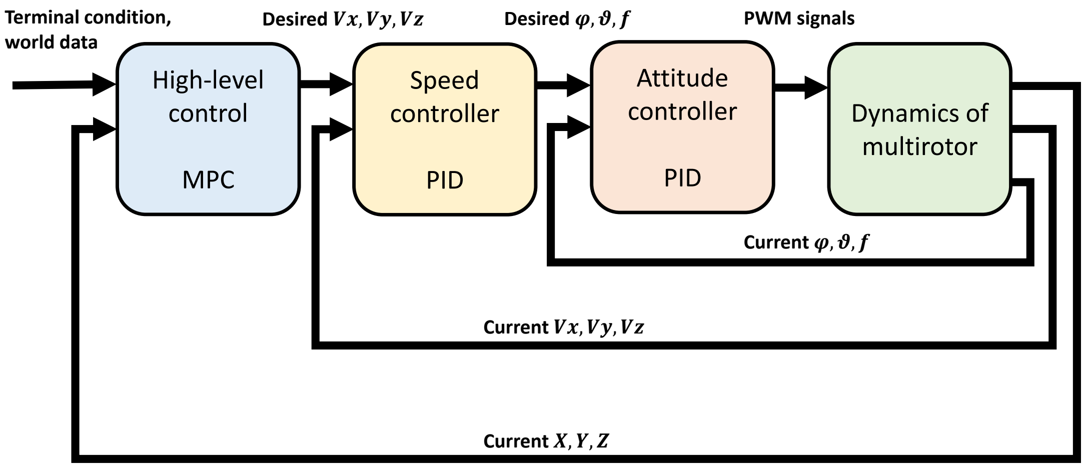


FIGURE 4.4: Model predictive control scheme considering holonomic trajectory planning.

4.5.1 Motion control

As already seen in 3.4.4.1, an inner motion control system can be used to chain trajectory planner and model, in order to chase trajectory references and perform attitude variations. However, thrust variations are directly commanded by a non-feedback controller whose effect depends on the vehicle's mass:

$$F_{tot(pwm)} = f * m \tag{4.21}$$

Alternatively, if the holonomic trajectory planning case is considered, a velocity control architecture is necessary to track the path, as the one proposed in 3.4.4.4,

4.6 Validation

The next subsections show how the MPC algorithm validation has been provided by numerical simulations as well as supported by real flight tests.

4.6.1 Numerical simulation

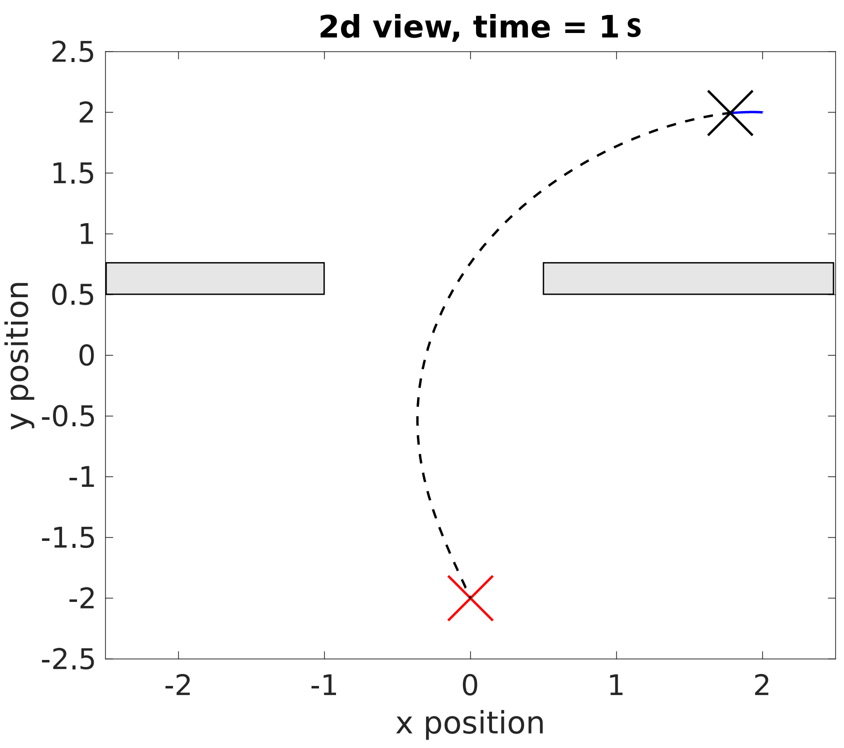


FIGURE 4.5: Quadrotor flying between two walls represented in 2D view at time $T = 1\,s$, the dashed line indicates the predicted trajectory that is corrected repeatedly.

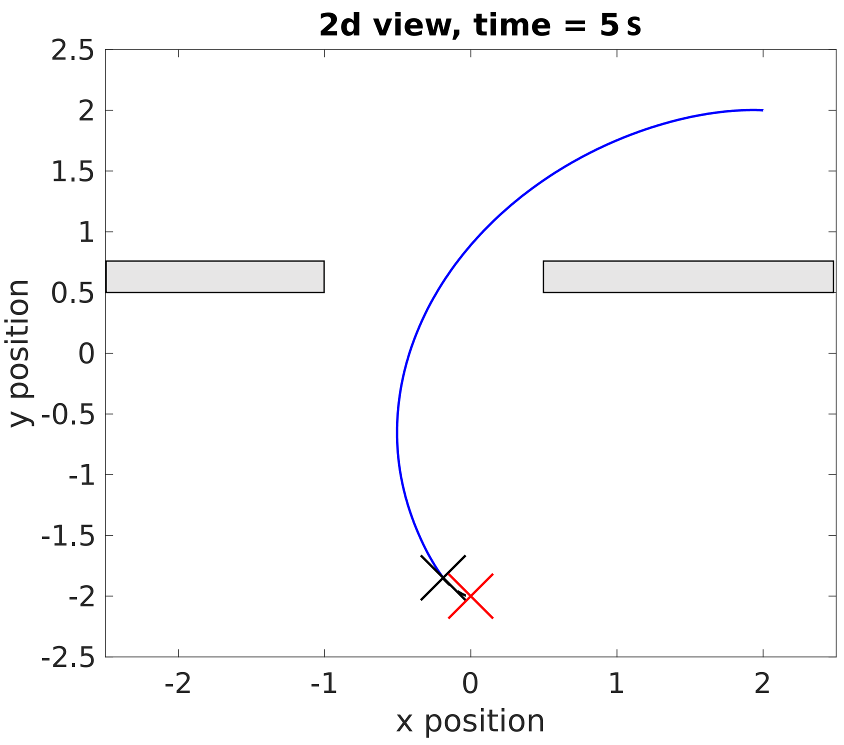


FIGURE 4.6: Quadrotor flying between two walls represented in 2D view at time $T = 5\,s$, the blue line represents the covered trajectory.

In the following example a quadrotor has to reach the imposed destination point at $[0, -2, 1]$, starting from position $[2, 2, 1]$ and avoiding two vertical walls which are included in the environment between initial and final position. The spline-based motion planning problem can be solved using the Optimal Motion Generation-tools software [23], as explained above, this is a user-friendly toolbox written in Python that uses the symbolic framework CasADi to perform nonlinear numerical optimization. In the motion planner trajectories are parameterized as cubic splines with 10 polynomial intervals, the control horizon T is set to $10\,s$. Environmental constraints, as obstacles and airspace dimensions, are added in advance in the airspace during the primary navigation problem construction. In order to obtain smooth and feasible trajectories the following model constraints are imposed:

$$\begin{aligned} & 2\,m/s^2 \leq f \leq 15\,m/s^2\,, \\ & -15\,^\circ \leq \phi \leq 15\,^\circ\,, \qquad -15\,^\circ \leq \theta \leq 15\,^\circ\,, \\ & -5\,^\circ/s \leq \dot{\phi} \leq 5\,^\circ/s\,, \qquad -5\,^\circ/s \leq \dot{\theta} \leq 5\,^\circ/s\,. \end{aligned} \tag{4.22}$$

During simulations the motion planning problem is solved with a prefixed frequency of 4 Hz, the average solving time for computing this particular point-to-point navigation problem is equal to $200ms$. It should be noted that average solving time can grow if more obstacles are added in the environment. The dynamic model uses a sample

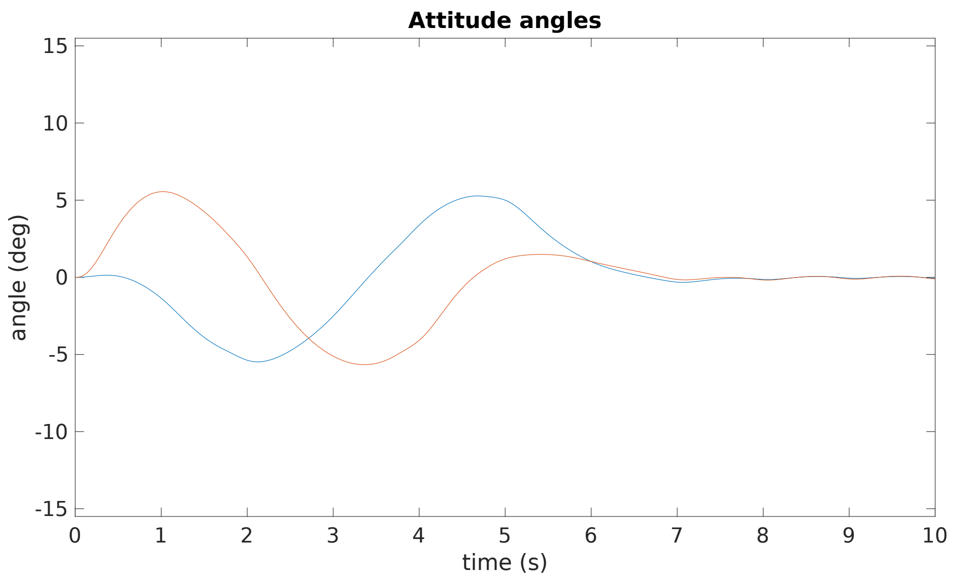


Figure 4.7: Attitude profiles during trajectory tracking; blue and red lines represent angles ϕ and θ respectively.

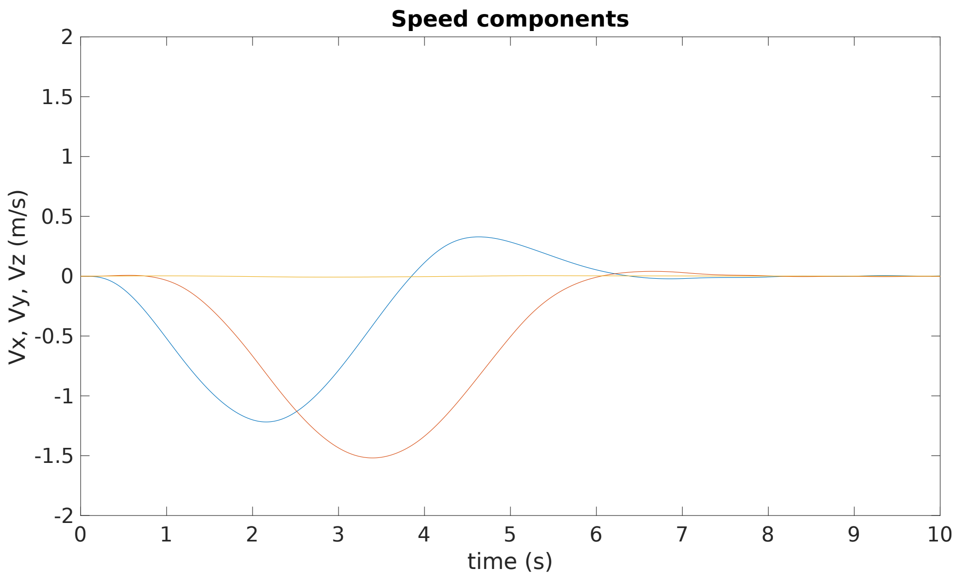


Figure 4.8: Velocity profiles during trajectory tracking; the blue, red and yellow lines represent $\dot{x}$, $\dot{y}$ and $\dot{z}$ respectively. The resultant multirotor's speed is maintained low along the path.

rate $t = 0.01s$ and the simulation is $10s$ long. The resulting motion is illustrated in Figure 4.6 and Figure 4.7, the vehicle reaches the destination avoiding obstacles and trajectories are constantly corrected with latest measured data during motion correcting models mismatch. As shown in Figure 4.7 and 4.8, attitude angles, angular rates and consequently resulting multirotor's speed are maintained low along the path thanks to the imposed constraints.

