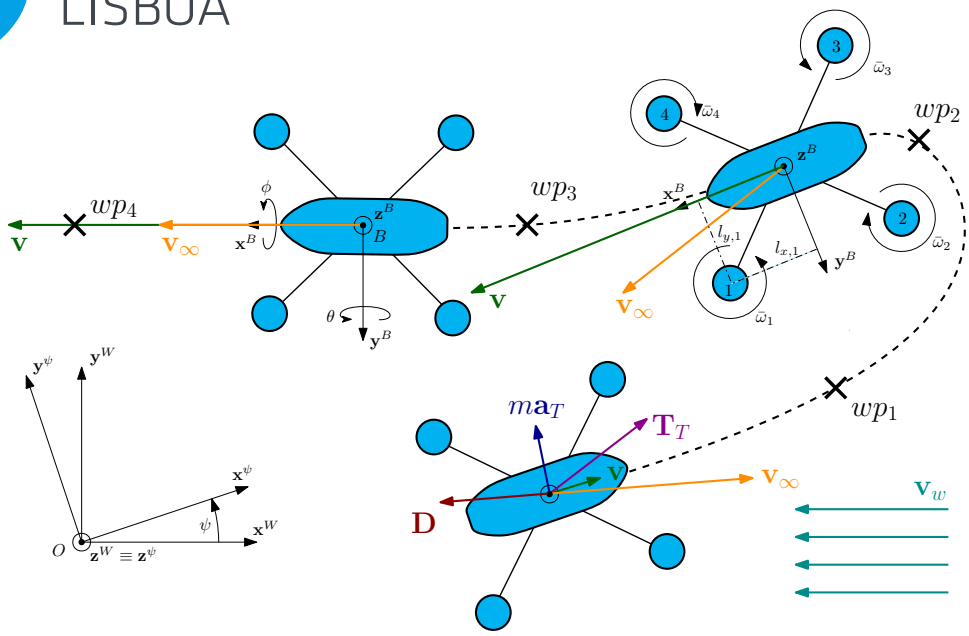


Quadrotor Thrust Vectoring Control with Time Optimal Trajectory Planning in Constant Wind Fields

João Paulo da Rocha Silva

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Guido Cornelis Henricus Eugène de Croon

Prof. João Manuel Lage de Miranda Lemos

Examination Committee

Chairperson: Prof. José Fernando Alves da Silva

Supervisor: Prof. João Manuel Lage de Miranda Lemos

Member of the Committee: Prof. Rita Maria Mendes de Almeida Correia da Cunha

March 2017

Acknowledgments

The academic path is a long and hard journey. It starts when we are little, young and naive, it starts when we are born. Twenty two years later, it is now the time to finish what has long been started, and as always, the goal is the hardest chapter. Throughout these years I have learned incredible theories which I hope to apply when needed. For us Engineers (or soon to be) that knowledge is amazing, and it changes our perception of life. Nevertheless, knowledge is not everything, and most important matters exist. Since the beginning I have lived incredible moments, some of sadness of course, but most of happiness. I believe that every moment is only truly ours if shared with someone, and that is why I want to acknowledge the people that made all this journey feasible, not focusing only on the dissertation period.

It all begins with family. Family is always there, always protects you, and always supports you, unconditionally. Therefore, I want give a heartfelt thank you to my resolute father, to my sweat mother and to my marvelous sisters. And evidently to the other members of my family, my godparents, my grandparents and to my cousins that started this long path before me.

They say you cannot chose family, but you can chose your friends. Mine are the best, at least in my opinion. So, I want to thank my friends from my old town Famalicão, in particular Fil and Tiago, who are genius in their own distinct ways. From Famalicão I moved to Lisbon to start with the Aerospace Engineering studies and Eduardo joined that journey with me. I want to thank him for that, because of him I have never felt alone in the big city. In Lisbon I have made incredible friends that would always help me and for that I want to thank them, in particular Filipa, for being the nicest person I have ever met, Marta, for being my sister from another home and Sónia, my school godmother and life adviser. I moved to Delft and the same process went on. This time Miguel joined the adventure and once again I have never felt alone, thank you for all the support. Delft would not be Delft without the *"Tugas em Delft"*, whom I want to thank for the dinners and celebrations we had here. They were a strong support during this last year.

I also want to thank every teacher and supervisor I had, they are the main reason why I am here. In particular Christophe, Lodewijk and Guido who carefully guided me in Delft, and professor João Miranda Lemos who always helped me, regardless of the physical distance.

There is family, friends, teachers, and then there is the girl. She has been my dearest ally since the moment I have met her, and I sincerely believe that I could not have done this without her always by my side. My Marina, thank you so much for taking care of my heart.

Thank you, I hope I can make you all proud!

João Paulo, 8 March 2017

Resumo

Neste trabalho é proposta uma estratégia de controlo que permite seguir trajectórias tempo-óptimas que visitam um conjunto de pontos de passagem em condições ventosas. São investigados os efeitos aerodinâmicos dos quadrirotores, com ênfase na resistência de *blade flapping*, resistência induzida e resistência parasítica. É desenvolvido um método completo para a identificação dos coeficientes aerodinâmicos, e é analisada a sua influência no desempenho do quadrirotor. É sugerida uma abordagem computacionalmente eficiente dividida em três passos para optimizar a trajectória, minimizando a resistência aerodinâmica e o *jerk*, garantindo resultados perto do óptimo. É feita uma comparação entre as trajectórias suave obtidas e trajectórias *standard,* que consistem em trajectórias discretas ponto a ponto seguidas de filtragem de passa-baixo, mostrando melhorias energéticas e reduções a nível da agressividade dos ângulos de Euler. É concebido um controlador de impulso vectorial, que explora os efeitos aerodinâmicos não lineares e usa informação *a priori* sobre a trajectória, e é de seguida comparado com controladores PID, mostrando uma melhoria no desempenho devido à redução do atraso temporal no seguimento da referência e ao aumento do envelope de vôo.

Palavras-Chave: Quadrirotor, Vento, Trajectória Óptima, Controlo, Impulso Vectorial, Efeitos de Resistência

Abstract

This work proposes a control strategy to follow time optimal trajectories planned to visit a given set of waypoints in windy conditions. The aerodynamic effects of quadrotors are investigated, with emphasis on blade flapping, induced and parasitic drag. An extended method to identify all the aerodynamic coefficients is developed, and their influence on the performance is analyzed. A computationally efficient three steps approach is suggested to optimize the trajectory, by minimizing aerodynamic drag and jerk while still guaranteeing near optimal results. The derived smooth trajectory is compared with standard discrete point to point followed by low-pass filtering trajectories, showing energetic improvements in thrust and reductions in Euler angles aggressiveness. By exploiting the non-linear aerodynamic effects and using *a priori* trajectory information, a thrust vectoring controller is designed and compared with a PID controller, showing an increase in performance by reducing the tracking delay and extending the flight envelope.

Keywords: Quadrotor, Wind, Optimal Trajectory, Control, Thrust Vectoring, Drag Effects

Acronyms

CPU	Central Processing Unit
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
ILP	Integers Linear Programming
INDI	Incremental Non-Linear Dynamic Inversion
LS	Least Squares
MILP	Multi Integers Linear Programming
MPS	Mathematical Programming System
NLP	Non-Linear Programming
PID	Proportional Integral Derivative
PMP	Pontryagin's Minimum Principle
QP	Quadratic Programming
SCP	Sequential Convex Programming
TCP	Transmission Control Protocol
TSP	Traveling Salesman Problem
TUDelft	Delft University of Technology
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
VAF	Variance Accounted For

List of Symbols

Greek Symbols

α	Wind velocity angle
β	Blade Flapping angle
γ	Angle of attack
δ	Equivalent velocity angle
Δ	Interval
ζ	Filter damping ratio
θ	Pitch Euler angle
λ	Co-state / Motor rotation direction
μ	Advancing ratio
π	Air velocity angle
σ	General trajectory time polynomial
ϕ	Roll Euler angle
χ	Ground velocity angle
ψ	Yaw Euler angle
ω_n	Filter natural/cut-off frequency

$\bar{\omega}$	Motor angular velocity
$\boldsymbol{\omega}$	Angular velocity in body frame

Roman Symbols

a_{bla}	Lumped drag coefficient
$\mathbf{A}, \mathbf{A}', \mathbf{A}''$	Drag constant matrices
$\mathbf{A}_{eq}, \mathbf{A}_{in}$	Cost function constrains matrices
$\mathbf{b}_{eq}, \mathbf{b}_{in}$	Cost function constrains vectors
$\mathbf{c}, J, \mathbf{H}, \mathbf{f}$	Cost function optimization vector, cost, quadratic matrix and linear vector
d	Specific drag force scalar
D	Drag force scalar
$\mathbf{D}$	Drag force vector
$\mathbf{e}_p, \mathbf{e}_p, \mathbf{e}_p$	Position, velocity and acceleration vector errors
$\mathbf{F}$	Force vector
g	Acceleration of gravity
h_1, h_2	Heuristic functions
H	Hamiltonian
$\mathbf{I}$	Identity matrix
J	Least squares cost
$\mathbf{J}$	Inertia matrix
k, n, p	Quadratic Programming problem specifications
k_{h_1}, k_u	Heuristic order constants
k_{ind}	Induced drag coefficient
k_{par}	Parasitic drag coefficient
k_T	Motor thrust coefficient
k_τ	Motor torque coefficient
K_a	Acceleration gain
K_{heu}	Heuristic constant

K_i	Constant to turn the integral dimensionless
K_p, K_{pi}	Position gains
K_v	Velocity gain
$\mathbf{l}$	Motor distance vector
m	Number of waypoints / Mass of the quadrotor
p, q, r	Angular velocity in body frame components
r	Radius of the motor blade
r, v, a, j	Position, velocity, acceleration and jerk scalars
$\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}$	Position and derivatives vectors
$\mathbf{R}$	Rotation matrix
t	Time
T	Thrust scalar / Trajectory time
$\mathcal{T}$	Torque vector
$\mathbf{T}$	Thrust vector
$\mathbf{v}$	Velocity vector
V	Velocity constant scalar
$\mathbf{wp}$	Waypoints position vector

Subscripts

$aero$	Aerodynamic
ave	Average
bla	Blade propeller
$flap$	Blade flapping
h	Hovering
ind	Induced
min	Minimum
T	Target / Thrust
w	Wind

$\times$	Skew-symmetric matrix associated with the cross-product
∞	With respect to the airstream

Superscripts

ψ	Intermediate ψ reference frame
B	Body reference frame
P	Projected on the rotor plane
W	World reference frame
$\parallel$	Parallel
eq	Equivalent
$\perp$	Perpendicular
$\top$	Transpose

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1-1 Motivation

Quadrotors are a popular type of Multicopter Unmanned Aerial Vehicles (UAVs) used for applications in which fast and aggressive trajectories on a three dimensional space are required (Hehn & D'Andrea, 2015). Application scenarios such as surveillance, package delivery or plant monitoring reflect the competence that quadrotors possess to follow predefined trajectories (Hoffmann, Huang, Waslander, & Tomlin, 2007). However, due to a variety of limitations such as maximum thrust, reduced energetic capacity or bounded bank angles, their performance is not efficient, wich even worsens when the wind is present. Therefore, planning the trajectory and designing the controller to include *a priori* information about the wind in order to increase the quadrotors performance becomes a natural solution (J.A. Guerrero, 2013).

1-2 Problem Statement

The goal of this work is to plan the time optimal trajectory for quadrotors in the presence of constant wind fields. The trajectory is formulated such that m predefined desirable waypoints, defined with three spatial components, are visited. There are no restrictions in the visiting sequence, as long as all waypoints are visited. The total trajectory time has to be minimized to guarantee time optimality while still maintaining feasibility concerning the quadrotors limitations. Moreover, the trajectory needs to be promptly computed to serve close to real-time applications.

The winds influence on the quadrotor dynamics forces the inclusion of aerodynamic effects in the trajectory generation phase and to understand those effects a literature study is required. In order to control the quadrotor, a state of the art controller needs to be designed, which should minimize the aerodynamic effects and follow the aggressive trajectory efficiently, helping it to overstep the referred limitations.

Being so, the problem can be stated in three different sub-problems or topics. These topics are mainly independent and are based on self-contained theories, but will be here connected in order to get to the goal of this work above referred. They are covered in the next three Chapters of this work, and are related with a study about wind and aerodynamic effects, Chapter 2, with solving the time optimal trajectory, Chapter 3, and with designing the controller, Chapter 4.

1-3 Related Work

Detailed studies on aerodynamic effects are already available in the literature for helicopters (Prouty, 2002; Leishman, 2006), which can be seen as a distant parent of quadrotors. However, for quadrotors the literature is more scarce. The usually approaches mention separated effects which are relevant for their individual work, such as Hoffmann et al. (2007); Mahony, Kumar, and Corke (2012). The most important effects are the blade flapping, induced and parasitic drag, although there are others such as total thrust variation or ground effect. The blade flapping and induced drag are sometimes lumped together as one single drag effect, for instance in the work of Allibert, Abeywardena, Bangura, and Mahony (2014); Waslander and Wang (2009). A detailed study of all effects was recently done by Bangura and Mahony (2012) and will be the basis of this work on this topic. Since this topic is the most theoretical of the three, a more detailed review of related work will be further addressed within the respective Chapter.

Trajectory generation for quadrotors has been studied in several approaches. Smooth point to point trajectories have been studied with Non Linear Programming (NLP) (Lai, Yang, & Wu, 2006) and using the Pontryagin's Minimum Principle (PMP) (Mueller, Hehn, & D'Andrea, 2013; Hehn & D'Andrea, 2015; Mueller, Hehn, & D'Andrea, 2015). For multiple waypoints the usual approaches are Sequential Convex Programming (SCP) (Augugliaro, Schoellig, & D'Andrea, 2012) or Quadratic Programming (QP) (D. W. Mellinger, 2012; Richter, Bry, & Roy, 2013), extended to Multi Integers Linear Programming (MILP) by D. Mellinger, Kushleyev, and Kumar (2012). Those approaches, however, aim at minimizing accelerations, jerks or snaps, disregarding wind and assuming fixed traveling times. J.A. Guerrero (2013) is among the few who include wind in the trajectory formulation, but plans only point to point trajectories. Bipin, Duggal, and Krishna (2014) achieve the goal of minimizing the time for multiple waypoints, but fail to include the wind.

Regarding quadrotor control, the typical controllers available are based on classic PID theory (Hoffmann et al., 2007; Waslander & Wang, 2009; Martin & Salaun, 2010). In these approaches the wind is disregarded and treated as a disturbance to be further rejected by the controller. More complex approaches that include aerodynamic effects, such as Feedback Linearization (Sydney, Smyth, & Paley, 2013) or Integral-Backstepping (Araar & Aouf, 2014) exist, but are only implemented in simulations. In a further step, D. Mellinger and Kumar (2011) consider the trajectory generation in the controller design, using a thrust vectoring approach, but neglect wind. The same approach is used by Omari, Hua, Ducard, and Hamel (2013) but without the trajectory information. Extending both the work of D. Mellinger and Kumar (2011) and Omari et al. (2013) to incorporate both the wind and the trajectory information seems to be an interesting approach that fulfills the controller requisites.

1-4 Proposed Approach and Contributions

A general method to achieve the proposed goal is thus absent in literature. Therefore, a new approach is proposed, that consists in three sequentially linked aspects. The initial aspect of the approach is to address an aerodynamic study in order to understand the three most important aerodynamic effects that influence the quadrotors dynamics: blade flapping drag, induced drag and parasitic drag. To complement the study, a contribution is made with an extended method to identify the drag terms and an experiment is performed to validate their influence on the controller performance. The next aspect of the approach is the definition and construction of the time optimal trajectory. The trajectory is defined to be as fast as possible while maintaining the velocity with respect to the wind below a certain limit such that the wind effects are diminished the most. A method to plan the trajectory is proposed, that is divided in three steps, so that the process is structurally fast, therefore saving computational time while still guaranteeing near optimal results. The first step is to determine the time optimal sequence of waypoints using a heuristic search algorithm. The second step is to determine the optimal trajectory throughout the waypoint sequence by solving a QP optimization problem that minimizes the jerk. The third step is to ensure that the velocity respects the mentioned limit. The last aspect of the approach is to design a controller optimized to explore the *a priori* information obtained about the wind and the trajectory. A cascade thrust vectoring controller is proposed and compared with a typical PID controller, showing an increase in performance, in particular by reducing tracking delay.