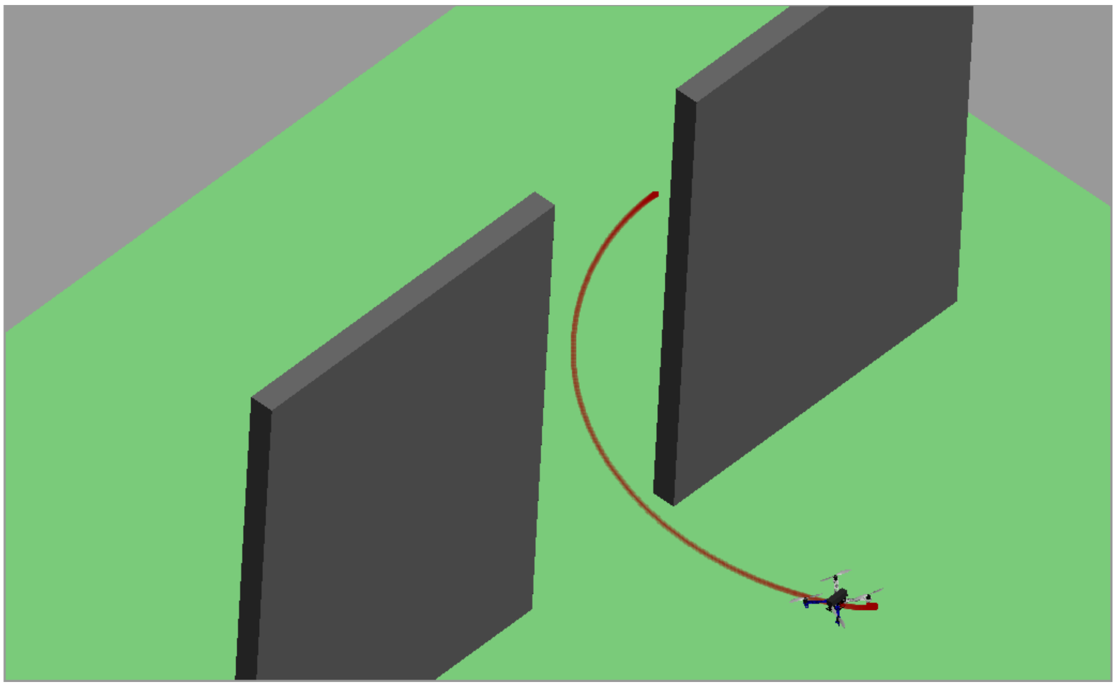


FIGURE 4.9: Quadrotor flying between two walls represented in 3D view, red line represents the covered trajectory

4.6.2 Flight tests

A flight test campaign has been led as support of the proposed strategy. As already said, the navigation algorithm is formulated in an MPC fashion in which two fundamental parts act and communicate together: a trajectory planner and a motion controller. With a view to hardware implementation, it was decided to physically separate these parts, by using different hardware elements and referring to them as trajectory planning unit and flight control unit. Reasons of this choice are two: firstly, solving the trajectory planning problem in a receding horizon requires a processing power not within reach of the flight control hardware, so it became necessary using a faster microcontroller. In addition, separating hardware which compute different tasks (trajectory planning and motion control) guarantees more safety during tests. The hardware used for trajectory planning is the Odroid®-XU4 single board computer with quad-core 2GHz A15 and RAM 2GB. The board is energy-efficient but powerful enough for running the MPC algorithm and solving optimization problems at high rates. Figure 4.10 shows the computer companion installed on-board the F550 hexa-rotor, used in flight tests. In the following example test, the vehicle is requested to fly from an initial location, also called home location, to a final destination situated $10m$ away to the east. A cylindrical obstacle with radius $0.5m$ and infinite height is present along the straight path, the obstacle centre is $5m$ away both from initial and final location. The MPC algorithm for motion planning is configured as follows:

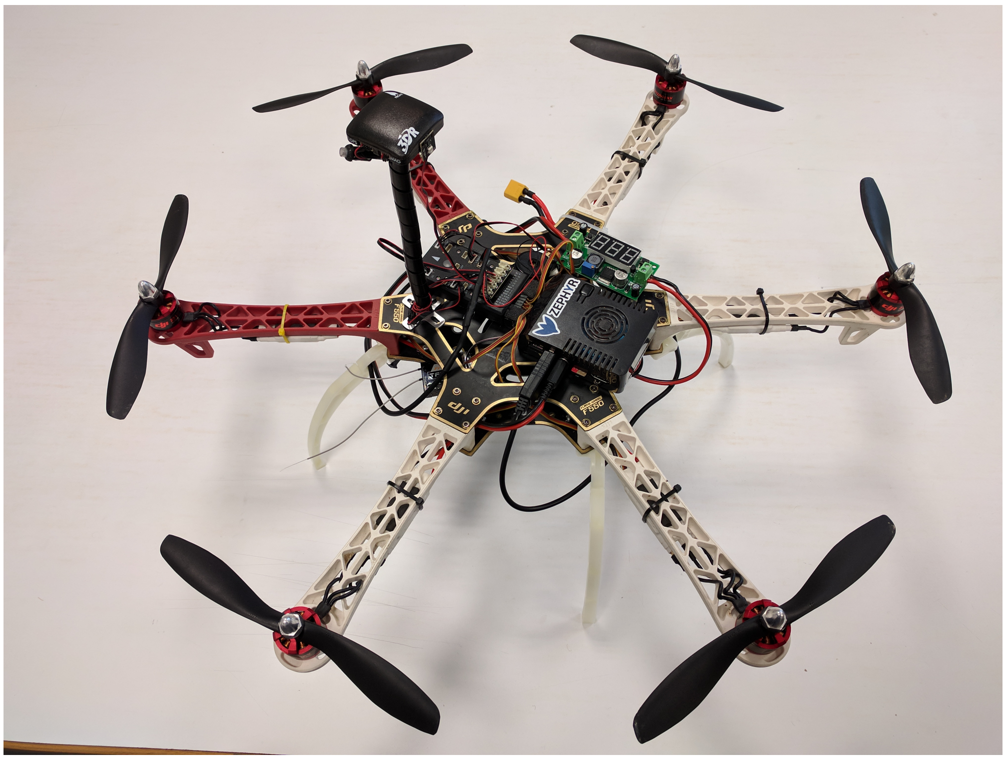


FIGURE 4.10: Multirotor used for experimental tests

1 - A holonomic 2D trajectory planning strategy is treated.

2 - The hexa-rotor is considered a holonomic 2D vehicle, circular in shape with radius $0.5m$, plus a safety distance of $0.2m$.

3 - The following model constraints are imposed:

$$\begin{aligned} 1\,m/s^2 \leq \ddot{x} \leq 1\,m/s^2\,, \quad & 1\,m/s^2 \leq \ddot{y} \leq 1\,m/s^2\,, \\ 1\,m/s \leq \dot{x} \leq 1\,m/s\,, \quad & 1\,m/s \leq \dot{y} \leq 1\,m/s \end{aligned} \tag{4.23}$$

4 - Control Horizon is set to $15s$.

5 - The algorithm's trajectory update time $DeltaT$ is set to 1 second, trajectory consists in a time sequence of 10 velocity inputs in north-east coordinates.

6 - The trajectory planner send velocity inputs to the flight control unit via Mavlink protocol [26].

With these settings, the average time requested by the computer companion for solving the trajectory planning task is $25ms$, which is in line with the chosen $DeltaT$. Relevant flight data are saved on a microSD card at 10 Hz. Figure 4.11 illustrates the flight test.

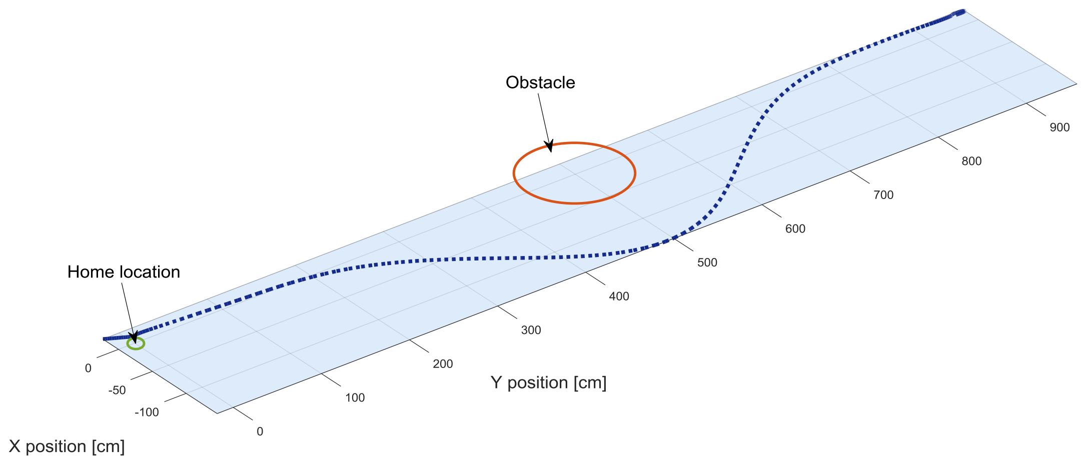


Figure 4.11: Trajectory covered hexa-rotor vehicle during flight test.

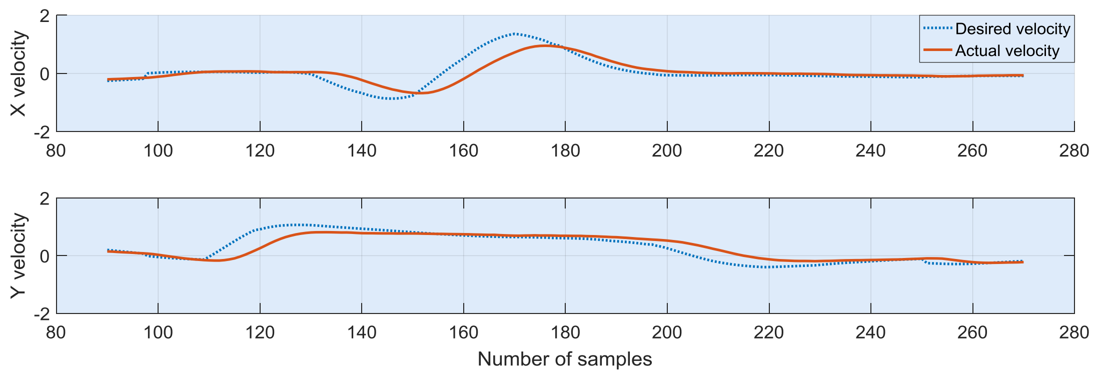


Figure 4.12: Speed during flight test.