In a summary, the proposed approach gives the following contributions:

- Detailed aerodynamic study focusing on blade flapping, induced and parasitic drag;
- Identification of the drag terms using a proposed extended method;
- Validation and analysis of the influence of the identified drag terms;
- Proposed structurally fast approach for solving the time optimal trajectory;
- Construction of the time optimal trajectory that minimizes the jerk;
- Design of a thrust vectoring controller, increase of performance and reduce of tracking delay.

1-5 Work Overview

The present work is structured in two main sets of Chapters.

The first set shows the main work Chapters of the work in hands. It starts with an introduction in the Chapter 1. Chapter 2 describes the wind influence and the associated aerodynamic effects related with the quadrotor. The time optimal trajectory is defined and constructed in Chapter 3. Chapter 4 presents the quadrotor mathematical model and the proposed cascade controller, while Chapter 5 shows the results. This set ends with Chapter 6 where a conclusion is presented and future work is suggested.

The second set of Chapters shows the Appendix Chapters that supported the work in hands. It starts by describing the framework that allowed to perform the experiments, showing the software, hardware and physical arena in Appendix A. Next, in Appendix B, the code used to implement both the time optimal trajectory and the proposed thrust vectoring controller is shown, and in Appendix C the quadratic programming problem used to achieve the time optimal trajectory is structured. Appendices D to G show the experimental data of all flights used to show the results, namely related with identification of the drag coefficients, influence of the coefficients on the controller, comparison of trajectories, and comparison of controllers. This set ends with Appendix H where the computational times are shown.

Chapter 2

Wind and Aerodynamic Effects

This Chapter aims at understanding the wind and associated aerodynamic effects in the quadrotor. It starts with a literature review in Section 2-1 and the following Sections describe each effect with importance for this work individually (see Sections 2-2 to 2-5). In Section 2-6 the other drag effects are shown.

In order to plan the optimal trajectory in the presence of wind fields it is firstly necessary to understand how the wind affects the performance of the quadrotor, during its maneuvers, and also how can the controller include the wind influence, instead of only considering it as a disturbance to be further rejected. In the absence of wind, the velocity of the quadrotor with respect to the ground, $\dot{\mathbf{r}}$, is equal to the velocity of the quadrotor with respect to the airflow, $\dot{\mathbf{r}}_\infty$. When this is not the case, and a stream of wind exists, then

$$\dot{\mathbf{r}} = \dot{\mathbf{r}}_\infty + \dot{\mathbf{r}}_w \tag{2-1}$$

where $\dot{\mathbf{r}}_w$ is the wind speed with respect to the ground. Equation (2-1) can also be written with velocity vectors as $\mathbf{v} = \mathbf{v}_\infty + \mathbf{v}_w$. As it will be seen later, all the aerodynamic effects are established with respect to the airstream, the apparent wind. For this fact, the wind effect is only to alter that airstream, and studying the airstream influence on the aerodynamics of the quadrotor alone is sufficient to withdraw the necessary conclusions when wind is present.

2-1 Literature Review

Detailed studies on aerodynamic effects are already available in the literature for the case of almost all aircraft types. Even for helicopters, which can be seen as a distant parent of quadrotors, precise studies have been performed, such as Prouty (2002) and Leishman (2006). In the latest, the aerodynamic properties of rotating blades are studied in detailed. However, for the case of small and more recent aircraft, such as quadrotors, the literature is more scarce.

Hoffmann et al. (2007) are one of the first to consider aerodynamic effects into the model of the quadrotor. There, drag effects such as the total thrust variation, blade flapping and airflow disruption are considered. Later on, Huang, Hoffmann, Waslander, and Tomlin (2009) also consider the first two aspects and Waslander and Wang (2009) modeled the total drag as a lumped factor linear with respect to the airspeed.

Bangura and Mahony (2012) are among the first who pursue a detailed explanation of all aerodynamic effects that affect the quadrotors dynamics. They consider in detail several aspects, such as blade flapping, induced drag, translational drag, profile drag, parasitic drag and also others such as ground effect and vertical descent.

Mahony et al. (2012) refers to induced drag and blade flapping and also treats them as a lumped parameter linear with respect to the horizontal air stream. Later they also refer to additional aerodynamic effects that are important for high-speed and highly dynamic maneuvers.

In the Allibert et al. (2014) approach, translational drag and blade flapping are considered bilinearly influenced by the thrust and the horizontal velocity with respect to the wind. They neglect parasitic drag since they operate near hovering. Moreover, they also consider the vertical thrust force variation due to the induced velocity.

More recently, Ryll, Bülthoff, and Giordano (2015) considers the aerodynamic effects and perform experiment to obtain the model corresponding coefficients. There, it is shown that the aerodynamic effects can be neglected. They also consider the induced drag and blade flapping as first order dynamic effects but still neglect them. However, they test the model with velocities smaller than 1 [m/s], which is clearly not adequate for this work.

To summarize, there are several aerodynamic effects that influence the quadrotor dynamics which have been reported, however briefly, in literature. The following Sections 2-2 to 2-5 analyze each effect separately, in a similar fashion as Bangura and Mahony (2012). Emphasis is given to the aerodynamic drag effects which are considered of significant importance for this work. The formulas have some simplifications, since the interest of this work is to obtain a relatively simple model of the wind/drag effects, in order to use them in the trajectory generation method proposed

in Chapter 3 and to include them in the controller to be designed in Chapter 4. Nevertheless, the introduced literature is suggested for further understanding of the drag effects.

2-2 Blade Flapping

Blade Flapping is a phenomena that occurs due to the flexibility of the rotor blades. When in translational movement, the tip advancing and retrieving blades, for each rotor, will not have the same equivalent velocity with respect to the airstream, as they will be shifted by a positive or negative δ angle (see Figure 2-1).

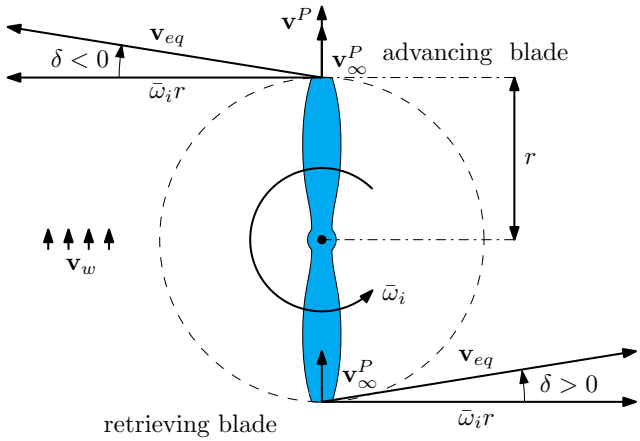


Figure 2-1: Blade Flapping angle illustration, top view of the rotor plane.

Due to the flexibility property, this causes the advancing blade to flex upwards, while the retrieving blade flexes downwards. In the literature, it is mentioned that this behavior creates roll and pitch moments at the blade root, and shifts the thrust vector $\mathbf{T}$ by a blade flapping angle β (see Figure 2-2). The angle β can be decomposed for each rotor in two components, parallel and perpendicular to the airstream, as

$$\beta_i^{\parallel} = -\frac{\mu_i a_1}{1 - \frac{1}{2}\mu_i^2}, \qquad \beta_i^{\perp} = -\frac{\mu_i a_2}{1 + \frac{1}{2}\mu_i^2}, \tag{2-2}$$

where $\mu_i = \frac{\mathbf{v}_\infty^P}{\bar{\omega}_i r}$ is the advancing ratio, i.e., the ratio between the airspeed projected in the rotor plane (similar to all rotors) and the rotor linear velocity. The constants a_1 and a_2 depend on blade properties, with $a_2 \ll a_1$, and one can consider that $\frac{1}{2}\mu_i^2 \ll 1$ due to the high rotation speed of the rotors (Bangura & Mahony, 2012).

Being so, most authors who describe blade flapping as a drag force due to aerodynamic effects (Mahony et al., 2012; Allibert et al., 2014) represent it as a function of thrust magnitude, rotor linear velocity and airspeed as

$$\mathbf{D}_{flap,i} = T_i \frac{\mathbf{A}_{flap}}{\bar{\omega}_i r} \mathbf{v}_\infty^P \tag{2-3}$$

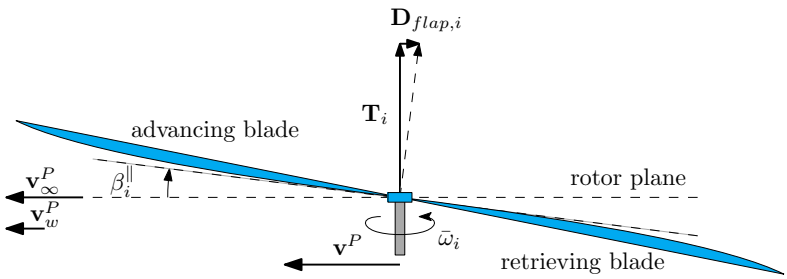


Figure 2-2: Blade Flapping angle illustration, lateral view of the rotor plane.

with

$$\mathbf{A}_{flap} = \begin{bmatrix} a_1 & a_2 & 0 \\ -a_2 & a_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

a constant matrix. Recently, Omari et al. (2013) uses the properties of blade theory to better estimate states of the state vector, using accelerometers and including blade flapping in the quadrotor model. There, blade flapping is not written as in Equation 2-3, although representing the same, since thrust can be written as a function of the rotor angular speed (see Section 4-1-2) as

$$T_i = k_T \bar{\omega}_i^2, \tag{2-4}$$

with k_T the thrust coefficient, leading thus to

$$\mathbf{D}_{flap,i} = \bar{\omega}_i \mathbf{A}'_{flap} \mathbf{v}_\infty^P \tag{2-5}$$

with $\mathbf{A}'_{flap}$ a constant matrix similar in structure to $\mathbf{A}_{flap}$ and defined by

$$\mathbf{A}'_{flap} = \begin{bmatrix} \frac{a_1 k_T}{r} & \frac{a_2 k_T}{r} & 0 \\ -\frac{a_2 k_T}{r} & \frac{a_1 k_T}{r} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a'_1 & a'_2 & 0 \\ -a'_2 & a'_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

2-3 Induced Drag

As discussed before, the blades are flexible, meaning that they can only bend to a certain angle β when a translational movement with respect to the airstream exists. Nevertheless, the blades still have stiff properties, meaning that when they produce produce lift, they produce an associated and proportional drag called induced drag. It is difficult to give a accurate representation of this drag force for quadrotors, but a similar representation can be shown for the case of a fixed wing aircraft, which is represented in Figure 2-3. In order to model it, a linear drag coefficient can be

introduced, leading to

$$\mathbf{D}_{ind,i} = T_i \mathbf{A}_{ind} \mathbf{v}_\infty^P, \tag{2-6}$$

with $\mathbf{A}_{ind}$ a constant matrix defined by k_{ind}, the induced drag coefficient, as

$$\mathbf{A}_{ind} = \begin{bmatrix} k_{ind} & 0 & 0 \\ 0 & k_{ind} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

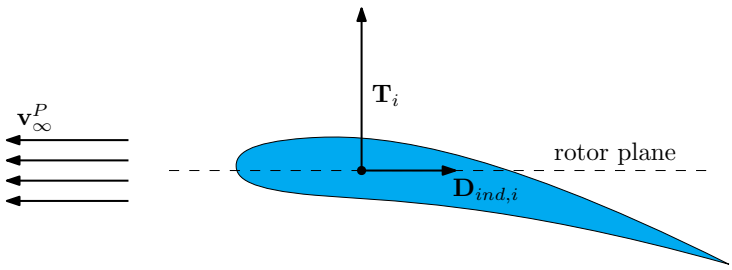


Figure 2-3: Induced drag illustration, cross-section view of the wing.

2-4 Lumped Drag Coefficient

The formulas obtained for blade flapping and induced drag are only with respect to one rotor. In the final model the four rotors should be added. For the induced drag it corresponds to linearly add the thrust of each motor, as $\sum_{i=1}^{4} T_i = T$. For the blade flapping case it is different. In Martin and Salaun (2010), the sum of all rotation speeds is found to be approximately constant during flights of near hovering thrust, leading to the blade flapping drag as

$$\mathbf{D}_{flap} = T_h \mathbf{A}''_{flap} \mathbf{v}_\infty^P \tag{2-7}$$

with $T_h \mathbf{A}''_{flap} \approx \sum_{i=1}^{4} \bar{\omega}_i \mathbf{A}'_{flap}$ a constant matrix and T_h the hovering thrust. When comparing with the drag caused by blade flapping and induced drag, it can be seen that the expressions are similar. Thus, it is possible to lump these two drag forces into only one, and obtain a drag force due to the mixed flexibility and stiffness of the blade propellers as

$$\mathbf{D}_{bla} = T \mathbf{A}_{bla} \mathbf{v}_\infty^P \tag{2-8}$$

with

$$\mathbf{A}_{bla} = \begin{bmatrix} a_{bla} & 0 & 0 \\ 0 & a_{bla} & 0 \\ 0 & 0 & 0 \end{bmatrix} \approx \begin{bmatrix} \frac{\sum \bar{\omega}_i a_1'}{T} + k_{ind} & \frac{\sum \bar{\omega}_i a_2'}{T} & 0 \\ -\frac{\sum \bar{\omega}_i a_2'}{T} & \frac{\sum \bar{\omega}_i a_1'}{T}' + k_{ind} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2-9}$$

the lumped drag matrix and a_{bla} the lumped drag coefficient.

2-5 Parasitic Drag

Parasitic Drag is caused by the non-lifting surfaces of the quadrotor in a translational movement. It must be considered when high velocities, usually greater that 10 [m/s] (Bangura & Mahony, 2012), are in question, since it is proportional to the square of the translational velocity. It depends on the shape and roughness of the object and an illustration is shown in Figure 2-4. It can be expressed as

$$\mathbf{D}_{par} = k_{par} \|\mathbf{v}_\infty\| \mathbf{v}_\infty, \tag{2-10}$$

with $k_{par} = \frac{1}{2}\rho S c_{D_{par}}$ depending on the non-lifting surfaces area, the air density and the non-lifting drag coefficient, with $c_{D_{par}}$ negligible with respect to the blades drag coefficient c_D (Bangura & Mahony, 2012), all assumed constant.

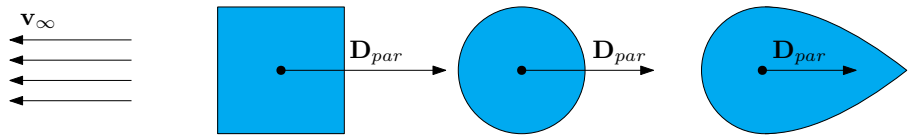


Figure 2-4: Parasitic drag illustration.

In environments where the wind speed and the ground speed are opposite, relatively big airspeeds will appear. Thus, the parasitic drag is of utmost importance in the trajectory generation phase, providing information on which direction to follow in order to minimize the global drag forces to optimize the trajectory.