Chapter 5

Formation flight control

The cooperative use of UAVs in civil field is very interesting for research, especially nowadays that many countries are going to approve new specific regulations for unmanned aircrafts. The operational potential of multirotor platforms can be strongly improved by making them fly within a formation. E. g. for remote sensing applications, an UAV fleet flying simultaneously may be able to quickly cover larger ground areas than a single aircraft, with the possibility to carry distributed payloads. In general, it is expected that aircraft cooperative control will play a fundamental role in future aerospace scenarios, where unmanned aerial vehicles will be required to cover large areas for monitoring with more robust and reliable results if compared to the use of single vehicle [27]. Depending on the sensing capability and the interaction topology of agents, a variety of applications and formation control problems can be found in the literature, based on the behavioral approach, [28, 29], virtual structures [30, 31], and the classic leader-follower configurations [32, 33]. In Ref. [34] a comparative study of the three different formation structures is performed and the superiority of the behavioral approach is proven for trajectory tracking and formation keeping, providing that each aircraft has knowledge of all the other state vectors. In this chapter, the case study of UAVs formation flight is treated starting with the explanation of the possible control architectures, following with two different formation control techniques which have been implemented into the experimental flight software for multirotors.

5.1 Formation flight control architectures

The choice of formation flight control architecture is crucial because it influences, of necessity, both software and hardware design. There are many possibilities and each architecture shows pros and cons in terms of performance of the control strategy and/or implementation complexity. In this section, three different formation arrangements are described, the same structures are also reported in literature, in [34, 35]. These arrangements are called Leader-wingman, virtual leader and behavioural structure. In the leader-wingman structure there is an UAV designed as leader, while the remaining vehicles are called wingmen. While the leader keeps a prescribed trajectory, the followers refer their position to the other vehicle in the formation, keeping a fixed relative distance from the neighbouring vehicles. The leader-wingman structure is widely employed in control and management of multiple vehicle formations because of its simplicity, despite it suffers of error propagation. In the virtual-leader structure each UAV receives the same leader instructions or information, that generally are the leader trajectory. Leader position may be represented by a real vehicle position or a virtual point in the formation that vehicles must track. In this case there is no error propagation effect because all vehicles refer to the same leader; moreover the formation behaviour is prescribed by simply specifying the behaviour of the virtual leader. The disadvantage is that there is no explicit feedback among the follower positions in the formation, each member has no information about its distance from the followers; so that increases chances of collision. At last, the behavioral approach considers the introduction of a virtual point called formation geometry center (FGC). The position of the FGC depends on the relative distance among vehicles, representing a sort of barycenter of the formation. Each vehicle has to maintain a prescribed distance from the FGC. This structure is good for safety because it allows vehicles to sense indirectly other vehicle movements by sensing FGC variations, increasing chances to avoid air collisions.

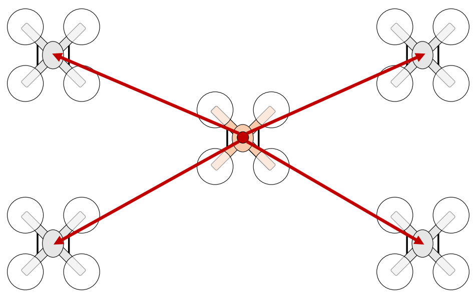


FIGURE 5.1: Leader-wingman configuration.

5.2 Formation modelling

For the purposes of the formation control strategy that will be exposed in Section 5.3, it is required to introduce a suitable UAVs' formation model, in which each vehicle is represented as a three degrees-of-freedom point mass model. This simplified model,

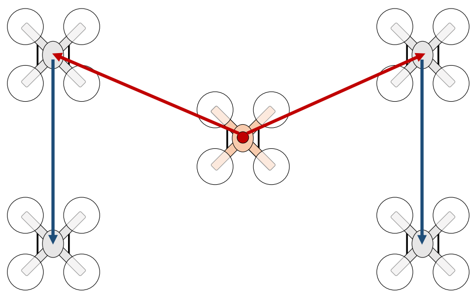


Figure 5.2: Virtual leader configuration.

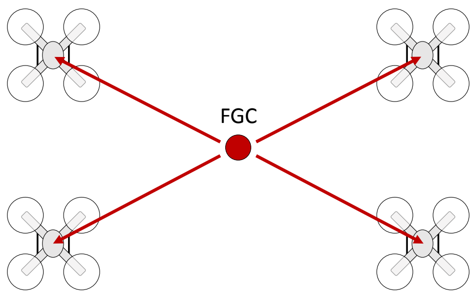


Figure 5.3: Behavioral configuration with formation geometry center.

that involves only the slow state variables, is widely used and considered proper with trajectory tracking problems and position-keeping autopilot systems design.

Assume the $i-$th vehicle is required to keep a specified distance from a reference point G, moving with velocity $\bar{V}$. By referring to the inertial frame, F^i, the following three position vectors are introduced: r_r is the position of the reference point, r_i is the current position of the $i-$th vehicle, and $\bar{r}_i$ represents its required position. Thus, as depicted

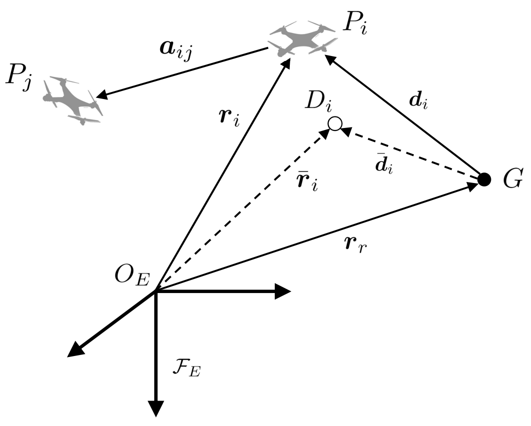


Figure 5.4: Definition of actual (P_i) and desired (D_i) positions of $i-$th multirotor.

in Fig. 5.4, d_i and $\bar{d}_i$ respectively indicate the current and required relative distance between the $i-$th vehicle and its reference point, G. The following relations are derived:

$$r_r + d_i = r_i \tag{5.1}$$

and

$$r_r + \bar{d}_i = \bar{r}_i \tag{5.2}$$

from which one obtains:

$$\bar{d}_i - d_i = \bar{r}_i - r_i \tag{5.3}$$

Provided $\dot{r}_i = V_i$ and $\dot{\bar{r}}_i = \bar{V}_i$, the time derivative of Eq. (5.3) evaluated in the $\mathsf{F}^{\,i}$ frame yields:

$$\dot{\bar{d}}_i - \dot{d}_i = \bar{V}_i - V_i \tag{5.4}$$

where V_i and $\bar{V}_i$ respectively represent the actual and the desired velocity of the $i-$th vehicle, relative to the inertial frame. Assuming the desired velocity $\bar{V}_i$ equals the velocity of the reference point $\bar{V}$, the position dynamics of the $i-$th vehicle with respect to the reference point becomes:

$$\dot{d}_i = \dot{\bar{d}}_i - \bar{V} + V_i \tag{5.5}$$

The translational dynamics of the $i-$th vehicle as expressed in $\mathsf{F}^{\,i}$ is described by the following equation:

$$\dot{V}_i = \frac{1}{m_i}\left(u_i + m_i\, g + f_{a\,i}\right) \tag{5.6}$$

where m_i is the mass of the vehicle, $g = [0\ 0\ g]^T$ is the local gravitational acceleration vector, and $f_{a\,i}$ is the aerodynamic drag. Provided ρ is the air density, a simple flat plate area model is considered such that

$$f_{a\,i} = -\frac{1}{2}\rho\left(R_b^v[A_x\ A_y\ A_z]^T\right)\|V_i\|\, V_i \tag{5.7}$$

where $\|\cdot\|$ indicates the euclidean norm and A_x, A_y, and A_z are the equivalent flat plate drag areas facing the three body-frame axes of the $i-$th agent. The total thrust vector $u_i = [u_{x_i}\ u_{y_i}\ u_{z_i}]^T$, directed along $z_{B\,i}$ and pointing upwards, represents the only control input to Eq. (5.6).

5.3 Potential strategy for formation flight

This section describes an analytical and experimental framework addressing formation control of multirotor aircraft with collision avoidance capability, developed in close co-operation with the all team of Flight Mechanics Laboratory. This strategy does not deal

with path generation and, in order to be successfully applied, it requires the implementation of a preventive path planning step in the software used for experimental tests, as the one seen in Chapter 4. The proposed strategy applies to a formation of cooperating vehicles under two assumptions:
1- Position data are available for all the agents by use of broadcast transmission.
2- Standard autopilot loops have been closed around each agent in the formation, controlling attitude variables.
The proposed formation control law is able to fulfill simultaneously three different tasks:
1- Trajectory tracking
2- Position keeping
3- Mutual collision avoidance
In order to accomplish the collision avoidance task, a potential function $V_p(a) : \mathcal{A} \to \mathbb{R}$ is defined:

$$V_p(a) = \frac{K_a}{4} \sum_{i,j} \left[\frac{\left(\bar{a}_{ij} - a_{ij}\right)^T \left(\bar{a}_{ij} - a_{ij}\right)}{a_{ij}^T a_{ij} - r^2} \right]^2 \tag{5.8}$$

In which $K_a > 0$, $a_{ij} = r_j - r_i = d_j - d_i$ and $\bar{a}_{ij} = \bar{r}_j - \bar{r}_i = \bar{d}_j - \bar{d}_i$ represent the distance and the desired distance respectively between the $i-$th and the $j-$th vehicle of the formation. Assume a sphere with radius $r > 0$ is centered at P_j and represents a safety zone for the $j-$th vehicle. To avoid collision, the $i-$th agent is requested not to enter the $j-$th sphere. The set of distance constraints which the above potential function takes into account is defined as follows:

$$\mathcal{A}_{ij} = \left\{ a_{ij} \in \mathbb{R}^3 \mid \|a_{ij}\| > r \right\} \tag{5.9}$$

The potential function (5.8) guarantees that each term a_{ij} satisfies the constraint in Eq. (5.9), the following properties can be proven [36]:

1. $V_p(\bar{a}) = 0$
2. $V_p(a) > 0$ for all $a \in \mathcal{A} \setminus \{\bar{a}\}$
3. $V_p(a)$ has a global minimum at $a = \bar{a}$.

About last property, the gradient $\nabla V_p|_{a_{ij}}$ defined as:

$$\nabla V_p|_{a_{ij}} = -K_a \left[\frac{\left(\bar{a}_{ij} - a_{ij}\right) \|\bar{a}_{ij} - a_{ij}\|^2}{\left(a_{ij}^T a_{ij} - r^2\right)^2} + \frac{a_{ij} \|\bar{a}_{ij} - a_{ij}\|^4}{\left(a_{ij}^T a_{ij} - r^2\right)^3} \right] \tag{5.10}$$

becomes null only at $a = \bar{a}$, which means $\bar{a}$ represents a global minimum.

5.3.1 Control design

A suitable control action is derived to guarantee that each multirotor in the formation has the following dynamics:

$$\dot{V}_i = K_V \left(\bar{V} - V_i\right) + K_d \left(\bar{d}_i - d_i\right) + \sum_j \boldsymbol{\nabla} V_p|_{a_{ij}} \tag{5.11}$$

To get the required control efforts, a dynamic inversion approach is employed. More precisely, the commanded acceleration in Eq. (5.11) is set equal to the actual acceleration in Eq. (5.6) and a solution is found for the control input u_i. By disregarding the effect of aerodynamic drag [37], one obtains:

$$u_i = m_i \left[-f_g + K_d \left(\bar{d}_i - d_i\right) + K_V \left(\bar{V} - V_i\right) + \sum_j \boldsymbol{\nabla} V_p|_{a_{ij}} \right] \tag{5.12}$$

The control input u_i represent the input of the inner control loop closed around each vehicle in the formation. Multirotor vehicles are generally controlled by using total thrust and attitude references, so it is necessary to transform u_i in terms of these inputs. In particular, the total thrust magnitude is given by $T_i = \|u_i\|$, while attitude commands have to be computed from the orientation of u_i with respect to the local vertical. Define $\xi_i \in \mathbb{R}$ as the angle between u_i and the unit vector directed along z_{Hi}, pointing upwards. It is

$$\xi_i = \cos^{-1}\left(-\frac{u_{z_i}}{T_i}\right) \tag{5.13}$$

Note that there are infinite attitudes of F^b with respect to F^v for which z_{Bi} is aligned with u_i. Hence, in order to specify a unique attitude, it is possible to select a pre-defined yaw angle equal to zero. This is equivalent to take into consideration the attitude represented by the principal Euler angle/axis (ξ_i, e_i), where $e_i \in \mathbb{R}^3$ is the unit vector given by

$$e_i = \frac{u_i \times z_{Hi}}{\|z_{Hi} \times u_i\|} \tag{5.14}$$

From (ξ_i, e_i), the attitude of F^b with respect to F^v can be finally expressed in terms of Euler angles or direction cosine matrix [38]. Global asymptotic stability of the closed-loop control system can be proven by means of the Lyapunov method, as it is showed in [36].