2-6 Other Drag Forces

The blade flapping, induced and parasitic drag represent most of the important aerodynamic effects useful for this work. Nevertheless, other aspects are referred in the literature, which will be briefly described here for the sake of completeness:

- Total Thrust Variation - is the phenomena that occurs when a quadrotor undergoes a translational motion or changes the angle of attack, both with respect to the airflow. When this

occurs, there is an induced velocity given by

$$V_{ind} = \frac{V_h^2}{\sqrt{(V_\infty \cos\gamma)^2 + (V_\infty \sin\gamma + V_{ind})^2}} \tag{2-11}$$

where V_h is the induced velocity at hover thrust and γ is the angle of attack. The induced velocity will alter the necessary thrust to perform the maneuver, and the ratio between the ideal thrust and the thrust at hovering point can be then obtained as function of γ and V_∞ with

$$\frac{T}{T_h} = \frac{V_h}{V_{ind} + V_\infty \sin\gamma} \tag{2-12}$$

where T_h is the nominal thrust when hovering;

- Translational Drag - also known as momentum drag, it appears when the induced velocity passes the rotors, creating a drag proportional to the lift. For small velocities, it can be discarded when compared with induced drag or the blade flapping effect. The same happens for high velocities, as translational drag starts decreasing after a velocity threshold;

- Profile Drag - is the result of the transverse velocity of the rotor blades moving through the air. It is zero at hovering and usually does not depend on the angle of attack. It can be discarded when compared with the induced drag or the blade flapping effect;

- Ground Effect - it appears for flights near the ground, and results in a reduced necessary power to hover when closer to it. This effect will not appear in this work, as the quadrotor will not fly close to the ground;

- Vertical Descent - it appears when a vertical descent maneuver is in place, causing opposite directions in the induced velocity and airspeed. This can lead to vortices or turbulence generation.

Chapter 3

Time Optimal Trajectory

The problem addressed in this Chapter is to determine the time optimal trajectory that covers multiple waypoints and considers both the quadrotor dynamics and wind information. The problem is defined by m waypoints distributed around the quadrotor, and there are no restrictions either in time nor space. Nevertheless, the objective is to minimize the trajectory time and thus some inherent restrictions have to be weighed, which are mainly imposed by the drag forces that where studied in the previous Chapter 2 and by the quadrotor dynamics so that the controller to be designed in the next Chapter 4 follows the time optimal trajectory in an efficient manner.

In order to solve the time optimal problem, two approaches can be considered. One approach is to consider the problem as a global one, i.e., to put all objectives and constraints into a global optimization problem. However, this optimization problem would be too complex to process within an acceptable time for on-line applications. Therefore, a second approach that separates the problem into three smaller problems, or steps, will be considered.

The first smaller problem of the proposed approach is to estimate the optimal sequence of waypoints (see Figure 3-2) and it will be addressed in Section 3-1. In order to do so, geometric trajectory definitions can be used, including constraints related with wind, but not including the vehicles dynamics. The second smaller problem, solved in Section 3-2, is to determine the optimal trajectory between each of the sequential waypoints, and can be solved using a Quadratic Programming (QP) approach. The third smaller problem of the time optimal trajectory planning is to adapt the QP problem solution so that the airspeed magnitude requirement is fulfilled, and it is covered in Section 3-3.

3-1 Optimal Sequencing

When determining the optimal trajectory sequence a purely geometric approach with constraints will be used that includes wind information. Although this approach can be considered simple, it can critically reduce the computational time for this step in the trajectory generation process, and lend time to the second, and more complex, one. It is necessary to note that if this step is not considered, instead of having just one second smaller problem, there would be $m!$, increasing factorially the processing time. This would push the selecting process towards the final third step in the proposed trajectory generation method.

The possible trajectories are thus restrained to lines segments, circles and splines (Barrientos, Gutierrez, & Colorado, 2009) or to B-splines (Bouktir, Haddad, & Chettibi, 2008; Lorenz & Adolf, 2010). Another geometrical approach is to consider vector fields such as in Zhou and Schwager (2014). Since at this point the goal lies in selecting the optimal sequence, the complexity of the trajectory itself between each waypoint does not need to be significant.

For the remaining of this Section 3-1, the focus will be on line segments, that are evidently less complex. In Section 3-1-1 it will be seen that in constant wind fields a straight line is the correct approach and in Section 3-1-2 the solution to solve the optimal sequencing problem will be obtained.

3-1-1 The Zermelo's Problem

Line segments are a direct geometric solution but they can also be considered as a simplification of the Zermelo's problem (Bryson & Ho, 1975) since the wind is constant. In the original Zermelo's problem a sea current (analogue to wind velocity) pushes the ship directional velocity. The current is dependent on the ship position and the trajectory is modeled in state-space formulation as

$$f(x, y, u, v, \chi, t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} V\cos(\chi) \\ V\sin(\chi) \end{bmatrix} = \begin{bmatrix} V_\infty \cos(\pi) + u(x, y) \\ V_\infty \sin(\pi) + v(x, y) \end{bmatrix} \tag{3-1}$$

where V and V_∞ are the absolute and relative velocities of the ship, χ and π are the absolute and relative course angles, respectively. The states x and y are the position that defines the trajectory. By decomposing the velocity into relative and current velocity, the inputs became the current velocities u and v in the x and y direction respectively. This theory was firstly introduced for quadrotors by J.A. Guerrero (2013), but not in constant wind fields. Using the Pontryagin's Minimum Principle (Pontryagin, 1987) to optimize the trajectory time T the cost function becomes

$$\min T = \min \int_0^T 1 \, \mathrm{d}t, \tag{3-2}$$

the Hamiltonian is expanded as

$$H = 1 + \lambda^\top f = 1 + \lambda_x(V_\infty \cos(\pi) + u) + \lambda_y(V_\infty \sin(\pi) + v) \tag{3-3}$$

and the co-state Euler-Lagrange equations are

$$\dot{\lambda}_x = -\frac{\partial H}{\partial x} = -\lambda_x \frac{\partial u}{\partial x} - \lambda_y \frac{\partial v}{\partial x}, \tag{3-4}$$

$$\dot{\lambda}_y = -\frac{\partial H}{\partial y} = -\lambda_x \frac{\partial u}{\partial y} - \lambda_y \frac{\partial v}{\partial y}, \tag{3-5}$$

$$0 = \frac{\partial H}{\partial \pi} = V_\infty(-\lambda_x \sin(\pi) + \lambda_y \cos(\pi)) \Rightarrow \tan(\pi) = \frac{\lambda_y}{\lambda_x}. \tag{3-6}$$

As it can be seen in Equation (3-1), in the original Zermelo's problem the sea current velocity is dependent on the position and it influences the solution of the problem. In the problem regarded in this work, the wind velocity is constant and independent of the position, meaning that all derivatives with respect to the position are zero. This will lead to

$$\left.\begin{aligned} \dot{\lambda}_x = 0 \Rightarrow \lambda_x = a \\ \dot{\lambda}_y = 0 \Rightarrow \lambda_y = b \end{aligned}\right\} \Rightarrow \pi = c \tag{3-7}$$

for some constants a, b and c. Thus, the desirable relative angle, and consequently absolute angle, is constant and the optimal trajectory is a line segment between two consecutive points. The problem can be expanded to include the z direction and still straight lines will be the solution.

Minimizing the time to get the time optimal solution implies maximizing the velocity, and thus the velocity can be determined using the restriction on the limit velocity with respect to the airstream, so that parasitic drag effects can be diminished, as in

$$V_\infty^2 = \|\mathbf{v}_\infty\|^2 = \|\mathbf{v} - \mathbf{v}_w\|^2 \leq V_{lim}^2 \tag{3-8}$$

in which the quadratic form is used to allow for negative velocities. Since V_{∞_x}, V_{∞_y} and V_{∞_z} are orthogonal, and converting the inequality to equality to maximize the velocity, the solution of Equation (3-8) is

$$V = V_w g \pm \sqrt{(V_w g)^2 + V_{lim}^2 - V_w^2} \tag{3-9}$$

where

$$\begin{aligned} g(\chi_1, \chi_2, \alpha_1, \alpha_2) = \cos(\chi_1)\cos(\chi_2)\cos(\alpha_1)\cos(\alpha_2) + \\ + \sin(\chi_1)\cos(\chi_2)\sin(\alpha_1)\cos(\alpha_2) + \sin(\chi_2)\sin(\alpha_2) \end{aligned}$$

and the plus sign is always used since V was defined as a positive constant. The angles can be seen in Figure 3-1.

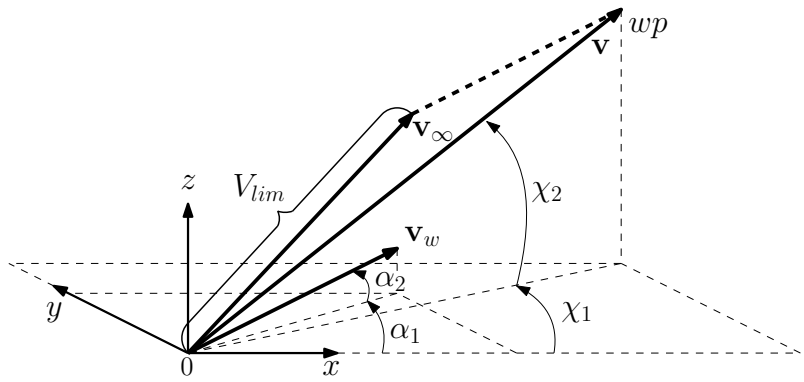


Figure 3-1: Illustration of the different velocities and angles.

With m waypoints, there are $m!$ possible waypoint sequences. For each possible sequence, the χ_{1i} and χ_{2i} angles are determined, the velocity V_i is computed and the time T_i is estimated. It is evident that depending on the geometry of the waypoints and on the wind velocity, different trajectory times will result. The Zeromelo's Problem goal is to obtain those times, while in the next Section 3-1-2 the optimal sequence which optimizes the time $T = \sum_{i=1}^{m} T_i$ will be determined using search algorithms.

3-1-2 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a well known NP-hard problem in the Artificial Intelligence field (Laporte, 1991). It aims at determining the optimal sequence between m possible cities that a salesman has to cover, with no restrictions on each city to visit first. Determining the optimal sequence of waypoints the quadrotor has to pass is analogous to choosing in which order the salesman travels between cities. In the original problem, the traveling times between each city are independent on the direction. In this work, however, when wind is present this condition does not prevails and for that reason an asymmetrical TSP is defined, which can be represented in a directed graph as in Figure 3-2.

In order to solve the TSP several approaches can be considered. The most intuitive is to evaluate all possible sequences and checking which one is the best, a brute force approach. Another approach is to write the problem in an Integer Linear Programming (ILP) formulation and numerically solve it (Papadimitriou & Steiglitz, 1998). One last approach is to use search algorithms to start from the origin and incrementally extend the sequence until the least costing solution is found (Rego, Gamboa, Glover, & Osterman, 2011).

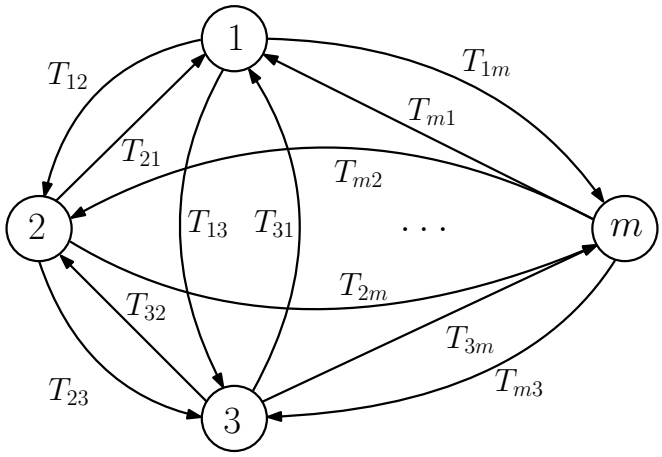


Figure 3-2: Directed Graph illustrating a generic distribution of m waypoints and traveling times.

The first approach works well in both computational time and memory for small m. However, when m increases the dimension increases by $\mathcal{O}(m!)$ and another approach must be chosen. The ILP formulation has a comparable computational complexity has the search algorithms, but it is slower. Therefore, for large m, the problem was solved using search algorithms.

Search Algorithms for the TSP

The search algorithms used were of two types. The first type, the uninformed (or blind) search, is general and allows to solve every search problem type. The second type, the informed (or heuristic) search, is dependent on the specific problem as it uses extra information that varies from problem to problem. Both methods start at the origin node (or a set of origin nodes). Every created, and still unexplored, node is part of the open-list of nodes. At each iteration the better node is chosen from the open-list, using a selection criteria, and it is expanded. The different search algorithms usually differ in the selection criteria. When a goal node is reached, the search stops.

A strategy to solve uninformed search problems is to, at each iteration, select the node with lower path cost, the so called uniform cost search. This method is optimal and complete, and thus is a suitable method for this work. Other methods are: breadth-first search and iterative deepening depth-first search, not used because in order to get to the solution the depth of the breadth must go all the way until the bottom (the optimal solution uses m points always), so almost all nodes are needed to be explored, and it is comparable with a brute-force algorithm; deep-first search and depth-limited search, not used because they are not optimal; and bidirectional search, not used because we the optimal node is not known *a priori*.

For this type of search algorithms, the total path cost is

$$f(i) = g(i) \tag{3-10}$$

where i is the $i$ths current waypoint in the sequence and

$$g(i) = \sum_{j=1}^{i} T_j \tag{3-11}$$

for the js waypoints in the sequence.

Informed search algorithms use heuristic functions that estimate the cost of going from a specific node until the goal node. This cost is obviously problem dependent and the challenge lies in how to evaluate it. If only the heuristic is used as the selecting criteria, then it is a greedy best-first search algorithm, which is not optimal. However, if the information of getting from the origin node until the current node, as well as the information of getting from the current node until the goal node are joined, it is an A^* search method. If the heuristic function used for the A^* search is admissible (never overestimates the true cost) then this approach is optimal.

For informed search algorithms the total path cost is

$$f(i) = g(i) + h(i) \tag{3-12}$$

where $h(i)$ is the heuristic function and in the greedy best-first search $g(i) = 0$.

Heuristics for the TSP

Two heuristic functions were used. The first one is admissible, and it is given as

$$h_1(i) = (m - i)T_{min} \tag{3-13}$$

where $m-i$ is the remaining number of waypoints to get to the goal node and T_{min} is the minimum traveling time between two nodes. The second heuristic method is not admissible, since it may happen that it overestimates the true cost of getting to the goal, and is given as

$$h_2(i) = K_{heu}(m - i)T_{ave} \tag{3-14}$$

where T_{ave} is the average traveling time between two nodes and K_{heu} is a constant, with $0 < K_{heu} \leq 1$. This approach is not optimal, but the solutions are still acceptable as it will be discussed later. The constant K_{heu} allows to tune the degree of optimality in contrast with the computational times. For a small K_{heu} the importance of the heuristic decreases and the performance gets similar to uniform search. For a big K_{heu} the performance moves towards a greedy best-first search.

3-2 Quadratic Programming

After determining the optimal sequence of waypoints, the next problem step is to determine the trajectory between each one of the waypoints. It is possible to obtain the trajectory by solving a QP problem (D. Mellinger & Kumar, 2011; D. Mellinger et al., 2012; Richter et al., 2013; Bipin et al., 2014) or using the Pontryagin's Minimum Principle, which results in an analytical solution (Mueller et al., 2013; Hehn & D'Andrea, 2015; Mueller et al., 2015). Other approaches to plan the trajectory between two points are Look-Up Tables (Corbets & Langelaan, 2007), and discretizing the model or the trajectory (Lai et al., 2006; Augugliaro et al., 2012). QP was considered due to its mathematical simplicity which leads to a general capability, since the solutions can be extrapolated to multiple goals (minimizing the velocity, acceleration, jerk or snap) within the same framework. Moreover, the precision in the polynomials of the QP and continuity in the derivatives can be guaranteed as far as wanted.

The trajectory is defined in Section 3-2-1 and the QP problem is generally constructed in Section 3-2-2. In Sections 3-2-3 to 3-2-6 the specifications of the QP problem are determined.

3-2-1 Trajectory Definition

There is a special class of systems, called Differentially Flat systems, for which there is an one-to-one correspondence between trajectories of a set of "flat outputs" and the full state space and inputs. This means that the trajectory can be defined in output space, and then mapped algebraically to the state and input space. These type of systems were introduced in Fliess, Lévine, Martin, and Rouchon (1992) and, because of their properties, are well suited for trajectory definition and generation. According to Nieuwstadt and Murray (1997), the differential flatness theory states that the set of outputs must be equal in number to the set of inputs, in order to allow a direct algebraic mapping. Generically, the non-linear state-space system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}, \mathbf{u}, t) \quad \mathbf{x} \in \mathbb{R}^n,\ \mathbf{u} \in \mathbb{R}^m \\ \mathbf{y}(t) &= h(\mathbf{x}, \mathbf{u}, t) \quad \mathbf{y} \in \mathbb{R}^m \end{aligned} \tag{3-15}$$

is differentially flat if it is possible to find outputs of $\mathbf{z} \in \mathbb{R}^m$ in the form

$$\mathbf{z}(t) = \zeta(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, ..., \mathbf{u}^{(l)}, t) \tag{3-16}$$

such that

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(\mathbf{z}, \dot{\mathbf{z}}, ..., \mathbf{z}^{(l)}, t) =: \mathbf{x}(\bar{\mathbf{z}}, t), \\ \mathbf{u}(t) &= \mathbf{u}(\mathbf{z}, \dot{\mathbf{z}}, ..., \mathbf{z}^{(l)}, t) =: \mathbf{u}(\bar{\mathbf{z}}, t), \end{aligned} \tag{3-17}$$

with $\mathbf{y}$ the tracking outputs and $\mathbf{z}$ the flat outputs. This means that every element of the state $\mathbf{x}$ and the input $\mathbf{u}$ are covered by $\mathbf{z}$.

Differentially flat systems are useful when explicit trajectory generation is required and for quadrotors the flat outputs are (Zhou & Schwager, 2014; D. Mellinger & Kumar, 2011) chosen as

$$\mathbf{z}(t) = [\ r_x(t) \quad r_y(t) \quad r_z(t) \quad \psi(t)\ ]^\top, \tag{3-18}$$

with the velocity, acceleration and the pitch θ and roll ϕ angles the other elements of the state vector to be withdrawn. A proof that quadrotors are differentially flat can be found in Zhou and Schwager (2014) for a model that doesn't use aerodynamic drag. When the aerodynamic forces are included the system is not differential flat because they depend on thrust which depends on the body orientation. Therefore, an explicit expression for the orientation as a function of the trajectory cannot be found. Nevertheless, we can still use differential flatness assuming that the thrust is constant and equal to hovering thrust, an assumption we will get again further.

Using differential flatness, a target time trajectory is defined as

$$\mathbf{r}_T(t) = [\ r_{T_x}(t) \quad r_{T_y}(t) \quad r_{T_z}(t) \quad \psi_T(t)\ ]^\top \tag{3-19}$$

Considering a general direction on $\mathbf{r}_T(t)$, furthermore referred as $\sigma_T(t)$, then m trajectories between the origin and each one of the m waypoints in that single direction can be defined as a n order time polynomial such that

$$\sigma_T(t) = \begin{cases} c_{10} + c_{11}t + c_{12}t^2 + \cdots + c_{1n}t^n & 0 \leq t \leq \Delta T_1 \\ c_{20} + c_{21}t + c_{22}t^2 + \cdots + c_{2n}t^n & \Delta T_1 \leq t \leq \Delta T_2 \\ \vdots & \\ c_{m0} + c_{m1}t + c_{m2}t^2 + \cdots + c_{mn}t^n & \Delta T_{m-1} \leq t \leq T \end{cases} \tag{3-20}$$

with $m(n+1)$ coefficients c_{ij} and ΔT_i the necessary time interval to get to the waypoint i from the origin and passing by the previews $i-1$ waypoints. In the previous definition time is monotonically increasing, meaning that the c_{i0} do not represent the m waypoints coordinates. To do so, a time-shift can be applied to Equation (3-20) such that

$$t = \Delta T_{i-1} + t_i \qquad \Delta T_{i-1} \leq t \leq \Delta T_i \tag{3-21}$$

and

$$\sigma_T(t) = \begin{cases} c_{10} + c_{11}t_1 + c_{12}t_1^2 + \cdots + c_{1n}t_1^n & 0 \leq t_1 \leq T_1 \\ c_{20} + c_{21}t_2 + c_{22}t_2^2 + \cdots + c_{2n}t_2^n & 0 \leq t_2 \leq T_2 \\ \vdots & \\ c_{m0} + c_{m1}t_m + c_{m2}t_m^2 + \cdots + c_{mn}t_m^n & 0 \leq t_m \leq T_m \end{cases} \tag{3-22}$$

The time trajectory of the position and its derivatives, without the i indexes for nomenclature convenience, is then given by

$$\begin{aligned} \sigma_T(t) &= c_0 + c_1 t + \cdots + c_n t^n \\ \dot{\sigma}_T(t) &= c_1 + 2c_2 t + \cdots + nc_n t^{n-1} \\ \ddot{\sigma}_T(t) &= 2c_2 + 6c_3 t + \cdots + n(n-1)c_n t^{n-2} \\ &\vdots \\ \frac{\mathrm{d}^k \sigma_T(t)}{\mathrm{d}t^k} &= \sum_{j=0}^{n-k} \frac{(k+j)!}{j!} c_{k+j} t^j \end{aligned} \tag{3-23}$$

3-2-2 Creating the Quadratic Programming Optimization Problem

The QP optimization problem is a special case of a nonlinear programming problem and is formulated to minimize or maximize the cost J of the vector $\mathbf{c}$ as

$$\begin{aligned} \min \ J(\mathbf{c}) &= \frac{1}{2}\mathbf{c}^\top \mathbf{H}\mathbf{c} + \mathbf{f}^\top \mathbf{c} \\ \text{subjected to } \mathbf{A}_{in}\mathbf{c} &\leq \mathbf{b}_{in} \\ \text{and } \mathbf{A}_{eq}\mathbf{c} &= \mathbf{b}_{eq} \end{aligned} \tag{3-24}$$

where $\mathbf{H}$ is a symmetrical matrix reflecting the quadratic form of the problem and $\mathbf{f}$ is a vector reflecting the linear one.

The objective function can be formulated as a function of the k derivative of the position (D. Mellinger & Kumar, 2011; D. Mellinger et al., 2012), a sum of different derivatives (Richter et al., 2013) or the Boor control points for the B-Spline (Bipin et al., 2014). If the objective is to minimize the power of a general derivative of the position then it can be formulated as

$$\sigma_T^*(t) = \min \int_0^T \sum_{i=1}^{4} K_i \left(\frac{\mathrm{d}^k \sigma_{T_i}(t)}{\mathrm{d}t^k} \right)^2 \mathrm{d}t \tag{3-25}$$

where the superscript denotes the optimal solution, and with K_i a constant to turn the integral

dimensionless. Since the trajectories are decoupled in all directions, the above can be seen as four different optimization problems, with $\mathbf{c}$ the vector with the concatenation of the referred c_{ij}. By choosing to minimize a derivative of the position, the $\mathbf{H}$ matrix will result in a block diagonal of m independent $(n+1) \times (n+1)$ matrices as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_m \end{bmatrix} \tag{3-26}$$

and the $\mathbf{f}$ vector will be null. Nevertheless, the waypoints are still related by imposing constraints in the continuity up until the p derivative of the position, constraints to be imposed by $\mathbf{A}_{eq}$ and $\mathbf{b}_{eq}$. Maximum or minimum values for acceptable velocities, accelerations, jerks or snaps can also be imposed using $\mathbf{A}_{in}$ or $\mathbf{b}_{in}$. However, minimizing a derivative of the position is already indirectly imposing acceptable trajectories, and in this work the inequality constraint is neglected.

3-2-3 Minimizing the Velocity with respect to the Wind

In order to minimize the effects of the parasitic drag, an interesting cost function is thus to minimize the power of the velocity of the quadrotor with respect to the wind. Considering a general direction $\sigma_T(t)$, the optimal trajectory solution is given by

$$\sigma_T^*(t) = \min \int_0^T \left(\dot{\sigma}_T(t) - V_{w,\sigma}\right)^2 \, \mathrm{d}t \tag{3-27}$$

with $V_{w,\sigma}$ the wind velocity in the direction of the trajectory direction σ_T. Equation (3-27) can be expanded, and the fact that wind is constant can be used to obtain

$$\sigma_T^*(t) = \min \int_0^T \dot{\sigma}_T^2(t) \, \mathrm{d}t - 2V_{w,\sigma} \int_0^T \dot{\sigma}_T(t) \, \mathrm{d}t. \tag{3-28}$$

The first term will give origin to the $\mathbf{H}$ matrix while the second term will give origin to the $\mathbf{f}$ vector. The necessary constraint to this problem is the displacement in the position, that is

$$\int_0^T \dot{\sigma}_T(t) \, \mathrm{d}t = \Delta\sigma_T \tag{3-29}$$

where the first term will give origin to $\mathbf{A}_{eq}$ and the second one to $\mathbf{b}_{eq}$. Thus, comparing Equation (3-29) with Equation (3-28) one can see that, by construction, the second part is a constant term and thus irrelevant to the minimization problem. Finally, it can be concluded that minimizing

Equation (3-27) is the same as

$$\sigma_T^*(t) = \min \int_0^T \dot{\sigma}_T^2(t)\ \mathrm{d}t \tag{3-30}$$

and a conclusion can be withdrawn: wind has no influence in the trajectory formulation. Thus, other derivatives of the position can be chosen that better represent the quadrotors dynamics.

3-2-4 What Derivative to Minimize?

It was proven that minimizing the power of the velocity with respect to the wind is not relevant. Thus minimizing the following position derivatives powers, which have influence on the quadrotors dynamics (see Equation (4-13)), is proposed: acceleration, that is directly implied in the quadrotor model and can be linked to thrust; jerk, the first derivative of the acceleration which directly corresponds to the quadrotor body rates; and snap, second derivative of the acceleration and proportional to the motor commands and attitude accelerations.

Acceleration is the simplest of all but it is also the most naive to define as the goal, since it will imply the less possible thrust, thus constraining excessively the aggressiveness of the trajectory. Smooth trajectories are desirable, but with some aggressiveness to explore the time optimal possible trajectory, and this is acceptable due to the thrust vectoring controller to be proposed in Section 4-4, which will allow a great flexibility in Euler angles.

According to Mueller et al. (2015) the jerk cost is a better representative of the aggressiveness of the true system inputs, and the jerk, like the acceleration, has a direct link with thrust. Plus, they say they can bound the body rates as functions of the jerk and the thrust. Moreover Hehn and D'Andrea (2015) affirm that maintaining constraints on the acceleration and jerk leads to a continuous thrust during the maneuver, which is then supported by (Bipin et al., 2014) when affirming that constraints on jerk are necessary for a smooth trajectory. Finally, it has also been studied (Flash & Hogan, 1985) that humans tend to minimize the integral of the square magnitude of the jerk, in order to increase their performance in motion coordination.

According to Richter et al. (2013) minimum snap trajectories have also been proven effective to generate quadrotor trajectories, due to the linkage in the motors commands and body rate derivatives. The same says D. W. Mellinger (2012) and supports that with a study of human movement (Kawato, Maeda, Uno, & Suzuki, 1990) that claims that the best criterion for modeling motion is minimizing the integral of the square norm of torque torque derivatives, which are related to snap.

In this work, minimizing the power of jerk was defined as the goal, since it can balance all the aspects discussed before. Moreover, minimizing the jerk relates with minimizing the variation in the acceleration that is linked to the Euler angles caused by drag.

3-2-5 Degree of the Polynomial and Constraints

The degree of the polynomial was set to $n = 5$ so that there are still sufficient coefficients when minimizing the jerk power. In the case that $n = 5$ the jerk is a second order time polynomial which is sufficient to give reliable results. Moreover, continuity constraints were imposed until $p = 2$, in order to guarantee continuity at least until the second derivative of the position, the acceleration.

3-2-6 Optimal Times

The time segments T_1, T_2 to T_m are constants and the optimization process is solved using the expected times obtained when solving the TSP problem. However, those times can be adapted by allowing more time to one segment than another, thus reducing the overall cost J while maintaining the total trajectory time T. Thus, an iterative process to optimize the time segments with a gradient descent method using a backtracking line search was used (D. Mellinger & Kumar, 2011).

3-3 Constraint on the Maximum Velocity

At this point, the total time optimal trajectory time T is only an estimate given by solving the TSP, meaning that the resulting maximum velocity of the QP problem can be greater or smaller than V_{lim} in some time intervals. Due to the non dimensional spatial property included by construction in the QP formulation, a new loop can be performed, by incrementally giving or taking more time to T, without the need to solve the QP problem again. By doing this third step of the approach in the planning of the total time optimal trajectory, the velocity is guaranteed to respect V_{lim} so that the parasitic drag is minimized.

Chapter 4

Thrust Vectoring Controller

From the two previous Chapters 2 and 3 it is known that the quadrotor dynamics are linked to aerodynamic effects and trajectory dynamics. In this Chapter, the model of the quadrotor will be obtained, in Section 4-1, and the quadrotor used in this work will be presented, in Section 4-2. Usual controllers neglect the aerodynamic effects and are generally naive when following a trajectory, since they only account for discrete values of desired position setpoints. Thus, powerful and real-time information that can be used to design the controller is being neglected for the usual cases, such as the the Paparazzi UAV embedded controller. Paparazzi UAV is an open-source platform for UAV development that will be used as framework for this work, and will be analyzed in Section 4-3. However, the referred time optimal trajectory information and the aerodynamic effects can be used in an optimal way using a proposed thrust vectoring controller approach, to be following presented in Section 4-4.

4-1 Model of the Quadrotor

The configuration of a typical quadrotor consists of four rigid blade propeller motors mounted in a "+" or "×" pairwise symmetrical counter-rotating fashion, and attached to the quadrotor rigid body as it can be seen in Figure 4-1.

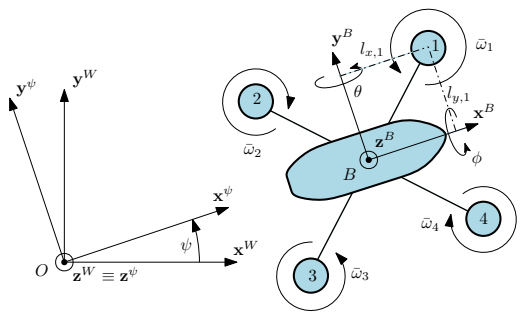


Figure 4-1: Top view of a typical quadrotor in "×" configuration with reference frames scheme.

4-1-1 Frames of Reference and other Formalisms

The quadrotor center of mass is B with mass m and inertia matrix $\mathbf{J}$. The inertial world frame of reference is $\mathcal{W} = \{O, \mathbf{x}^W, \mathbf{y}^W, \mathbf{z}^W\}$ and the frame attached to the quadrotor, the body frame of reference, is $\mathcal{B} = \{B, \mathbf{x}^B, \mathbf{y}^B, \mathbf{z}^B\}$. To represent the relative frame orientation a rotation matrix from $\mathcal{B}$ to $\mathcal{W}$ is used, defined as $\mathbf{R}_B^W = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x = \mathbf{R}$. The inverse rotation is $\mathbf{R}_W^B = (\mathbf{R}_B^W)^{-1} = (\mathbf{R}_B^W)^\top$ since the rotation matrix is orthogonal. $\mathbf{R}_B^W$ is composed of three consecutive $Z-Y-X$ rotations of the Euler angles yaw, pitch and roll, with

$$\mathbf{R}_z = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_y = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \quad \mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{4-1}$$

using c and s for the $\sin(\cdot)$ and $\cos(\cdot)$ notation, such that

$$\mathbf{R}_B^W = \begin{bmatrix} c\theta c\psi & s\theta s\phi c\psi - c\phi s\psi & s\theta c\phi c\psi + s\phi s\psi \\ c\theta s\psi & s\theta s\phi s\psi + c\phi c\psi & s\theta c\phi s\psi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}. \tag{4-2}$$

With this convention of a right-handed coordinated system, ψ is positive if the quadrotor is yawing left, θ is positive if the quadrotor is pitching down and ϕ is positive if the quadrotor is rolling right. The canonical basis of $\mathbb{R}^3$ is $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. The quadrotor position, velocity and acceleration, in the inertial frame, are $\mathbf{r}$, $\dot{\mathbf{r}}$ and $\ddot{\mathbf{r}}$, while the wind velocity is $\mathbf{v}_w$. The angular velocity, measured in the body frame, is $\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^\top$. The roll rate is measured directly in the body frame, the pitch rate is measured in an intermediate frame rotated by the roll angle and the yaw rate is measured

in the next intermediate frame rotated by pitch and roll angles, such that, in body coordinates

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{I} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_x^\top \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + (\mathbf{R}_y \mathbf{R}_x)^\top \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4-3}$$

where $\mathbf{I}$ is a 3×3 identity matrix. The Euclidean norm is $\|\mathbf{a}\| := \sqrt{a_1^2 + a_2^2 + a_3^2}$, the dot-product is $\mathbf{a} \cdot \mathbf{b} := a_1 b_1 + a_2 b_2 + a_3 b_3$ and the notation $\mathbf{a}_\times \mathbf{b} := \mathbf{a} \times \mathbf{b}$ relates to the skew-symmetric matrix associated with the cross-product, such that

$$\mathbf{a}_\times := \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{4-4}$$

4-1-2 Motor Dynamics

The motors are symmetrically attached to B at a distance of $\mathbf{l}_i = \begin{bmatrix} l_{x,i} & l_{y,i} & 0 \end{bmatrix}^\top$. The motors dynamics are assumed relatively fast when compared to the rigid body dynamics and aerodynamics effects, so they can be neglected. Considering the body frame $\mathcal{B}$. each rotor rotating at $\bar{\omega}_i$ generates a thrust $\mathbf{F}_{T,i} = T_i \mathbf{e}_3$, with $T_i = k_T \bar{\omega}_i^2$, and an aerodynamic torque $\mathcal{T}_i = \lambda_i k_\tau \bar{\omega}_i^2 \mathbf{e}_3$. The thrust and torque constants, k_T and k_τ, depend on blade properties and the λ_i constant depends on the direction of rotation of the blade ($\lambda_i = 1$ for clockwise and $\lambda_i = -1$ for counterclockwise).