5.3.2 Numerical simulation

The proposed control technique is evaluated on the basis of computer simulations performed in the Matlab-Simulink environment. A formation made of an array of quadrotor vehicles is considered in simulations, where all vehicles always are requested to fly at the same altitude while following a prescribed trajectory. The non-linear 6-DoF mathematical model, presented in Chapter 2, represents each multirotor dynamics, in which the additional effect of aerodynamic drag is considered. All relevant simulation parameters are listed in Table 5.1. All vehicles are assumed identical; each one has four identical

TABLE 5.1: Simulation parameters

Parameter	**Symbol**	**Value**	**Units**
Vehicle data			
Mass	m	1.32	kg
Moments of inertia	$J_x,\ J_y$	$9.75 \cdot 10^{-3}$	kg m^2
Moment of inertia	J_z	$1.93 \cdot 10^{-2}$	kg m^2
Arm length	l	0.19	m
Equivalent flat plate areas	$A_x,\ A_y$	$1.32 \cdot 10^{-2}$	m^2
Equivalent flat plate area	A_z	$6.44 \cdot 10^{-2}$	m^2
Gravitational acceleration	g	9.81	m s^{-2}
Air density	ρ	1.225	kg m^{-3}
Formation control			
Trajectory tracking control gains	$K_{V_x},\ K_{V_y},\ K_{V_z}$	12	s^{-1}
Position keeping control gains	$K_{d_x},\ K_{d_y},\ K_{d_z}$	5	s^{-2}
Safety zone radius	r	0.3	m

electrical motors driving fixed-pitch propellers with paired spin directions that generate propulsive forces and moments. The internal attitude controller is based on proportional-integral-derivative contributions tuned to make the attitude dynamics have a bandwidth significantly larger than the bandwidth of the formation control dynamics[37]. The efficiency of the controller in simultaneously performing trajectory tracking, formation geometry keeping, and collision avoidance was evaluated time after time by adding disturbance forces on vehicles. In order to evaluate the efficiency of the controller in the presence of non-modeled dynamics, constraints are set on the maximum total thrust magnitude, namely $T_i \in [0,\ 2\,mg]$, and on the commanded inclination, $\xi_i \in [0, 45]$ deg. A fixed time delay, $t_d = 0.02$ s, is also implemented which accounts for local processing and communication issues between the agents of the formation. Finally, the additional effect of aerodynamic drag is considered according to the model in Eq. (5.7). In the present framework, the moment generated by $_{ai}$ about the center of mass of the $i-$th multirotor is disregarded.

Two simulation test cases are analyzed. In the first case the robustness of the proposed controller against disturbances is evaluated and the contribution of the collision avoidance controller to formation keeping is investigated. In the second test case the stability of the controller and its capability of performing collision avoidance task are tested in a scenario where both formation acquisition and formation reconfiguration would lead the agents to repeatedly cross their trajectories.

5.3.2.1 Case 1

All vehicles are requested to fly at the same altitude while following a rectilinear trajectory. In particular, the quadrotors leave their initial hovering positions $r_1(0) = [-3 \ -2 \ -2]^T$ m, $r_2(0) = [-4 \ 3 \ -2]^T$ m, and $r_3(0) = [-1 \ 1 \ -2]^T$ m at time $t = 0$ s and follow a reference point identified by the position $r_r(t) = [\lambda t \ 0 \ -2]^T$ m, where $\lambda = 0.5$ m/s and $t \geq 0$ s. Formation geometry is defined by $\bar{d}_1 = [0 \ -0.5 \ 0]^T$ m, $\bar{d}_2 = [0 \ 0.5 \ 0]^T$ m, and $\bar{d}_3 = [0 \ 0 \ 0]^T$ m. Collision avoidance control gain is $K_a = 1$ m^2 s^{-2}. Figure (5.5) shows the trajectory of vehicles leaving their initial positions while

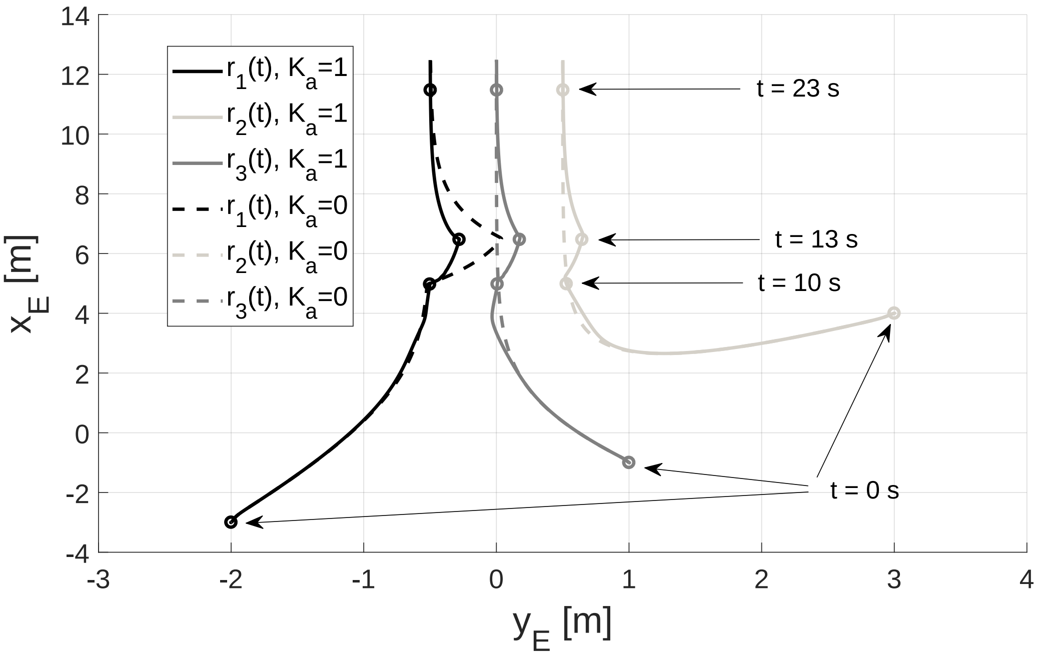


FIGURE 5.5: Formation trajectories over the x_E-y_E plane (Case 1).

circle markers indicate the formation configuration at discrete time instants. Also, to assess the robustness of the proposed controller against disturbances, an inertially-fixed constant force, $f_d = [0 \ 5 \ 0]^T$ N, is applied to vehicle 1 for $t \in [10, 13]$ s. Dashed lines in Fig. (5.5) indicate the trajectories followed when $K_a = 0$. In such a case, formation geometry is not maintained since vehicles 2 and 3 are not influenced by the state of vehicle 1. Also, the trajectories of vehicles 1 and 3 intersect when $t = 13$ s, leading to a collision. Solid lines represent the case when $K_a \neq 0$ and each agent can get the position information of all the other vehicles. In this configuration the line formation is preserved during the collision avoidance maneuver (as indicated by circle markers in Fig. (5.5)) and the maximum lateral displacement of vehicle 1 is reduced with respect to the case when $K_a = 0$. In Figure (5.6) relative distances $\|a_{12}\|$, $\|a_{23}\|$, and $\|a_{31}\|$ are plotted as a

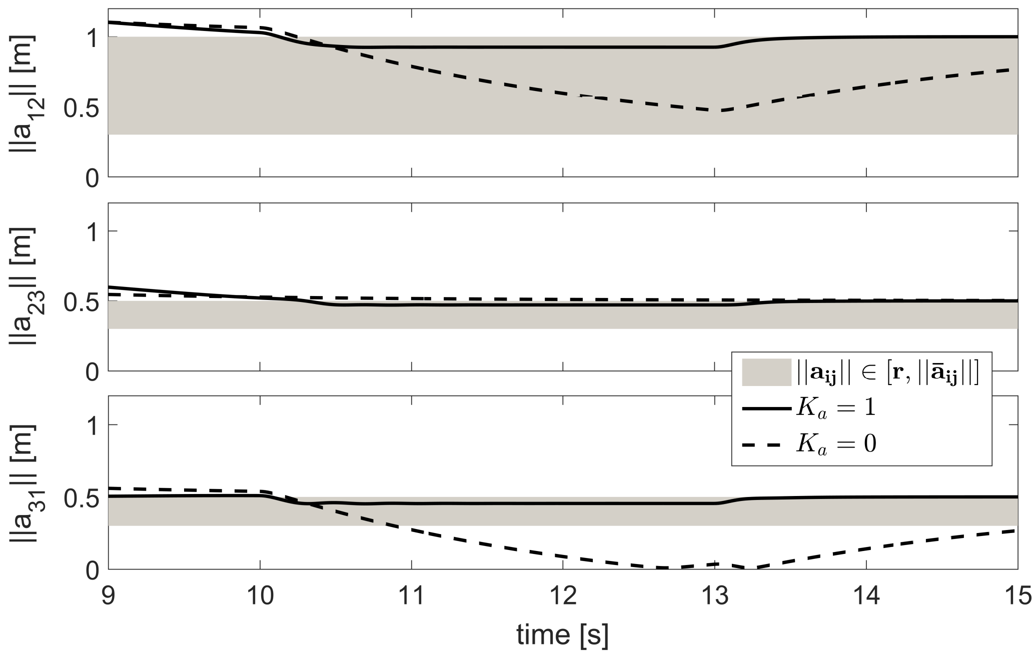


FIGURE 5.6: Mutual distances between agents of the formation (Case 1).

function of time. The gray zone is upper-limited by the desired value $\|\bar{a}_{ij}\|$, which is the prescribed value for $\|a_{ij}\|$, and lower-limited by the radius of the safety zone, r. Note that, in this particular scenario, gray zones related to $\|a_{23}\|$ and $\|a_{31}\|$ are narrower than the case related to $\|a_{12}\|$, thus showing a higher collision risk. In fact, when no collision avoidance control is performed, the dashed line representing $\|a_{31}\|$ almost goes to zero, thus implying the above-mentioned collision between vehicles 1 and 3. Constraints are instead respected when $K_a \neq 0$. In Figure (5.7) the norm of the error vectors $\|eta_1\|$, $\|eta_2\|$, and $\|eta_3\|$ are reported as a function of time. It is evident how, after the initial transient, during which formation acquisition is performed, the external disturbance determines a deviation of the error variables from the equilibrium condition. It can be noted that a steady-state error is reached for $t \geq 13$ s: this is due to the not modeled effect of the aerodynamic drag in the design phase of the controller. As a matter of fact, the final error remains bounded, with $\|delta_i\| \to 0.013$ m and $\|epsilon_i\| \to 6.5 \cdot 10^{-6}$ m/s for $i = 1, 2, 3$ and $t \to \infty$. As a final remark, commanded thrust magnitude and attitude signals are reported in Fig. 5.8 for each vehicle, with particular focus on the effect of external disturbance. It is evident how, for $t \in [10, 13]$ s, the electric motors are required to provide more thrust in order to keep the vehicles in the desired positions

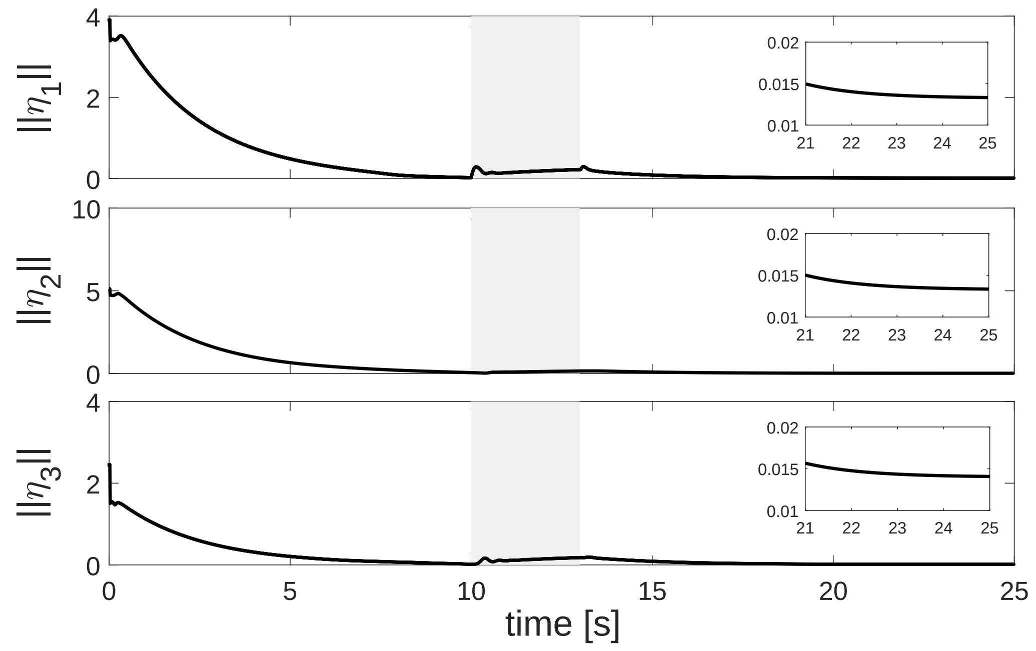


FIGURE 5.7: Stability of the formation under external disturbances (Case 1).

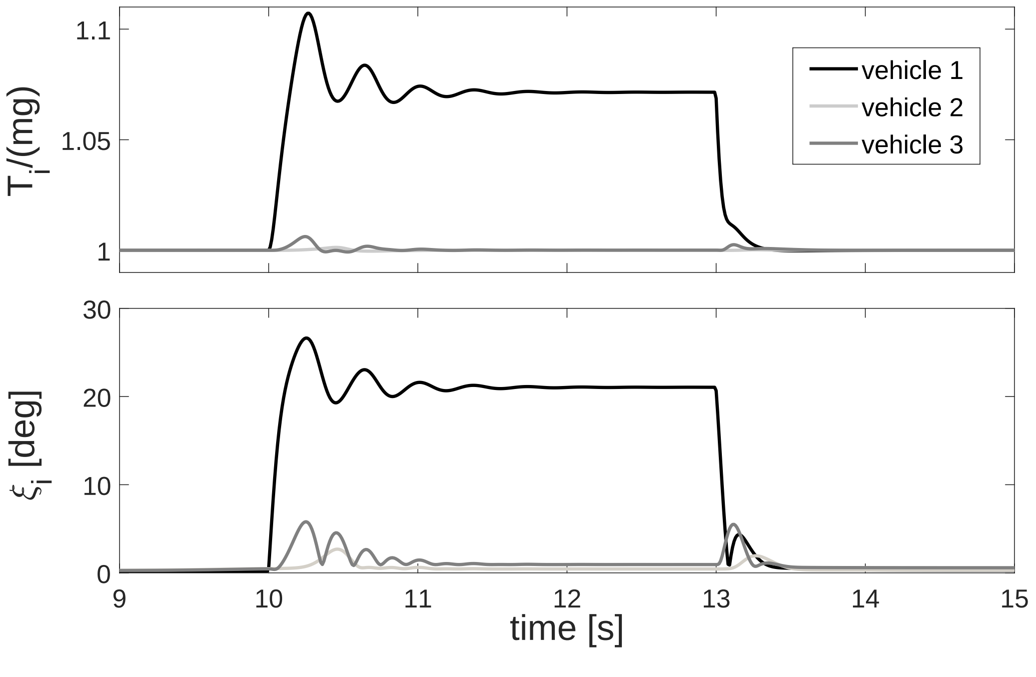


FIGURE 5.8: Thrust magnitude and inclination angle commands (Case 1).

and maintain the desired mutual distances. As expected, the highest control effort is demanded to multirotor 1, although the values of maximum thrust and inclination angle do not exceed the given constraints.

5.3.2.2 Case 2

The quadrotors leave their initial hovering positions $r_1(0) = [0\ 2\ -3.5]^T$ m, $r_2(0) = [0\ -2\ -1]^T$ m, and $r_3(0) = [0\ 0\ -3]^T$ m at time $t = 0$ s. The reference point is the same analyzed in Case 1, identifying a constant-altitude rectilinear trajectory for $t \geq 0$ s. A line formation geometry is defined by $\bar{d}_1 = [0\ -0.5\ 0]^T$ m, $\bar{d}_2 = [0\ 0.5\ 0]^T$ m, and $\bar{d}_3 = [0\ 0\ 0]^T$ m for $t \in [0, 12[$ s. At time $t = 12$ s a formation reconfiguration is commanded in order to obtain a classic triangular shape with $\bar{d}_1 = [1\ 0\ 0]^T$ m, $\bar{d}_2 = [0\ -1\ 0]^T$ m, and $\bar{d}_3 = [0\ 1\ 0]^T$ m. Collision avoidance control gain is $K_a = 0.01$ m^2 s^{-2}.

In this framework, the initial hovering positions are designed in such a way that the trajectories of the agents would intersect during the formation acquisition phase if no collision avoidance control were performed. The same situation would occur during the formation reconfiguration phase, where vehicle 1 moves to the front vertex of the triangular shape while vehicles 2 and 3 swap their respective initial positions. With respect to Case 1, altitude changes are also required for the multirotors to reach the line formation for $t \in [0, 12[$ s. Figure (5.9) shows the trajectory of vehicles leaving their

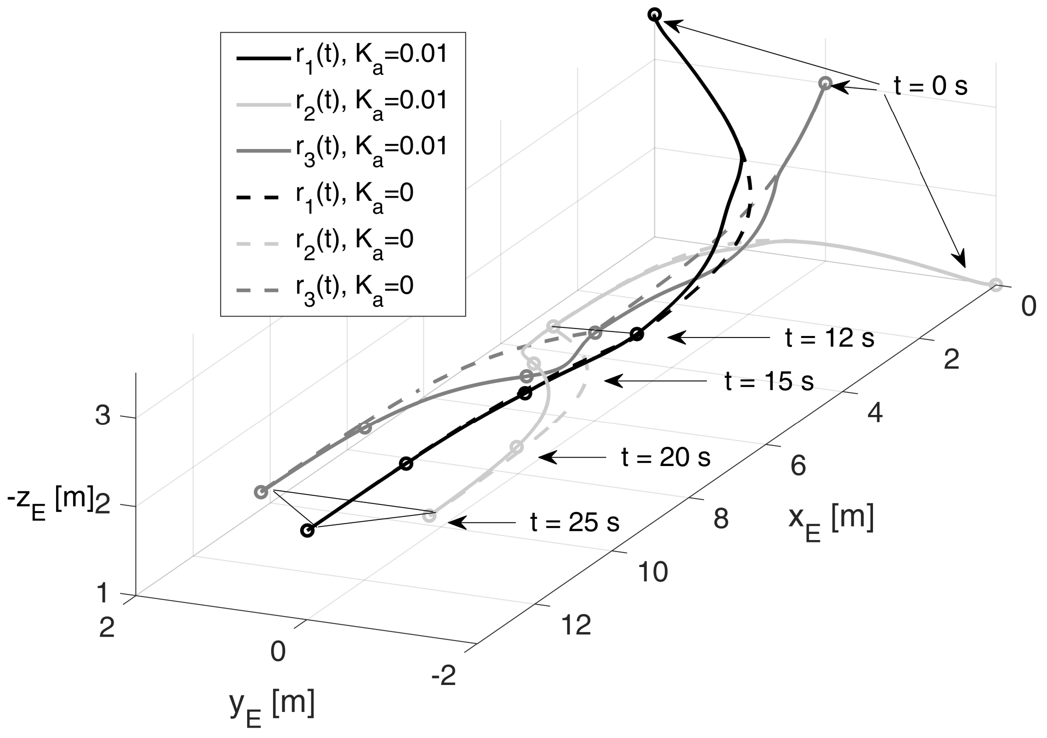


FIGURE 5.9: Formation trajectories (Case 2).

initial positions while circle markers indicate the formation configuration at discrete time instants. Relative distances $\|a_{12}\|$, $\|a_{23}\|$, and $\|a_{31}\|$ are also plotted in Figure (5.10) as a function of time. The gray zone is still lower-limited by the radius of the safety zone,

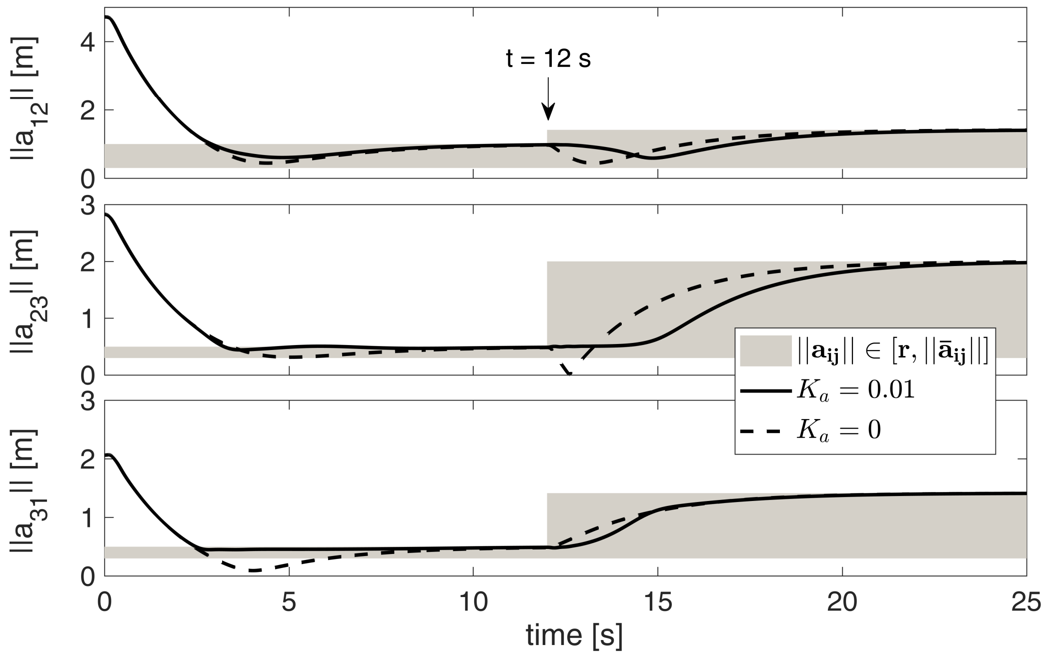


FIGURE 5.10: Mutual distances between agents of the formation (Case 2).

while the upper-limit increases at $t = 12$ s because of the formation reconfiguration. When no collision avoidance control is performed, the dashed line representing $\|a_{31}\|$ almost goes to zero during the first phase of the maneuver, thus implying collision between vehicles 1 and 3. At the same time, vehicle 2 hazardously approaches vehicles 1 and 3. After formation reconfiguration, collision occurs between agents 2 and 3, while $\|a_{12}\|$ becomes very close to the lower bound. Constraints are instead respected when $K_a \neq 0$, despite the unfavorable initial conditions and the unusual reconfiguration maneuver of the formation.