4-1-3 Dynamic Model

The dynamic model of the quadrotor, which represents the Newton-Euler Equations associated with force, orientation and momentum, is given by

$$\begin{cases} m\ddot{\mathbf{r}} = \mathbf{R} \sum_{i=1}^4 \mathbf{F}_i - mg\mathbf{e}_3 + \mathbf{F}_{aero} \\ \dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}_\times \\ \mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}_\times \mathbf{J}\boldsymbol{\omega} + \sum_{i=1}^4 (\mathcal{T}_i + \mathbf{l}_i \times \mathbf{F}_i) + \mathcal{T}_{aero} \end{cases} \tag{4-5}$$

All forces acting on the quadrotor, except for $\mathbf{F}_{aero}$, are the sum of the thrust and blade flapping and induced drag, lumped in one drag force as in Equation (2-8). Thus

$$\sum_{i=1}^4 \mathbf{F}_i = \sum_{i=1}^4 (\mathbf{F}_{T,i} + \mathbf{D}_{bla,i}) = T\mathbf{e}_3 + \sum_{i=1}^4 T_i \mathbf{A}_{bla} \mathbf{v}_{\infty,i}^P \tag{4-6}$$

with

$$\sum_{i=1}^{4} T_i \mathbf{A}_{bla} \mathbf{v}_{\infty,i}^{P} = T A_{bla} \mathbf{R}^{\top} (\dot{\mathbf{r}} - \mathbf{v}_w). \tag{4-7}$$

The aerodynamic force is the parasitical drag force (see Equation (2-10)) given by

$$\mathbf{F}_{aero} = \mathbf{D}_{par} = k_{par} \|\mathbf{v}_\infty\| \mathbf{v}_\infty = k_{par} \|\dot{\mathbf{r}} - \mathbf{v}_w\| (\dot{\mathbf{r}} - \mathbf{v}_w) \tag{4-8}$$

The torque terms are

$$\sum_{i=1}^{4} (\mathcal{T}_i + \mathbf{l}_i \times \mathbf{F}_i) = \mathcal{T} + \sum_{i=1}^{4} T_i \mathbf{l}_i \times \mathbf{A}_{bla} \mathbf{v}_{\infty,i}^{P} \tag{4-9}$$

with

$$\mathcal{T} = \sum_{i=1}^{4} \lambda_i k_\tau \bar{\omega}_i^2 \mathbf{e}_3 - k_T \bar{\omega}_i^2 \mathbf{l}_i \times \mathbf{e}_3 \tag{4-10}$$

and

$$\sum_{i=1}^{4} T_i \mathbf{l}_i \times \mathbf{A}_{bla} \mathbf{v}_{\infty,i}^{P} = 0 \tag{4-11}$$

due to symmetrical properties of the quadrotor. Assuming that the external aerodynamic forces cause no torque on the quadrotor then

$$\mathcal{T}_{aero} \approx 0. \tag{4-12}$$

Finally, the full system can be written, with manipulated inputs given by the rotors thrust and torque as

$$\begin{cases} m\ddot{\mathbf{r}} = T\mathbf{R}\mathbf{e}_3 - mg\mathbf{e}_3 + T\mathbf{R}\mathbf{A}_{bla}\mathbf{R}^{\top}(\dot{\mathbf{r}} - \mathbf{v}_w) + k_{par}\|\dot{\mathbf{r}} - \mathbf{v}_w\|(\dot{\mathbf{r}} - \mathbf{v}_w) \\ \dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}_\times \\ \mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}_\times \mathbf{J}\boldsymbol{\omega} + \mathcal{T} \end{cases} \tag{4-13}$$

where the thrust magnitude T and torque vector $\mathcal{T}$ relation with the angular velocity of the blade propellers is represented as

$$\begin{bmatrix} T \\ \mathcal{T} \end{bmatrix} = \begin{bmatrix} k_T & k_T & k_T & k_T \\ k_T l_y & k_T l_y & -k_T l_y & -k_T l_y \\ -k_T l_x & k_T l_x & k_T l_x & -k_T l_x \\ k_\tau & -k_\tau & k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} \bar{\omega}_1^2 \\ \bar{\omega}_2^2 \\ \bar{\omega}_3^2 \\ \bar{\omega}_4^2 \end{bmatrix}. \tag{4-14}$$

4-2 Parrot Bebop

The quadrotor used in this work is the Parrot Bebop (see Figure 4-2), a commercially available quadrotor[1] used worldwide as a research platform. It uses a "×" motor configuration and has features that are adequate for the work at hand: resistant structure, also protected by side bumpers to increase safety; lightweight; powerful dual-core CPU; and wide variety of sensors.

Figure 4-2: Parrot Bebop with side bumpers.

4-3 Paparazzi UAV - Reference Generator and PID Framework

The Paparazzi UAV is an open-source platform firstly conceived as a tool for development of standard fixed-wing UAVs[2]. Paparazzi is also widely used and it was chosen since it incorporates a standard reference generator and a PID controller with whom the performances will be compared.

For those UAVs there is a traditional separation between longitudinal and lateral movement control. The heritage that the rotorcraft autopilot received is currently also implemented and for that reason, there are two main and independent control loops (see Figure 4-3) . The vertical loop is responsible for the commanded thrust while the horizontal loop is responsible for the commanded Euler angles. Afterwards, the Motor-Mixing Unit (see Equation (4-14)) is responsible for mapping those commands into the motors rotating speeds.

4-3-1 Vertical and Horizontal Control Loops in Paparazzi

The vertical control loop is composed by a reference generator and a vertical controller. The reference input is the z vertical setpoint, a discrete value of desired altitude, and the output is the desired thrust. The horizontal control loop is similar but also includes a stabilization controller. The reference inputs are the x and y lateral setpoints and the outputs are the desired Euler angles.

The reference generators create discrete point to point trajectory steps followed by second order low-pass filtering, used so that there are no aggressive requests of velocity and acceleration. The

[1] http://www.parrot.com/products/bebop-drone/
[2] http://wiki.paparazziuav.org/wiki/Main_Page

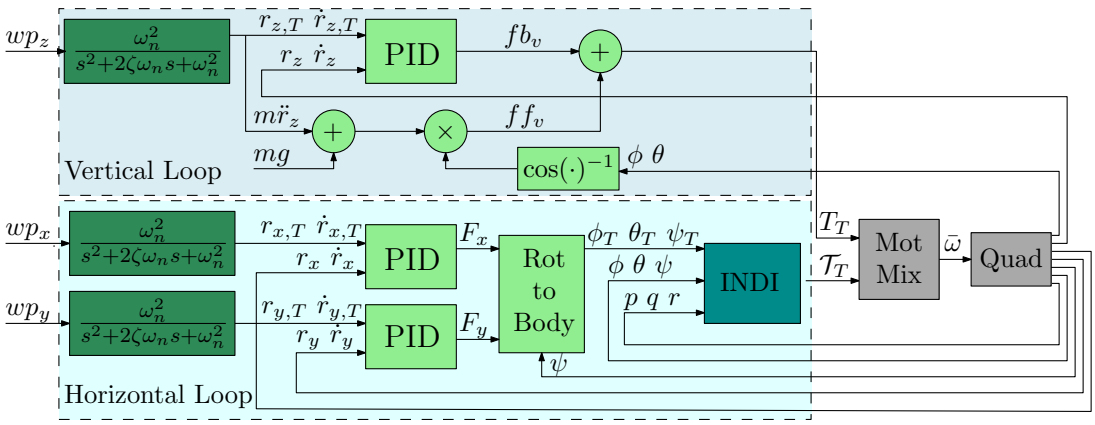


Figure 4-3: Illustration of a standard controller.

reference generators will be further referred as the standard low-pass filtering case. The vertical and lateral controllers are typical PID controllers, with feedback action for the position and velocity, and feedforward action for the acceleration. The stabilization controller currently implemented uses an Incremental Non-Linear Dynamic Inversion (INDI). Both Control Loops can be seen in Figure 4-3.

4-3-2 Incremental Non-Linear Dynamic Inversion

The INDI controller is an attitude based controller developed by Smeur, Chu, and Croon (2016) at Delft University of Technology for the Parrot Bebop. The INDI approach is an improvement of the Non-Linear Dynamic Inversion in order to increase the controller robustness as it calculates increments in the control action based solely in the desirable increment in the angular acceleration, avoiding problems with unmodeled dynamics. The INDI inputs are the Euler angles and derivatives, together with the body rates, and it outputs the rotors torque (see Equation (4-14)). The inclusion of an optimal trajectory generator and the thrust vectoring controller, together with the INDI is thus a major step towards an optimal quadrotor controller.

4-4 New Controller Approach

Due to the loops separation in Paparazzi the code implementation firstly performs the horizontal loop and afterwards the vertical one. The only part in which the loops may be seen as linked is through the $(\cos(\phi)\cos(\theta))^{-1}$ term in the vertical loop, although these angles being the measured ones, and not the desired. This separation will then impose a natural oscillatory behavior by the exchange in vertical and horizontal performance. Moreover, in the usual PID controllers there

is no information about the aerodynamic effects, manly composed of drag terms, and critically affected by wind.

For these reasons, the performance of the current controller is obviously not optimal. Thus, it is proposed the replacement of the first two outer loops by a single one, which weights both the vertical and horizontal movement at the same time. The current reference generator is replaced by the equations discussed in Chapter 3, meaning that instead of a low-pass filter there will be optimal trajectories, which minimize the jerk power. *A priori* information of the desirable velocity and acceleration is used so that there is no theoretical delay, which does not happen when a low-pass filter is implemented. Moreover, a thrust vectoring implementation allows to determine the desirable orientation of the quadrotor so that the thrust vector, in the direction of $\mathbf{z}^B$, is aligned with the desired force. The stabilization controller based on the INDI implementation is also used.

4-4-1 Thrust Vectoring Equations

The proposed controller was inspired by D. Mellinger and Kumar (2011), but divergences exist, which will be further described. Extending the PID approach of the traditional controller to include a double-derivative term for the acceleration and incorporating aerodynamic effects one has

$$\begin{aligned}\mathbf{F}_T = K_p\mathbf{e}_p + K_{pi}\int_0^{\Delta t}\mathbf{e}_p\ \mathrm{d}t + K_v\mathbf{e}_v + K_a\mathbf{e}_a + mg\mathbf{z}^W + \\ + m\ddot{\mathbf{r}}_T + T\mathbf{R}\mathbf{A}_{bla}\mathbf{R}^\top(\dot{\mathbf{r}} - \mathbf{v}_w) + k_{par}\|\dot{\mathbf{r}} - \mathbf{v}_w\|(\dot{\mathbf{r}} - \mathbf{v}_w)\end{aligned} \tag{4-15}$$

with $\mathbf{F}_T$ the desirable thrust. The first four terms are feedback related and the others are feed-forward related. Equation (4-15) shows the first divergence with D. Mellinger and Kumar (2011), since here the drag terms are accounted, whereas they neglect them. The feedback errors are obtained as

$$\begin{bmatrix}\mathbf{e}_p\\ \mathbf{e}_v\\ \mathbf{e}_a\end{bmatrix} = \begin{bmatrix}\mathbf{r}_T - \mathbf{r}\\ \dot{\mathbf{r}}_T - \dot{\mathbf{r}}\\ \ddot{\mathbf{r}}_T - \ddot{\mathbf{r}}\end{bmatrix}. \tag{4-16}$$

The desirable orientation of the quadrotor $\mathbf{z}^B$ axis is given by

$$\mathbf{z}_T^B = \frac{\mathbf{F}_T}{\|\mathbf{F}_T\|}. \tag{4-17}$$

With the desirable yaw angle, from the flat outputs, one can derive the intermediate $\mathbf{x}^\psi$ orientation

of the first reference frame rotation, given by

$$\mathbf{x}_T^{\psi} = \mathbf{R}_z \mathbf{e}_1 = \begin{bmatrix} \cos(\psi_T) & \sin(\psi_T) & 0 \end{bmatrix}^\top . \tag{4-18}$$

The desirable orientation of the quadrotor $\mathbf{y}^B$ axis can be obtained considering the orthogonality property of the axes

$$\mathbf{y}_T^B = \frac{\mathbf{z}_T^B \times \mathbf{x}_T^{\psi}}{\|\mathbf{z}_T^B \times \mathbf{x}_T^{\psi}\|} \tag{4-19}$$

and the desirable orientation of the quadrotor $\mathbf{x}^B$ axis is

$$\mathbf{x}_T^B = \mathbf{y}_T^B \times \mathbf{z}_T^B. \tag{4-20}$$

Equation (4-19) is only valid if $\mathbf{z}_T^B \times \mathbf{x}_T^{\psi} \neq 0$ meaning that the axes are not collinear. In order for this to happen the quadrotor should be rotated on its side, which for practical reasons is never the case. The three axes define the desirable rotation matrix in the same way as in Equation (4-2) as

$$\mathbf{R}_T = \begin{bmatrix} \mathbf{x}_T^B & \mathbf{y}_T^B & \mathbf{z}_T^B \end{bmatrix} \tag{4-21}$$

and the other desirable Euler angles can finally be obtained using

$$\begin{bmatrix} \phi_T \\ \theta_T \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(\mathbf{R}_{T,32}, \mathbf{R}_{T,33}\right) \\ \operatorname{atan2}\left(-\mathbf{R}_{T,31}, \sqrt{\mathbf{R}_{T,32}^2 + \mathbf{R}_{T,33}^2}\right) \end{bmatrix}. \tag{4-22}$$

Equation (4-22) shows the other major divergence from D. Mellinger and Kumar (2011), since here the angles are computed from the rotation matrix and inputted to the INDI stabilizer. In their case, they compute the *vee map* error between $\mathbf{R}_T$ and $\mathbf{R}$, as well as the error in the angular velocities $\boldsymbol{\omega}_T$ and $\boldsymbol{\omega}$ to feed directly to the inputs $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$. The proposed approach is less subjected to measurement errors, since the tracking of the angles is made directly through the INDI stabilizer, instead of propagating the error in a rotation matrix

Finally, the desirable thrust magnitude can be obtained by projecting the desirable thrust into the quadrotor $\mathbf{z}^B$ axis

$$T_T = \mathbf{F}_T \cdot \mathbf{z}^B \tag{4-23}$$

.

4-4-2 Three Stages Cascade Controller

With the equations derived in Section 4-4-1, together with the equations derived from Chapter 3 and including the INDI stabilizer, the block diagram of the proposed three stages cascade controller can be represented, as in Figure 4-4.