5.3.3 Flight test

In order to validate the proposed approach in a real mission scenario, an experimental campaign is performed by using a DJI® F550 hexarotor with take-off mass $m = 1.47$ kg and arm length $l = 0.46$ m. The term "hexarotor" stands for multirotor vehicle equipped with six rotors. The vehicle is equipped with the Pixhawk® PX4 autopilot, already presented in Chapter 3.2. The proposed control technique is first model-based designed and validated in Matlab/Simulink® environment. Then Matlab® scripts and the Simulink blocks related to formation control are included in the flight management software, comprehensive of estimation algorithm and aircraft attitude control, and directly coded by using the Pixhawk® Pilot Support Package toolbox. At this point the dedicated firmware can be easily deployed to the Pixhawk® unit. The autopilot unit performs estimation and control tasks in real-time with a frequency set to 250 Hz while relevant flight data are saved on a microSD card at 25 Hz. Focusing the attention on the collision prevention task, a simple maneuver is performed outdoor where $r_r(t) = [0\ 0\ \ -3]^T$ m and the hexarotor, here named vehicle 1, is required to leave its initial position $r_1(0) = [-3.9\ 6.7\ \ -2.9]^T$ m and reach the desired position defined by $\bar{d}_1 = [5\ 0\ 0]^T$ m, that is $\bar{r}_1(t) = [5\ 0\ -3]^T$ m. Constraints are set on the maximum total thrust magnitude, namely $T_1 \in [0,\ 2\,mg]$, and on the commanded inclination, $\xi_1 \in [0, 45]$ deg. It is assumed that a virtual vehicle 2 is at hover in $r_2(t) \equiv \bar{r}_2(t) = [0\ 0\ \ -3]^T$ m for all $t \geq 0$ s, emulating a condition where vehicle 2 is able to perfectly perform position keeping but its condition is not affected by the state of vehicle 1. The vehicle 1 global position, expressed in terms of latitude, longitude, and altitude, is estimated in real-time by filtering raw measurements from on-board GNSS and inertial measurement unit and then it is rotated to local position according to the North-East-Down frame. The experimental setup represents the case when vehicle 1 has the knowledge of vehicle 2 state vector (written in the firmware as constant data), but the converse does not occur anymore due to communication loss. In this framework, the formation control gain matrices are given by $K_V = 1 \cdot I_3$ s^{-1} and $K_d = 0.5 \cdot I_3$ s^{-2}, while the collision avoidance control gain is $K_a = 1$ m^2 s^{-2}.

The sample maneuver is depicted in Fig. 5.11, where the trajectory of vehicle 1 is plotted for $t \in [0, 12.9]$ s. At time $t = 0$ s, when formation control is activated, the attitude of vehicle 1 is described by Euler angles $\Phi_1(0) = -10.9$ deg, $\Theta_1(0) = -21.2$ deg, and $\Psi_1(0) = -28.7$ deg. It can be noted that $\Psi_1(0) \neq 0$ deg represents a non-nominal condition for the computation of attitude commands as described in Section 5.3. Thus, the initial part of the formation acquisition maneuver is characterized by a non-nominal trajectory during which correction of the yaw angle by the internal autopilot is still ongoing. During the non-nominal formation acquisition phase, vehicle 1 results to be driven toward the sphere with radius $r = 0.7$ m centered at the position of virtual vehicle 2, where the potential function contribution of the controller in Eq. (5.12) becomes predominant and collision avoidance occurs. Figure 5.11 also reports an intuitive representation of potential function V_p in a 3-dimensional environment together with the trajectory of vehicle 1. In particular, the values of V_p can be read along the vertical axis as a function of the horizontal position on the x_E-y_E plane. It can be finally noted that V_p increases its value near the safety sphere, ideally going to infinity on the sphere surface, while a minimum of V_p characterizes the desired position $\bar{r}_1$.

During the last part of the maneuver, vehicle 1 finally reaches the desired position and stabilizes at hover. The proposed experimental setup, here focused on collision avoidance, determines severe maneuvers for vehicle 1. It is in the intentions of the authors, however, to intentionally validate the efficiency of the proposed approach under non-nominal conditions: a) vehicle 1 is required to perform formation acquisition with large

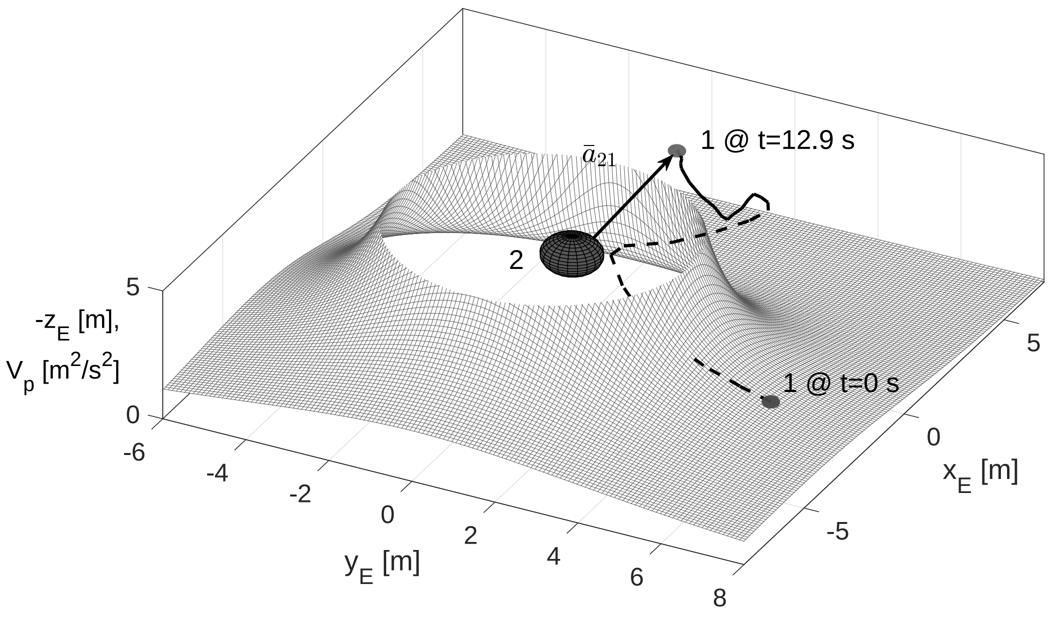


FIGURE 5.11: Trajectory of vehicle 1 during the experimental test.

initial error $\|delta_1(0)\| = \|\bar{r}_1(t) - r_1(0)\| \approx 11.1$ m, b) vehicle 1 starts the maneuver from a non-nominal yaw condition, c) vehicle 2 is not influenced by the state of vehicle 1 and does not perform its expected part of collision avoidance task (in this case, vehicle 2 rather represents a fixed obstacle for vehicle 1). This latter aspect particularly makes the closed-loop control task of vehicle 1 demanding. As a matter of fact, the experiment proves the efficiency of the proposed controller in a real application and shows encouraging performance capabilities in unusual conditions.

5.4 Optimal navigation strategy for formation flight

With respect to the strategy exposed in 5.3, a different approach is represented by optimization methods, which attempt to find an input that minimizes a performance index to avoid obstacles. Literature illustrates how most of these methods calculate the performance index for a finite time horizon, which can be easily combined with model predictive control [39–41]. This section shows how the navigation strategy described in Chapter 4 can be extended to formation of multiple UAVs. This approach, in line with the case of single agent, proposes to solve the resulting problem globally, considering all agents objectives and constraints. The proposed technique requires a single central unit for computing trajectories and communicating them to each vehicle. This approach results in a good solution if the number of agents in the formation is small, indeed it scales badly with the number of agents in terms of computation as well as communication load [42, 43]. The offline motion planning problem can be written as follows:

$$\begin{aligned} \underset{\forall i : q_i(\cdot)}{\text{minimize}} \quad & \sum_{i=1}^{N} (J_i(q_i) + \sum_{k=1}^{M_i} \mathsf{w} \int_0^T \epsilon_k \, dt) \\ \text{subject to} \quad & q_i^{(j)}(0) = q_{i,0}^{(j)} \,, \quad j \in \{0, \dots, r\} \,, \\ & q_i^{(j)}(T) = q_{i,T}^{(j)} \,, \quad j \in \{0, \dots, r\} \,, \\ & g_{ik}(q_i, q_k, \epsilon_k) \leq 0 \,, \\ & h_i(q_i, t) \geq 0 \,, \\ & \forall t \in [0, T] \,, \quad \forall i \in [0, N] \,, \quad \forall k \in [0, M_i] \,. \end{aligned} \tag{5.15}$$

The optimization problem 5.15 deals with motion planning for multi-vehicle systems and considers the problem of finding optimal input trajectories q_i for each vehicle i to steer it from an initial location $q_i(0)$ towards a desired destination $q_i(T)$, while satisfying interaction constraints $g_{ik}(q_i, q_k, \epsilon_k)$ between the agents, such as attaining a formation, avoiding collisions with each other or meeting at the destination position. In the meantime, each vehicle should respect its own kinematic and dynamic limitations and avoid collisions with the environment, these constraints are represented by $h_i(q_i, t)$. In comparison to the single agent case, an extra term is added to the objective function to trade-off time optimality against formation keeping during motion:

$$\mathsf{w} \int_0^T \epsilon_k(t) \, dt \tag{5.16}$$

in which ϵ_k is called soft-formation deviation, and w is the soft-formation parameter has to be tuned. Formation constraints are adapted as follows:

$$-\epsilon_k(t) \leq q_i(t) - q_k(t) - \Delta q_{ik} \leq \epsilon_k(t) \,, \tag{5.17}$$

This construction motivates agents to fly in fleet but it allows little variations in the formation geometry if e.g. a dynamic obstacle has to be avoided. In order to use the approach described in Chapter 4, new variables ϵ_k are parameterized as extra splines such that constraints can be reformulated as polynomial constraints in q, its derivatives and anti-derivatives. Also this strategy was tested with simulations and implemented as part of the software toolbox OMG-tools.

Figure 5.13 shows how different values of the soft-formation parameter w can affects the average solving time for trajectories. According to simulations, high values of this parameter can reduce the problem complexity.

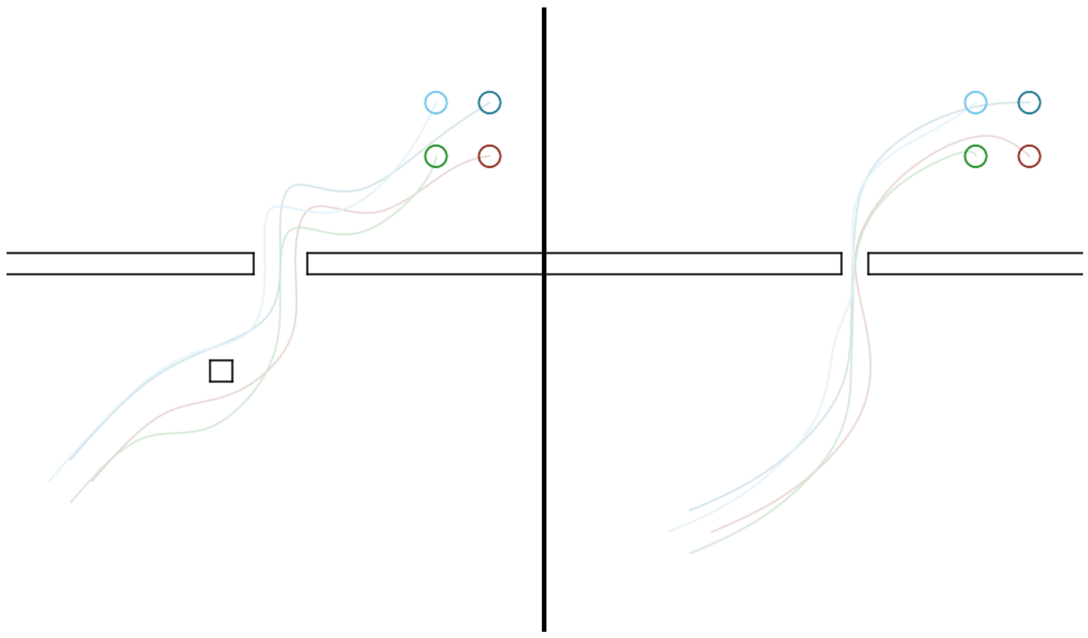
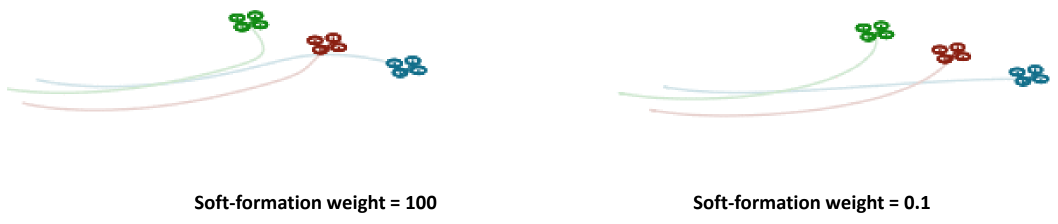


FIGURE 5.12: Examples of optimal motion planning problems solved for a formation of four multirotor vehicles.

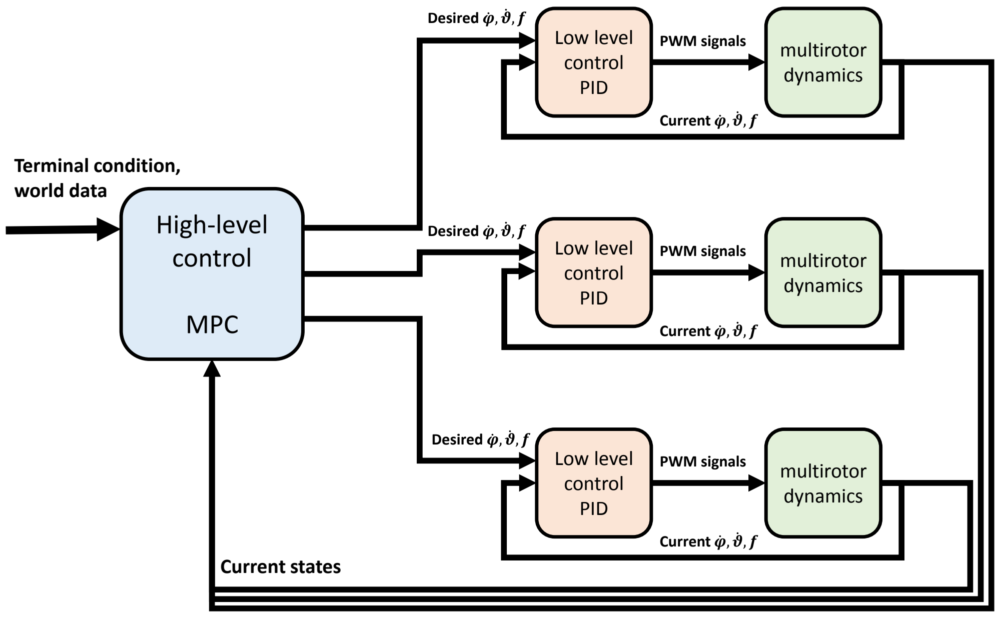


Soft-formation weight	Average update time
100	569.465 ms
10	861.049 ms
1	1269.33 ms
0.1	1281.14 ms

FIGURE 5.13: Examples of optimal motion planning problems solved for a formation of four holonomic vehicles.

5.4.1 Model predictive control strategy for formation flight

The multi-agent motion problem can be solved in a receding horizon by running it into a model predictive control architecture. Following Section 4.5 as outline, the motion-planning problem is coupled with multirotor model derived in Section 2. That model will be used in simulations to perform the calculated trajectories. A low-level control system, required to lead vehicle models along the trajectories, has already explained in section 4.5.1. Figure 5.14 shows the architecture used for solving online the navigation problem for the multi-agent case. For a real implementation, it has to be said that



Terminal condition, world data
High-level control
MPC
Desired $\dot{\varphi}, \dot{\vartheta}, f$
Low level control PID
PWM signals
multirotor dynamics
Current $\dot{\varphi}, \dot{\vartheta}, f$
Desired $\dot{\varphi}, \dot{\vartheta}, f$
Low level control PID
PWM signals
multirotor dynamics
Current $\dot{\varphi}, \dot{\vartheta}, f$
Desired $\dot{\varphi}, \dot{\vartheta}, f$
Low level control PID
PWM signals
multirotor dynamics
Current $\dot{\varphi}, \dot{\vartheta}, f$
Current states

FIGURE 5.14: Model predictive control scheme.

communication among vehicles, provided of necessity by wireless devices, could require more time in comparison with the single-agent case. Since the motion problem is going to be solved globally, the MPC navigation algorithm is very similar to the one already proposed in Chapter 4. The algorithm uses the update time $\Delta T : k = 0, 1, 2, ..$ and can be described by the following steps:

1. At time t_k, execute trajectories $q_{k\,i}$.
2. At time $= t \in [t_k, t_{k+1}]$ update each vehicle's current state $x_{k\,i}(t)$ from measurements and world information.
3. Estimate $\hat{x}_{k\,i}(t_{k+1})$.
4. Solve optimization problem globally using $\hat{x}_{k\,i}(t_{k+1})$ as initial data of each vehicle dynamics and environment constraints. Obtain complete motion trajectories $q_{k+1\,i}$ for a granted time horizon.
5. At time $t_{k+1} = t_k + \Delta T$, each vehicle's attitude and thrust controllers receive trajectories $q_{k+1\,i}$.
6. Repeat.

5.4.2 Numerical simulation

Figure 5.15 illustrates an example of the strategy explained above. A fleet of quadrotor vehicles have to reach the imposed formation center's destination point at $[-3, 4, 1]$. Initial locations of all agents are spread in the airspace at $[0, 2, 1]$, $[2, 2, 1]$ and $[4, 2, 1]$, without fulfilling the desired formation geometry. As already said, the spline-based motion planning problem is solved using the Optimal Motion Generation-tools software [23], a user-friendly toolbox written in Python that uses the symbolic framework CasADi to perform nonlinear numerical optimization. The motion planner parameterizes trajectories as cubic splines with 10 polynomial intervals, the control horizon T is set to $10\,s$. Environmental constraints are added in advance during the primary navigation problem construction. In order to obtain smooth and feasible trajectories the following model constraints are imposed for each vehicle, which are considered identical:

$$\begin{aligned} &2\,m/s^2 \leq f \leq 15\,m/s^2\,, \\ &-15\,^\circ \leq \phi \leq 15\,^\circ\,, \qquad -15\,^\circ \leq \theta \leq 15\,^\circ\,, \\ &-5\,^\circ/s \leq \dot{\phi} \leq 5\,^\circ/s\,, \qquad -5\,^\circ/s \leq \dot{\theta} \leq 5\,^\circ/s\,. \end{aligned} \tag{5.18}$$

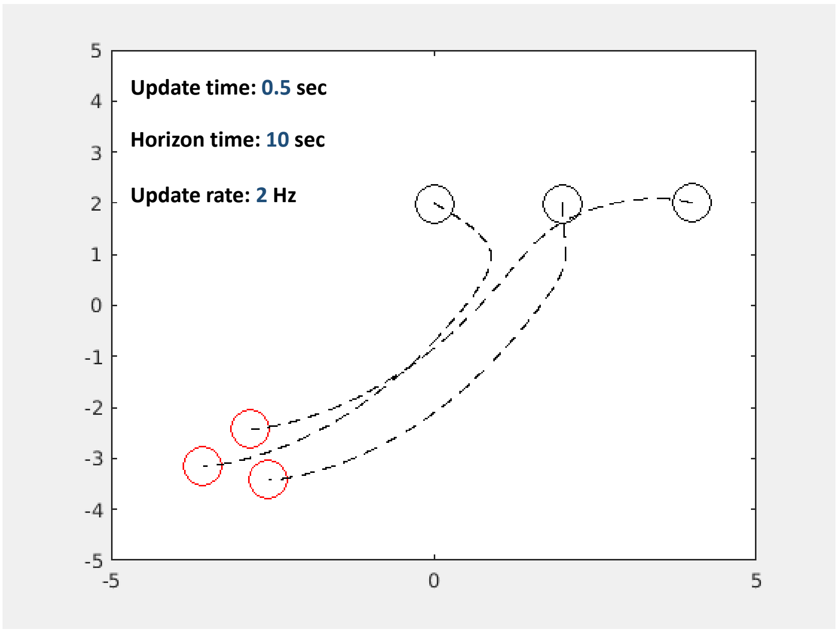


FIGURE 5.15: MPC Simulations for a fleet of multirotors (Two-dimensional view).

Chapter 6

Conclusions

The purpose of this thesis was to describe the architecture behind a flight control software for multirotor UAVs and to demonstrate how the usage of Model-based approach in UAV flight control design has the potentiality to shorten the design cycles and reduce the development cost. Through various control problems addressed and solved on the different chapters, this thesis proves how the application of the Model-based design in flight control enables fast transitions from development to simulation and experimental stages. Furthermore, the advantages of that approach become more tangible for realities characterized by restricted working group and budget. In Matlab/Simulink® a software model of the general control architecture of a multirotor UAV has been built, addressing various flight control tasks, including trajectory generation, attitude and navigation control. Problems like trajectory planning, collision avoidance, formation flight control, could be tackled and implemented into the model by using different strategies. Embedded codes have automatically been generated from Simulink® models, so that each strategy presented in the thesis could be validated by simulations and experimental tests. In conclusion, the model-based design approach has been proved to be practical and effective during software development. The model-based environment guarantees great operational flexibility during the whole development process: design, analysis, simulation, automatic code generation and verification. Trying different control strategies becomes easy, without the need of building prototypes. Testing and validation can be done several times throughout the design process rather than at the end of it, so that many errors can be found and corrected before hardware testing. Automatic embedded code generation from system model reduces effort and eliminates hand-coding errors, in general it results to be more efficient and useful for testing in real-time simulations. Finally, model-based environments can be adapted and re-used on subsequent projects, the model lends itself to team-work, resulting particularly suitable for small engineering teams.

Appendix A

Control design considering drag

Section 3.4.4.4 shows how to design a PID strategy for controlling a multirotor vehicle by desired velocity commands and how required relations between desired accelerations and desired attitude angles can be found. This is obtained under the assumption of zero speed, which means zero drag forces acting on vehicle. This Appendix explains how it is conceptually possible, but improper for a number of reasons, computing desired roll angle ϕ_{des} and pitch angle θ_{des}, starting from desired accelerations $a_x|des$ and $a_y|des$, also considering drag forces.