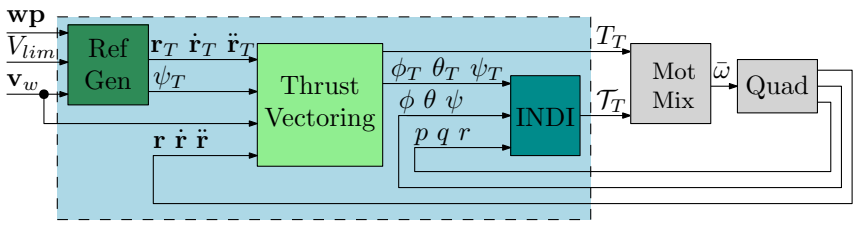


Figure 4-4: Illustration of the proposed Cascade Controller.

Chapter 5

Results

In this Chapter the results of the work in hands will be analyzed, in a fashion that follows the previous Chapters sequence. Firstly, in Section 5-1, the lumped and parasitic drag coefficients will be identified through flight data, and their influence on the controller will be shown in Section 5-2. Next, the optimal approach when solving the TSP will be determined in Section 5-3, the influence of the wind in the time trajectory generation will be analyzed in Section 5-4, and in Section 5-5 an overview of the trajectory formulation in computational times will be discussed. Later on, in Section 5-6, the time optimal trajectory generation is compared versus a standard low-pass reference generator. Finally, in Section 5-7, the thrust vectoring controller is compared with a PID controller.

5-1 Identification of the Drag Coefficients

In order to determine the coefficients related with aerodynamic properties of the quadrotor, discussed in Chapter 2, a set of flight tests were conducted. The purpose of these tests is to identify the lumped drag coefficient, $\mathbf{A}_{bla}$ that combines blade flapping and induced drag, and the second order drag coefficient, k_{par}.

5-1-1 Proposed Identification Equations

Omari et al. (2013) proposed a method to obtain the lumped drag coefficient based on the measures of the acceleration and the velocity in body coordinates. Extending their work, an approach to

obtain the parasitic drag coefficient is proposed, by including it in Equation (5-1). To obtain the coefficients, Equation (4-13) was used, and the movement of the quadrotor was constrained to the x axis direction, which allows some crucial simplifications. By restraining the movement, and assuming equilibrium of forces, the only non-zero Euler angle is the the pitch angle θ, and

$$m\begin{bmatrix} a_x \\ 0 \\ 0 \end{bmatrix} = T\begin{bmatrix} s\theta \\ 0 \\ c\theta \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + Ta_{bla}\begin{bmatrix} c^2\theta \\ 0 \\ -s\theta c\theta \end{bmatrix} v_x + k_{par}\|v_x\|\begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix}. \tag{5-1}$$

Furthermore, using the small angles approximation for θ ($\cos(\theta) \approx 1$, $\sin(\theta) \approx \theta$) and assuming

$$Ta_{bla}\theta << T - mg, \tag{5-2}$$

a force equilibrium is achieved in the z axis naturally when $T = mg$, thus leading to the equation in the x axis as

$$ma_x = mg\theta + mga_{bla}v_x + k_{par}\|v_x\|v_x = mg\theta + D_x, \tag{5-3}$$

showing a direct correspondence between velocity, acceleration and pitch angle that allows to obtain the desired coefficients by means of a statistical fitting. From every measure of the collected flight data the specific drag force was determined as

$$d_{x,i} := \frac{D_{x,i}}{m} = a_{x,i} - g\theta_i \ \left[\frac{\mathrm{m}}{\mathrm{s}^2}\right] \tag{5-4}$$

and a linear Least Squares (LS) fitting was used with cost

$$J = \sum_{i=1}^{N}\left(d_{x,i} - \hat{d}_{x,i}(v_{x,i})\right)^2 = \sum_{i=1}^{N}\left(d_{x,i} - g\hat{a}_{bla}v_{x,i} - \frac{\hat{k}_{par}}{m}\|v_{x,i}\|v_{x,i}\right)^2 \tag{5-5}$$

producing the following estimate of the drag coefficients

$$\begin{bmatrix} g\hat{a}_{bla} \\ \frac{\hat{k}_{par}}{m} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} v_{x,i}^2 & \sum_{i=1}^{N}\|v_{x,i}\|v_{x,i}^2 \\ \sum_{i=1}^{N}\|v_{x,i}\|v_{x,i}^2 & \sum_{i=1}^{N}\|v_{x,i}\|^2 v_{x,i}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{N} v_{x,i}d_{x,i} \\ \sum_{i=1}^{N}\|v_{x,i}\|v_{x,i}d_{x,i} \end{bmatrix} \tag{5-6}$$

with N the number of samples.

5-1-2 Methodology

For the experimental tests, the tracking position was set to be a ramp, resulting in a step in velocity and null acceleration. Due to the drag forces, an expected pitch angle can be computed.

An illustration can be seen in Figure 5-1 for a selected flight with velocity $V_T = 2$ [m/s]. The movement can be qualitatively described as follows: the quadrotor starts at the center of the test arena and moves diagonally to the corner, so it can maximize the flight distance; it does this at a constant negative V_T; then the quadrotor waits six seconds to stabilize its position and orientation; afterwards it moves to the other corner at positive V_T. Due to the space limitations in the test arena, the corners positions were restricted to $r_x \in \{\pm 3.0,\ \pm 3.5,\ \pm 4.0\}$ [m] accordingly to V_T, which was varied in the set $V_T \in \{0.5,\ 1.0,\ 1.5,\ \cdots,\ 4.0\}$ [m/s].

5-1-3 Results - Identification of the Drag Coefficients

For each V_T, seven flights were performed. The average result for $V_T = 2$ [m/s] can be seen in Figure 5-1, which allow to withdraw some conclusions:

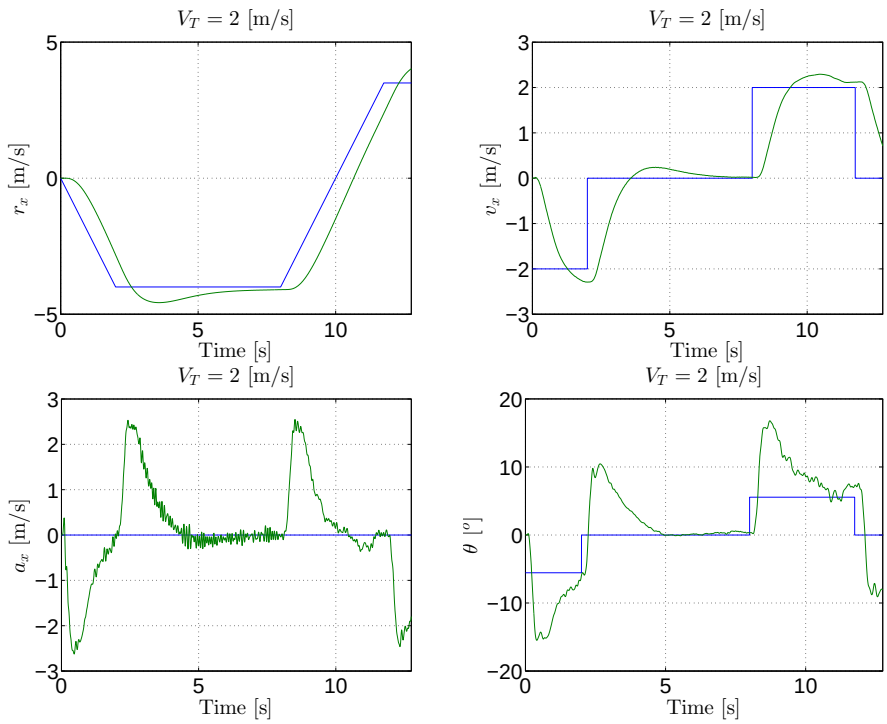


Figure 5-1: Position, velocity, acceleration and pitch angle response with $V_T = 2$ [m/s].

- The position request is fulfilled with a constant delay and some overshoot, but with null steady-state error;
- The velocity request is fulfilled also with some delay but with worst performance relatively to the steady-state error;
- The acceleration and the pitch angle appear to compensate for each other (apart from a

scale factor) when there is a discontinuity in the velocity, but later the acceleration goes to zero while the pitch angle goes to a constant value.

These conclusions are in line with the previous predictions, mainly the constant pitch angle when there is no acceleration, due to the compensation of the drag forces. The results of the LS fitting can be seen in Figure 5-2, with the identified coefficients

$$\hat{a}_{bla} = -0.0476 \left[\frac{\mathrm{s}}{\mathrm{m}}\right], \qquad \hat{k}_{par} = -0.0036 \left[\frac{\mathrm{kg}}{\mathrm{m}}\right].$$

The results indicate that in fact there is a linear term causing drag due to the velocity. However, the influence of the identified second order drag is too small when compared to the linear term, thus validating the assumption that second order drag can be neglected when flying at relatively slow speeds below 10 [m/s]. From the evaluation of the drag coefficients one can derive the necessary pitch angle to compensate for the drag in steady flight, which can be seen in Figure 5-2.

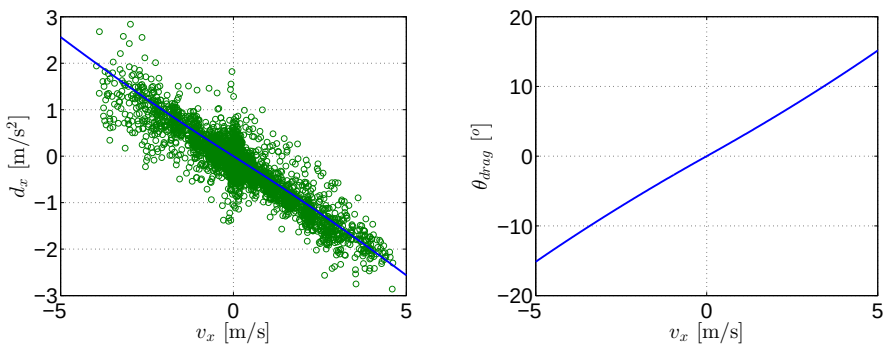


Figure 5-2: LS fitting to obtain the drag coefficients (left) and necessary pitch angle to compensate for the drag force (right).

5-2 Influence of the Lumped Drag Coefficient in the Controller

To see the influence of the lumped linear drag coefficient previously identified in the controllers, trajectories of the same type as in the the previous Section 5-1-2 were tested. Two controllers were used, that differ only in Equation (4-15). The manipulated variable was thus a_{bla}, with $a_{bla} = -0.0476 \ \vee \ a_{bla} = 0$ [s/m]. The second order parasitic drag coefficient was neglected for both.

5-2-1 Results - Influence of the Lumped Drag Coefficient

Figure 5-3 shows that the velocity response is faster when the drag term is accounted. For $V_T = 4$ [m/s] the quadrotor is not able to get to the goal due to space limitations, but still the response is faster. It is noted that the velocity is higher than the desirable, but that fact is explained by the position error.

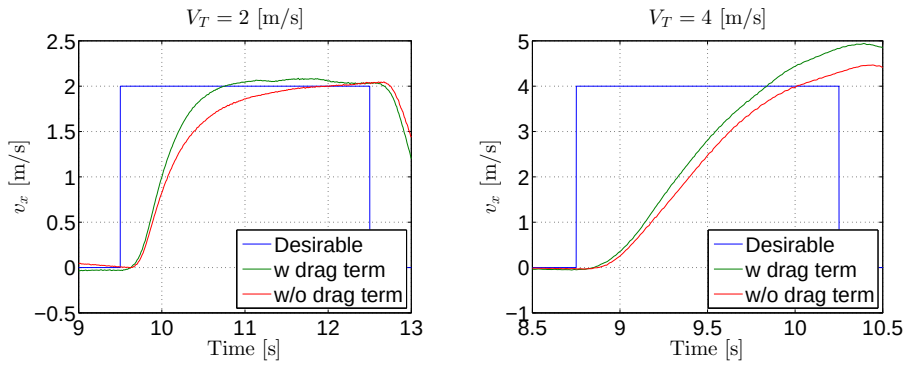


Figure 5-3: Velocity for $V_T = 2 \vee V_T = 4$ [m/s] (left, right).

Figure 5-4 shows the position error, which is smaller when the drag term is accounted, by almost 0.5 meters. Concluding, these Figures show that for small velocities (smaller than 10 [m/s]) the linear drag is relevant and should be accounted in order to increase the performance of the overall controller, thus sustaining the hypothesis that the drag terms should have a significant influence in the controller design.

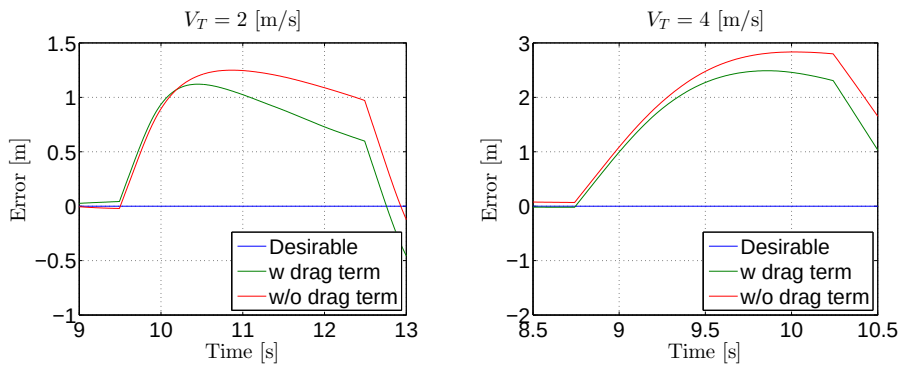


Figure 5-4: Position error for $V_T = 2 \vee V_T = 4$ [m/s] (left, right).

5-3 Optimal Sequencing - TSP Solution

This Section aims at comparing the performances of the alternatives in order to solve the TSP (see Section 3-1-2). The methodology will be described in Section 5-3-1 and the results will be shown in section 5-3-2.

5-3-1 Methodology

Unless stated otherwise, the experiments were set to randomly generate waypoints in a $50 \times 50 \times 50$ [m^3] box, while randomizing the wind velocity as fractions of the limit velocity, with a random angle $\alpha_1 \in [0, 360]$ [^{o}]. The value of K_{heu} is set to $K_{heu} = 0.9$, as it will be discussed later when optimizing K_{heu}. The experiments were performed in a 64 bits Windows 7 OS, running MATLAB 2013a, with an Intel i5-2430M 2.40 GHz CPU processor and 4.00 GB of memory.

5-3-2 Results - Optimal Sequencing

Firstly, the approaches are compared in terms of computational times (CPU) and problem dimension, resulting that the heuristic h_2 performs the best. However, this heuristic is not admissible, so an analysis in the difference between the heuristic h_2 solution and the optimal solution is accounted. Finally, the optimal value of K_{heu} is determined by means of a sensitivity analysis.

Computational Times: The computational times can be seen in Table 5-1. One can see that for small m the permutations approach works well, even when comparing with the second heuristic h_2. Nevertheless, after $m = 5$ the second heuristic h_2 performs much better. Moreover, an increase in performance is achieved from uniform search, passing to heuristic h_1 and up until the heuristic h_2. Only with the second heuristic method it is reasonable to generate trajectories with large m, in particular greater than $m = 8$.

Table 5-1: CPU times for the different approaches.

		CPU Time [ms]			
		Per.	Uni.	h_1	h_2
	1	0.175	0.306	0.330	0.448
	2	0.295	0.729	0.763	1.112
	3	0.501	1.853	1.879	1.971
	4	1.339	7.097	5.413	4.022
m	5	6.061	34.95	25.62	7.197
	6	42.26	262.0	136.8	11.89
	7	413.7	5101	1816	19.36
	8	3159	-	20282	19.84
	9	-	-	-	20.58

Problem Dimension: The computational times are dependent on the computer in which the simulations are performed and code efficiency. Although the first aspect may be irrelevant, since all simulations were done in the same computer, the second is not. The code for all three search algorithms is similar but it is structured very differently than the permutations method, which uses MATLAB predefined functions. For these reasons, the computational times may not be a fair

comparison criteria. With that in mind, another criteria is the problem dimension, a measure of its complexity. For the permutations method, this is measured as all the possible permutations, and for the search methods this is measured as the nodes in the open list. The results can be seen in Table 5-2, which clearly shows the incremental increasing in performance from the permutations method until the heuristic h_1 method, and a big increase in performance towards the heuristic h_2.

Table 5-2: Problem dimension for the different approaches ($k_{h_1} < k_u < 1$).

		Problem Dimension [-]			
		Per.	Uni.	h_1	h_2
	1	1	1	1	1
	2	2	2	2	2
	3	6	6	6	5
	4	24	23	22	12
m	5	120	108	92	25
	6	720	536	411	44
	7	5040	2847	1963	61
	8	40320	-	8375	70
	9	-	-	-	75
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	m	$\mathcal{O}(m!)$	$k_u\mathcal{O}(m!)$	$k_{h_1}\mathcal{O}(m!)$	$\mathcal{O}(m^2)$

Trajectory Optimality: The second heuristic h_2 is not admissible and thus the resulting solution is not always optimal. The average error in determining the trajectory time is displayed in Table 5-3, when comparing with the optimal solution. Values for m larger than $m = 8$ are not available since the optimal solution is obtained using one of the first three methods. It is possible to see that the error increases as m increases.

Although it seems like the error is uncontrollably increasing, this is not the case, since the above experiments where taken for a random distribution of waypoints. When the waypoints are well distributed, which is the case for most applications, this heuristic gives better results, in particularly giving the correct solution, as it will be seen later in Section 5-4.

Table 5-3: Trajectory time error for the heuristic h_2.

		T [s]		Error [%]
		Optimal	Heuristic h_2	
	1	7.63	7.71	1.09
	2	15.54	15.90	2.30
	3	22.21	23.30	4.92
	4	28.23	30.44	7.82
m	5	33.27	37.17	11.74
	6	37.99	43.44	14.35
	7	42.88	49.00	14.26
	8	46.97	54.71	16.47

Optimal K_{heu}: To determine the optimal value of K_{heu}, which represents a trade-off between the optimal solution and the computational time, multiple values of K_{heu} were studied in a sensitivity analysis. The comparison is being done with an average wind speed V_w of half the limit velocity V_{lim}, which is already a considerable amount of wind, and the results can be seen in Figure 5-5. For each K_{heu} the CPU times and trajectory times are determined for up until $m = 8$ waypoints, which reflects the increase in the trajectory time. When K_{heu} decreases the computational time increases but the average trajectory time decreases. A good trade-off between trajectory optimality and CPU time is achieved for $K_{heu} = 0.90$.

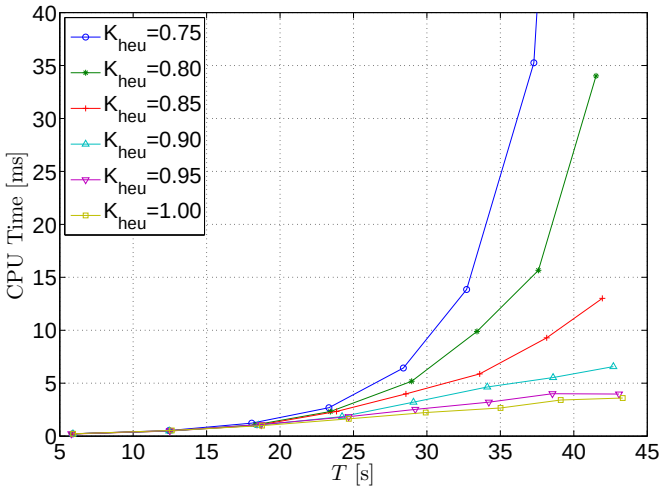


Figure 5-5: Influence of K_{heu} in the trajectory time versus the computational (CPU) time.

5-4 Time Optimal Trajectories

This Section aims at showing the smooth time optimal planned trajectories after the three steps of the proposed approach are performed. It will be seen results for the wind influence when it is symmetrically and asymmetrically distributed around the origin.

Wind on a Symmetric distribution of Waypoints: Symmetrical wind is tested using $V_w = V_{lim}/3$, $V_{lim} = 3$ [m/s], and for four symmetric wind directions, only in the horizontal plane, with $\alpha_1 \in \{20,\ 110,\ 200,\ 290\}$ [^{o}] (shifted 90 [^{o}] in each quadrant). The results are shown in Figure 5-6. The time intervals are obtained by solving the TSP and are then optimized as previously referred. Due to the adequate distribution of waypoints around the origin, the solution of the TSP either using an optimal method or using the heuristic h_2 are the same, which sustain the hypothesis given in Section 5-3.

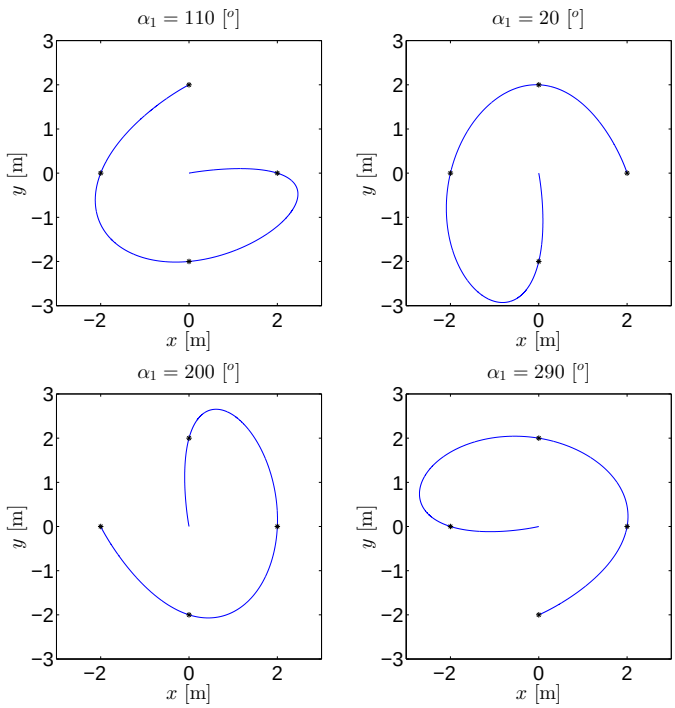


Figure 5-6: Optimal trajectory for a symmetric distribution of $m = 4$ waypoints.

The resulting trajectories are also symmetric and one can confirm the critical importance of wind when choosing the optimal sequence of waypoints. This is a result from the TSP and is illustrated here. The trajectories appear to be smooth and logical. The resulting velocity with respect to the airstream, acceleration and jerk are shown in Figure 5-7 . It is validated that the maximum $V_\infty = 3$ [m/s], while the average velocity $V_{ave} = 2.1$ [m/s]. The total trajectory time is $T = 6.4$ [s], for every wind direction. The jerk is minimal by definition leading to accelerations of 4 [m/s^2] which are clearly acceptable and within limits used, for instance, by Mueller et al. (2013, 2015); Hehn and D'Andrea (2015).

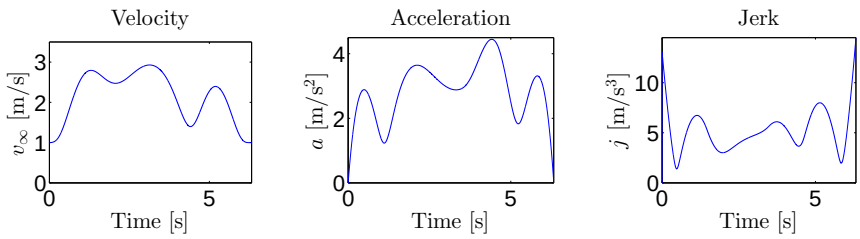


Figure 5-7: Optimal velocity, acceleration and jerk.

Wind on an Asymmetric distribution of Waypoints: When the waypoints are not symmetrically distributed around the origin, the trajectories for different wind directions are not symmetrical. The results are shown in Figure 5-8 and Table 5-4. Once again the trajectories

appear to be logical for passing through the set of waypoints, and the sequences appear to be natural as well. This qualitatively validates the use of the heuristic h_2 for selecting the waypoints sequence, because since $m = 15$ is relatively big, the optimal sequence could not be computed using the other three methods.

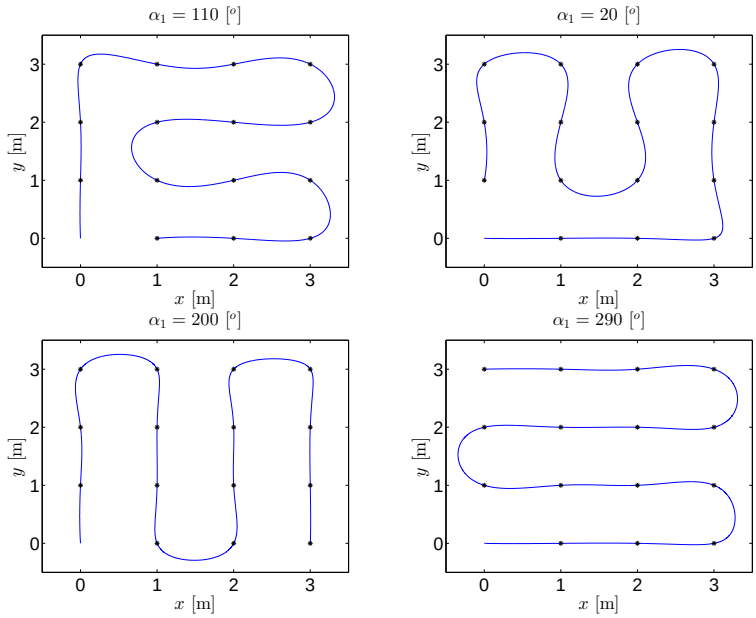


Figure 5-8: Optimal trajectory for an asymmetric distribution of $m = 15$ wayoints.

Due to the restriction on the maximum velocity, the trajectory times are different. In particular, the first two angles will create usually back wind, which allows faster trajectories, and the last two angles will create usually front wind, which allows slower trajectories. Thus, in environments where wind exists and it is known, the selection of the sequence of waypoints is critical to reduce the trajectory time and obtain the time optimal trajectory.

Table 5-4: Trajectory times for an asymmetric distribution of waypoints.

α_1 [^{o}]	20	110	200	290
T [s]	8.2466	8.2160	9.5278	9.0250

5-5 Trajectory Planning Overview

It was proposed to divide the trajectory planning into three smaller problems, or steps, in order to reduce the complexity associated with finding the optimal sequence. Therefore, an initial simplification was used leading to the following question: "Is the initial obtained sequence the optimal one, after all steps?". It was checked that the sequence solution of the TSP is usually the best after all steps, but in some cases gives a trajectory time that is 5-10 [%] over the real optimal.

Nevertheless, the CPU times when solving he TSP problem are negligible when compared to the other two next steps. There is a difference of milliseconds to seconds, three orders of magnitude. Therefore, by simplifying into three smaller steps, the overall CPU time is reduced by $\mathcal{O}(m!)$ and the purpose of the simplification is still satisfied, since the trade-off between CPU time and trajectory optimality is positive.

5-6 Reference Generator Comparison

This Section aims at comparing the performance of a standard trajectory generator, that uses a second order low-pass filter, versus the proposed optimal reference generator. Therefore, the difference in the two controllers is only in the first stage, the reference generator. The trajectory selected consists of $m = 8$ equally distributed waypoints and the limit velocity is set to $V_{lim} = 1$ [m/s]. The quadrotor starts at $(x, y) = (-2, 2)$ [m].

5-6-1 Results - Reference Generator

The optimal trajectory, seen in Figure 5-9, is smooth and similar in shape to what has been seen in the previous Sections, whereas the standard trajectory consists of straight lines with some overshoot after the waypoints.

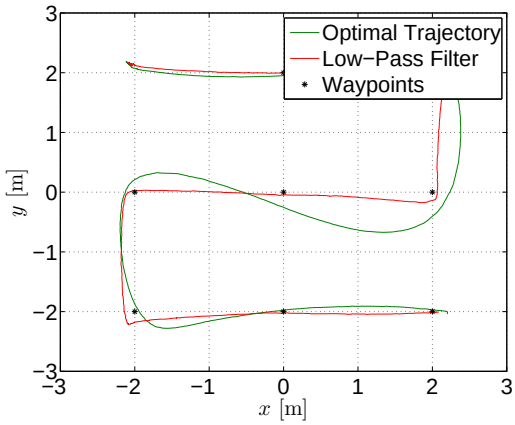


Figure 5-9: Comparison of the 2D trajectory.

Velocity magnitude: In Figure 5-10 it is shown the time evolution of the velocity magnitude. In both approaches, the total trajectory time is approximately the same, i.e., $T = 27$ [s]. One can see the velocity constraint $V_{lim} = 1$ [m/s] to be respected for almost all time instants. The standard reference generator makes the velocity behavior similar between each waypoint: there is an initial increase in velocity until steady state is reached; then the quadrotor senses that is near the next

waypoint and the reference is changed; due to the low-pass filter the velocity reduces; the process repeats. Thus the velocity profile is in a sawtooth wave fashion. For the optimal trajectory the velocity profile behavior is much smoother and natural, in particular with less variations.

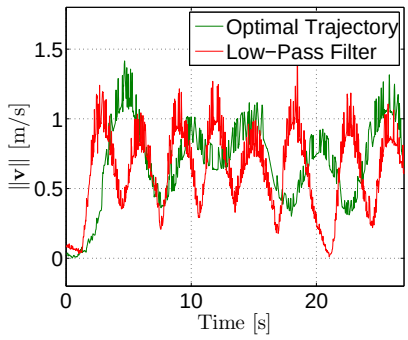


Figure 5-10: Caparison of the velocities magnitudes.

Body velocity: Since the yaw angle ψ_T is constant in the standard case, depending on the turn direction, the body velocity behavior will be separated in both axes. This behavior can be seen in Figure 5-11, where one can see a big velocity fluctuation for both axes. For the optimal case, the yaw angle ψ_T follows the desirable velocity direction and thus the velocity only varies in the x body axis. For the y body axis the velocity is near zero. Thus, the velocity variations are reduced in half and only one axis direction is used, increasing the stability.

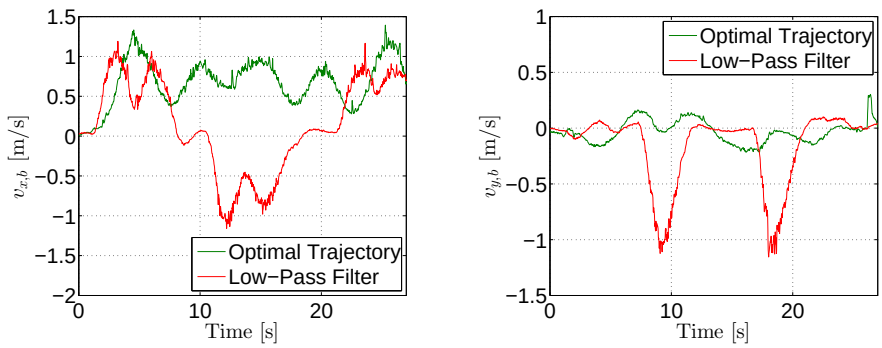


Figure 5-11: Comparison of the velocity in x (left) and y (right) body axes directions.

Euler angles: The velocities contrasts have influence evidently on the accelerations and thus on the Euler angles, which can be seen in Figure 5-12. The Figure shows a clear difference in the results of both references generators, since the angles are bigger and more aggressive for the standard low-pass filter case, whereas in the optimal case the angles are smoother. For the standard reference generator $\phi \in [-3, 6]$ [^{o}] and $\theta \in [-6, 6]$ [^{o}], whereas in the optimal generator $\phi \in [-2, 2]$ [^{o}] and $\theta \in [-2, 4]$ [^{o}]. Therefore, a save in angles variation of 50 [%] is achieved when the optimal trajectory generator is used.