We start rotating the velocity vector from the body fixed frame to the vehicle frame:

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \end{bmatrix} = R_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} c\theta & s\phi s\theta & c\phi s\theta \\ 0 & c\phi & -s\phi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A.1}$$

Differentiating equation (28) and neglecting $\dot{R_b^v}$ gives:

$$\begin{bmatrix} \ddot{p_x} \\ \ddot{p_y} \\ \ddot{p_z} \end{bmatrix} = \begin{bmatrix} c\theta & s\phi s\theta & c\phi s\theta \\ 0 & c\phi & -s\phi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} \tag{A.2}$$

Now we can plug equation (25) into equation (29), neglecting the Coriolis terms we obtain:

$$\begin{bmatrix} \ddot{p_x} \\ \ddot{p_y} \\ \ddot{p_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} -c\phi s\theta \\ s\phi \\ -c\phi c\theta \end{bmatrix} \frac{F_{tot}}{m} - \frac{1}{2m} C_D \rho \begin{bmatrix} A_x c\theta & A_y s\phi s\theta & A_z c\phi s\theta \\ 0 & A_y c\phi & -A_z s\phi \\ -A_x s\theta & A_y s\phi c\theta & A_z c\phi c\theta \end{bmatrix} \begin{bmatrix} u|u| \\ v|v| \\ w|w| \end{bmatrix} \tag{A.3}$$

By setting

$$\ddot{p_x} = a_x \tag{A.4}$$

$$\ddot{p_y} = a_y \tag{A.5}$$

$$\ddot{p_z} = a_z \tag{A.6}$$

From equations (30) we can find the relationship between acceleration and attitude

$$a_x = -\frac{F_{tot}}{m}\cos\phi\sin\theta - \frac{R_1}{m}\cos\theta - \frac{R_2}{m}\sin\phi\sin\theta - \frac{R_3}{m}\cos\phi\sin\theta \tag{A.7}$$

$$a_y = \frac{F_{tot}}{m}\sin\phi - \frac{R_2}{m}\cos\phi + \frac{R_3}{m}\sin\phi \tag{A.8}$$

$$a_z = g - \frac{F_{tot}}{m}\cos\phi\cos\theta + \frac{R_1}{m}\sin\phi - \frac{R_2}{m}\sin\phi\cos\theta - \frac{R_3}{m}\cos\phi\cos\theta \tag{A.9}$$

Where we have set:

$$R_1 = \frac{1}{2}\rho C_D A_x u|u| \tag{A.10}$$

$$R_2 = \frac{1}{2}\rho C_D A_y v|v| \tag{A.11}$$

$$R_3 = \frac{1}{2}\rho C_D A_z w|w| \tag{A.12}$$

We have to solve equation (37) for $\frac{F_{tot}}{m}$ and replace it into eq. (35) and (36), we obtain:

$$a_x = (a_z - g)\tan\theta - \frac{R_1}{m}\frac{1}{\cos\theta} \tag{A.13}$$

$$a_y = (\frac{a_z - g}{\cos\theta} + \frac{R_1}{m}\tan\theta)\tan\phi - \frac{R_2}{m}\frac{1}{\cos\phi} \tag{A.14}$$

To solve the equations we can use half angle formulas reported below for a generic angle α:

$$\sin\alpha = \frac{2t}{1+t^2} \tag{A.15}$$

$$\cos\alpha = \frac{1-t^2}{1+t^2} \tag{A.16}$$

$$\tan\alpha = \frac{2t}{1-t^2} \tag{A.17}$$

with $t = \tan\frac{\alpha}{2}$, $\alpha \neq \pi + 2k\pi$ We have to solve the following equations for $t_1 = \tan\frac{\theta}{2}$ and $t_2 = \tan\frac{\phi}{2}$:

$$(R_1 - ma_x)t_1^2 - 2m(a_z - g)t_1 + R_1 + ma_x = 0 \tag{A.18}$$

$$(R_2 - ma_y)t_2^2 - 2(\frac{m(g - a_z)}{\cos\theta} + R_1\tan\theta)t_2 + R_2 + ma_y = 0 \tag{A.19}$$

The solutions for ϕ and θ are given by:

$$\theta_{des} = 2\arctan\left(\frac{a_z - g \pm \sqrt{(g-a_z)^2 - (\frac{R_1}{m} - a_{x|des})^2}}{\frac{R_1}{m} - a_{x|des}}\right) \tag{A.20}$$

$$\phi_{des} = 2\arctan\left(\frac{(\frac{(g-a_z)}{\cos\theta_{des}} + \frac{R_1}{m}\tan\theta_{des}) \pm \sqrt{(\frac{g-a_z}{\cos\theta_{des}} + \frac{R_1}{m}\tan\theta_{des})^2 - (\frac{R_2}{m} - a_{y|des})^2}}{\frac{R_2}{m} - a_{y|des}}\right) \tag{A.21}$$

Relations above provide more accurate solutions for angles and they can theoretically be used as references for the inner-loop covering attitude control. However, these solutions also present several disadvantages which exclude in practice their usage for control design. Disadvantages can be summarized as follows:
1 - It is not easy to quantify properly parameters $A_{x,y,z}$
2 - For control design purposes, including drag into solutions for angles results in a vehicle attitude that varies continuously with the vehicle flying speed. This gives uncomfortable piloting sensations to most of users.
3 - In the end, equations (A.20) and (A.21) result quite complex without presenting recognizable benefits in multirotor control design.

Bibliography

[1] Andrew M. St. Laurent. Understanding open source and free software licensing. pages 34–84, August . URL http://www.oreilly.com/openbook/osfreesoft/book/.

[2] K.P. Valavanis. Advances in unmanned aerial vehicles: State of the art and the road to autonomy. *Micro-processor-Based and Intelligent Systems Engineering Series*, pages 15–46, July 2007.

[3] Giulietti F. Avanzini, G. Maximum range for battery-powered aircraft. *Journal of Aircraft*, 50(1):304–307, 2013.

[4] de Angelis E.L. Avanzini, G. and F. Giulietti. Optimal performance and sizing of a battery-powered aircraft. *Aerospace Science and Technology*, 59:132–144, December 2016.

[5] Giulietti F. Gatti, M. and M. Turci. Maximum endurance for battery-powered rotary-wing aircraft. *Aerospace Science and Technology*, 45:174–179, September 2015.

[6] R. Beard. Quadrotor dynamics and control rev 0.1. 2008.

[7] Mohammed Rizwanullah N Md Jubair basha, Salman Abdul Moiz. Model based software develeopment: Issues challenges. *International Journal of Computer Science Informatics*, 2:2231–5292, 2012.

[8] URL https://pixhawk.org/.

[9] URL https://www.mathworks.com/hardware-support/pixhawk.html.

[10] Ferrarese G. Giulietti F. Modenini D. de Angelis, E.L. and P. Tortora. Gaussian deterministic recursive estimator with online tuning capabilities. *Journal of Guidance, Control, and Dynamics*, 38(9):1827–1833, September 2015.

[11] Ferrarese G. Giulietti F. Modenini D. de Angelis, E.L. and P. Tortora. Terminal height estimation using a fading gaussian deterministic filter. *Aerospace Science and Technology*, 55:366–376, August 2016.

[12] . URL https://github.com/priseborough/InertialNav.

[13] . URL https://github.com/priseborough/InertialNav/blob/master/derivations/GenerateEquations22states.m.

[14] R. Czyba G. Szafranski. Different approaches of pid control uav type quadrotor. *International Micro Air Vehicles conference 2011*, 2011.

[15] Pipeleers G. Swevers J. Van Loock, W. B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction. *Mechanical Sciences*, 6(2):163–171, 2015.

[16] Pipeleers G. Van Parys, R. Spline-based motion planning in an obstructed 3d environment. *20th IFAC World Congress*, July 2017.

[17] URL https://www.mech.kuleuven.be/en/pma/research/meco.

[18] C. De Boor. A practical guide to splines, revised edition, vol. 27 of applied mathematical sciences. *Mechanical Sciences*, 27, August 2001.

[19] Lvine J. Martin P. Fliess, M. and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *Int. J. Control*, 61:1327–1361, 1995.

[20] W. Van Loock T. Mercy and G. Pipeleers. Real-time motion planning in the presence of moving obstacles. *2016 European Control Conference*, pages 1586–1591, July 2016.

[21] Vandenberghe L. Boyd, S. Convex optimization. *Cambridge university press*, 2004.

[22] H. Asada M. West and C. Zhu. Design of holonomic omnidirectional vehicle. *International Conference of Robotics and Automation*, May 1992.

[23] Mercy T. OMG-tools Van Parys, R. Omg-tools. 2016. URL https://github.com/mecogroup/.

[24] Zheng A. Some practical issues and possible solutions for nonlinear model predictive control. in: Allgwer f., zheng a. (eds) nonlinear model predictive control. progress in systems and control theory. *Aerospace Science and Technology*, 26:129–143, 2000.

[25] G. Rossetti F. Giulietti, G. Pipeleers and R. Van Parys. *RED UAS 2017 International Conference*, October .

[26] URL http://qgroundcontrol.org/mavlink/start.

[27] Kendoul F. Suzuki S.-Wang W. Nonami, K. and D. Nakazawa. Autonomous flying robots: Unmanned aerial vehicles and micro aerial vehicles. pages 22–24, 2010.

[28] T. Balch and R. Arkin. Behavior-based formation control for multi-robot systems. *IEEE Trans. Robot. Autom.*, 14(6):926–939, December 1998.

[29] Menhaj M.B. Asl, A.N. and A. Sajedin. Control of leader-follower formation and path planning of mobile robots using asexual reproduction optimization (aro),. *Appl. Soft Comput.*, 14:563–576, January 2014.

[30] Abdollahi F. Rahimi, R. and K. Naqshi. Time-varying formation control of a collaborative heterogeneous multi agent system. *Robot. Auton. Syst.*, 62:1799–1805, December 2014.

[31] Abdollahi F. Aghaeeyan, A. and H.A. Talebi. Uav-ugvs cooperation: with a moving center based trajectory. *Robot. Auton. Syst.*, 63:1–9, January 2015.

[32] Wen G. Rahmani A. Peng, Z. and Y. Yu. Leader-follower formation control of nonholonomic mobile robots based on a bioinspired neurodynamic based approach. *Robot. Auton. Syst.*, 61:988–996, September 2013.

[33] D. Panagou and V. Kumar. Cooperative visibility maintenance for leader-follower formations in obstacle environments. *IEEE Trans. Robot.*, 30:831–844, March 2014.

[34] F. Giulietti and G. Mengali. Dynamics and control of different aircraft formation structures. *The Aeronautical Journal*, 108(1081):117–124, March 2004.

[35] Innocenti M. Napolitano M. Giulietti, F. and L. Pollini. Dynamic and control issues of formation flight. *Aerospace Science and Technology*, 9(1):65–71, January 2005.

[36] Giulietti F. de Angelis, E.L. and G. Rossetti. Multirotor aircraft formation flight control with collision avoidance capability. *Aerospace Science and Technology - in press*, 2017.

[37] Acampora-Prado I.A. dos Santos D.A. Viana, I.B. and L.C. Sandoval-Goes. Formation flight control of multirotor helicopters with collision avoidance. *International Conference on Unmanned Aircraft Systems*, pages 757–764, June 2015.

[38] Damaren C.J. de Ruiter, A.H.J. and J.R. Forbes. Spacecraft dynamics and control - an introduction. pages 19–24, 2013.

[39] R. Van Parys and G. Pipeleers. Online distributed motion planning for multi-vehicle systems. *2016 European Control Conference*, pages 1580–1585, July 2016.

[40] W.B. Dunbar and R.M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, pages 549–558, April 2006.

[41] Borrelli F. Fregene K. Godbole D. Kevickzy, T. and G.J. Balas. Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on Control System and Technology*, 16(1):19–33, December 2007.

[42] D. Sun S. Liu and C. Zhu. Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. *Mechatronics, IEEE/ASME Transactions on*, 16(6):1021–1031, 2011.

[43] A. Kushleyev D. Mellinger and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 477–483, 2012.