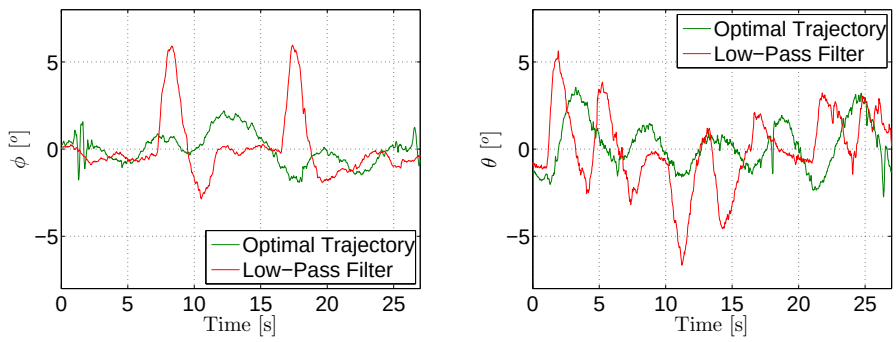


Figure 5-12: Comparison of the Euler angles (ϕ left and θ right).

Thrust: Eventually the summed effects of the velocities, accelerations and angles have influence in the energetic performance of the quadrotor. As discussed in Section 3-2-4, it was chosen to minimize jerk since it is a better representative of the aggressiveness of the system inputs, the rotors angular velocities, and thus it is linked to thrust from Equation 4-23. Figure 5-13 compares the thrust for both reference generators and shows that in fact there is less energy consumption when an optimal trajectory is chosen. The hover thrust for the working quadrotor is $T_h = 3.874$ [N], while the optimal mean thrust is $T_{opt} = 3.886$ [N] and the mean thrust resulting from the low-pass filter is $T_{lp} = 3.923$ [N]. Thus, thrust is reduced from T_h plus 1.3 [%] to T_h plus 0.3 [%] when the optimal reference generator is chosen, meaning a reduction in thrust variation of 77 [%]. This Figure also shows that the thrust during a flight is approximately equal to the hovering thrust, within 5 [%] of thrust variation, validating the hypothesis that almost all thrust goes to compensate the gravity force.

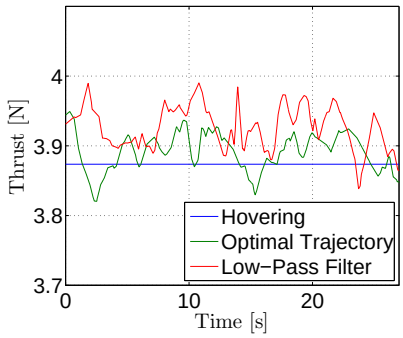


Figure 5-13: Comparison of the Thrust.

5-7 Thrust Vectoring versus PID Control

This Section aims at comparing the performance of a classic PID controller versus the proposed thrust vectoring controller. The tested trajectory consists of $m = 3$ equally distributed waypoints. Two different limit velocities were tested, and throughout all this Section Figures 5-14-5-18 show results for the limit velocity $V_{lim} = 1$ [m/s] on the left and for $V_{lim} = 2$ [m/s] on the right.

5-7-1 Results - Controllers Comparison

Figure 5-14 shows the overall trajectory. One can clearly see the increase in performance on the thrust vectoring controller. Furthermore, the trajectory performance of the proposed controller is not affected by the increase in the velocity, which widens the flight envelope.

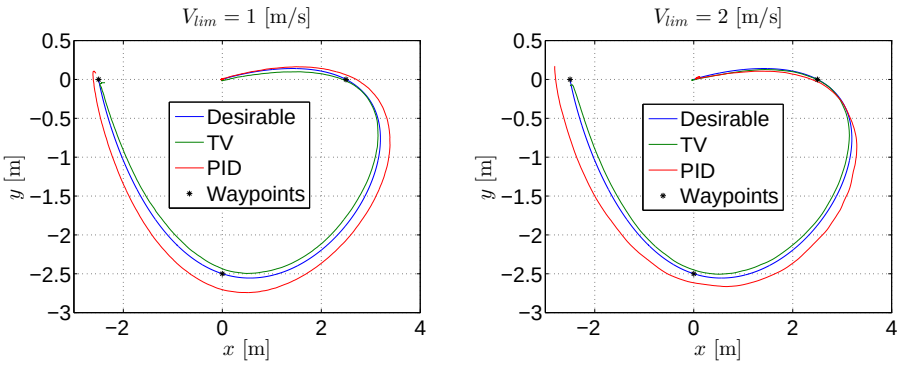


Figure 5-14: Comparison of the 2D trajectory.

Horizontal Position: For the PID controller it seems that the performance increases when the velocity increases, but this is a false conclusion caused by the Figure type. Seeing Figure 5-15 one can conclude that in fact the performance of the PID controller worsens, whereas in the proposed controller the performance is relatively the same (note the different time scales), and a reduction of one second time delay is achieved. The delay caused in the PID controller is mainly due to the integrator term. The integrator role is to compensate the steady state error mainly caused by the drag term, but since the trajectory is aggressive, as it is always changing, the integrator can not perform fast enough. The results are similar for the y axis.

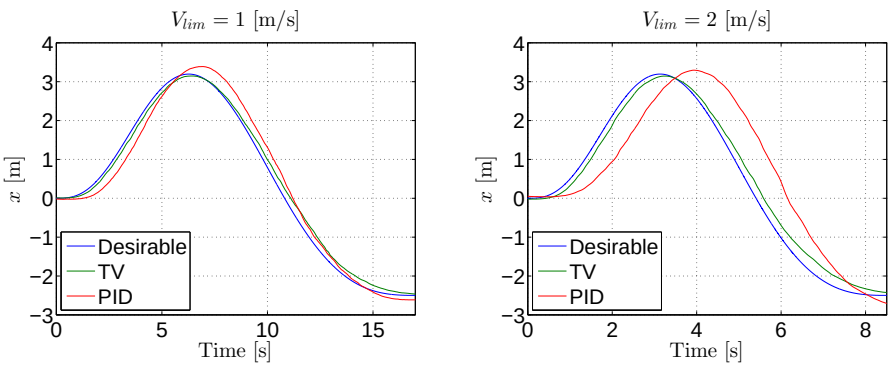


Figure 5-15: Comparison of the x trajectory.

Vertical Position: Due to separation in the control loops it is expected that the performance in the vertical position is poorer for the PID controller, as it can be seen in Figure 5-16, since the position has more error and variation in the PID controller. For the thrust vectoring case, there is an initial oscillatory behavior but then the following becomes evident, whereas in the PID

controller the tracking never truly happens. Note the different scale when comparing with Figure 5-15. This is a result of the vertical and horizontal loops inclusion into one single stage.

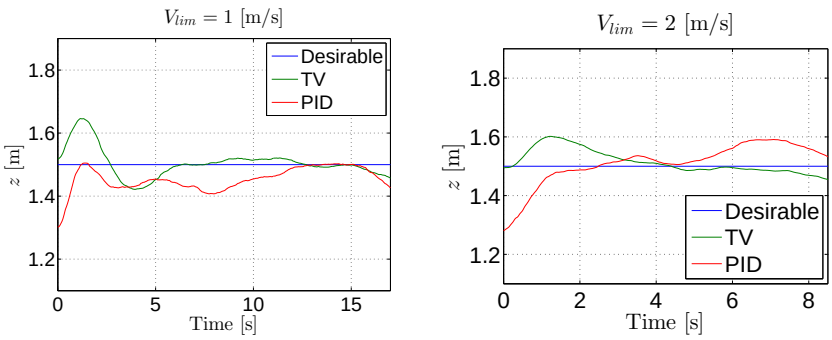


Figure 5-16: Comparison of the z trajectory.

Euler Angles: The final results (Figures 5-17 and 5-18) show the performance in terms of the Euler angles. With the differential flatness outputs and derivatives, and assuming a perfect following of the trajectory, the desirable pitch and roll angles can be computed, including the influence of the drag effects or not. One can see that for the thrust vectoring case the performance is better when comparing with the PID case. The performance increases drastically when the velocity also increases, proving the robustness of our controller, which is able to follow trajectories in a wider flight envelope. It can be seen that by including the drag into the thrust vectoring controller the initial response is much faster (see Figure 5-18). Furthermore, one can qualitatively validate the identification of the drag term, since the angles appear to follow the dark blue line, that accounts for the drag effects, better than the soft blue line, that does not.

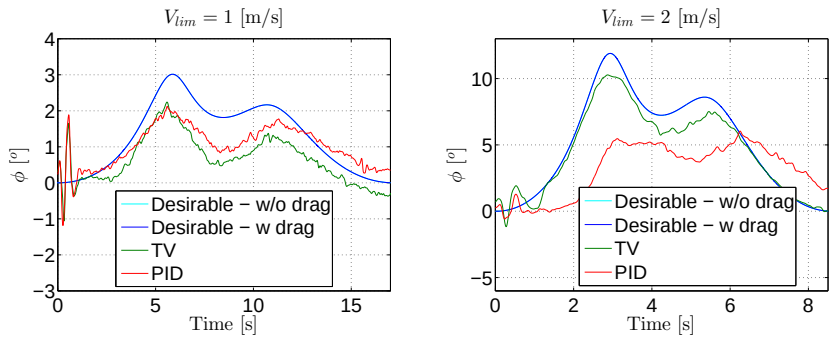


Figure 5-17: Comparison of the roll angle ϕ.

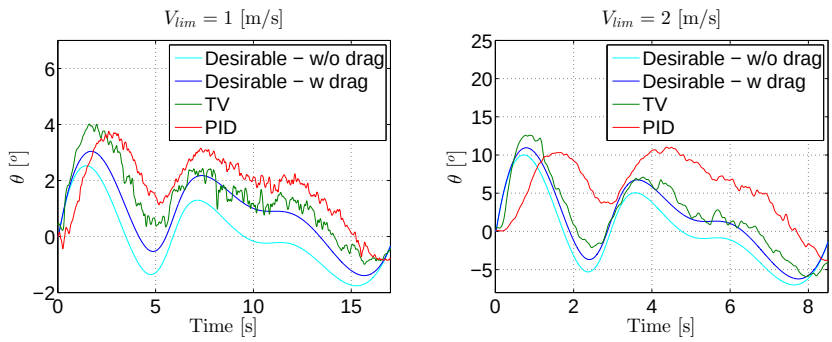


Figure 5-18: Comparison of the pitch angle θ.

Chapter 6

Conclusion

The aim of this work resided on three topics, namely: studying the aerodynamic effects present in the dynamics of a quadrotor; planning the time optimal trajectory in the presence of wind and; designing a controller that uses wind and trajectory information.

A literature study was performed and the formulas for the most important aerodynamic effects were derived. With an experiment, the lumped and parasitic drag coefficients were identified, resulting in $\hat{a}_{bla} = -0.0476$ [s/m] and $\hat{k}_{par} = -0.0036$ [kg/m], and it was shown that even in environments where wind is absent, the inclusion of the drag terms in the controller increases significantly the overall performance. Therefore, the identification of the drag terms should be a step considered in controllers design stage.

Considering the time optimal trajectory, a simplified method to determine the solution was proposed, based on three steps. It was proven that the steps division saves computational time, reducing it approximately by an $\mathcal{O}(m!)$ order, while still maintaining acceptable time optimality. The QP was constructed and solved, showing results that indicate acceptable values for acceleration and jerk when compared to related work. The reference generator was compared to a standard one, a low-pass filter, showing a performance increase. The profit comes from saving energy related with thrust variation, reduced by 77 [%], and from reducing the aggressiveness related with angle variations, reduced by half.

Finally, a thrust vectoring controller was proposed and augmented to include the drag terms. The controller was compared with a current available open-source PID controller and it was proven that a clear increase in performance of trajectory following is achieved. A reduction of time delay

was obtained, and the position following was improved both in horizontal and vertical planes. Moreover, it was observed that the thrust vectoring controller maintains the performance in a higher flight envelope.

6-1 Future Work

The following future work is proposed, which covers the three main aspects of this work. For the aerodynamic effects, it is advised to test the quadrotor at higher velocities to obtain a relevant parasitic drag coefficient, as well as the identification of the thrust variation coefficients, extending the model to include them. For the optimal sequencing, it is recommend a study to explore the less optimal results of the TSP. In particular by verifying the increase in time optimality versus the increase in CPU time to be achieved with them, after the QP problem when assuring the constraint on the maximum velocity. Finally, it is proposed to test the performance of the thrust vectoring controller with higher aggressive maneuvers, higher velocities, and with 3D movement. This will allow to see the real extension of the possible flight envelope.

Bibliography

Allibert, G., Abeywardena, D., Bangura, M., & Mahony, R. (2014, October). Estimating body-fixed frame velocity and attitude from inertial measurements for a quadrotor vehicle. In *Ieee conference on control applications (cca)* (p. 978-983).

Araar, O., & Aouf, N. (2014, July). Quadrotor control for trajectory tracking in presence of wind disturbances. In *Ukacc international conference on control.*

Augugliaro, F., Schoellig, A. P., & D'Andrea, R. (2012, October). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *Ieee/rsj international conference on intelligent robots and systems* (p. 1917-1922).

Bangura, M., & Mahony, R. (2012, December). Nonlinear dynamic modeling for high performance control of a quadrotor. In *Australasian conference on robotics and automation.*

Barrientos, A., Gutierrez, P., & Colorado, J. (2009, January). Advanced uav trajectory generation planning and guidance. In (chap. 4). InTech.

Bipin, K., Duggal, V., & Krishna, K. M. (2014, December). Autonomous navigation of generic quadrocopter with minimum time trajectory planning and control. In *Ieee international conference on vehicular electronics and safety (icves)* (p. 66-71).

Bouktir, Y., Haddad, M., & Chettibi, T. (2008, June). Trajectory planning for a quadrotor helicopter. In *16th ieee mediterranean conference on control and automation* (p. 1258-1263).

Bryson, A. E., & Ho, Y.-C. (1975). *Applied optimal control: Optimization, estimation, and control.* Taylor and Francis.

Corbets, J. B., & Langelaan, J. W. (2007, August). Parameterized optimal trajectory generation for target localization. In *Guidance, navigation and control conference.*

Flash, T., & Hogan, N. (1985, July). The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, 1688-1703.

Fliess, M., Lévine, J., Martin, P., & Rouchon, P. (1992, January). On differential flat nonlinear systems. *Comptes Rendus de L'Académie des Sciences*.

Hehn, M., & D'Andrea, R. (2015, August). Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*.

Hoffmann, G. M., Huang, H., Waslander, S. L., & Tomlin, C. J. (2007, August). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Aiaa guidance, navigation and control conference and exhibit.*

Huang, H., Hoffmann, G. M., Waslander, S. L., & Tomlin, C. J. (2009, May). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Ieee international conference on robotics and automation* (p. 3277-3282).

J.A. Guerrero, Y. B., J.A. Escareno. (2013, May). Quad-rotor mav trajectory planning in wind fields. In *Ieee international conference on robotics and automation (icra)* (p. 778-783).

Kawato, M., Maeda, Y., Uno, Y., & Suzuki, R. (1990). Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion. *Biological Cybernetics*, 275-288.

Lai, L.-C., Yang, C.-C., & Wu, C.-J. (2006). Time-optimal control of a hovering quad-rotor helicopter. *Journal of Intelligent and Robotic Systems (2006)*, 115-135.

Laporte, G. (1991, July). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 231-247.

Leishman, G. J. (2006). *Principles of helicopter aerodynamics* (second ed.). Cambridge University Press.

Lorenz, S., & Adolf, F. M. (2010). A decoupled approach for trajectory generation for an unmanned rotorcraft. In (chap. 1). Springer Berlin Heidelberg.

Mahony, R., Kumar, V., & Corke, P. (2012, September). Multirotor aerial vehicles - modelling, estimation and control of quadrotor. *IEEE Robotics & Automation Magazine*, 20-32.

Martin, P., & Salaun, E. (2010, May). The true role of accelerometer feedback in quadrotor control. *IEEE International Conference on Robotics and Automation*, 1623-1629. (hal-00422423v1)

Mellinger, D., & Kumar, V. (2011, May). Minimum snap trajectory generation and control for quadrotors. In *Ieee international conference on robotics and automation* (p. 2520-2525).

Mellinger, D., Kushleyev, A., & Kumar, V. (2012, May). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Ieee international conference on robotics and automation* (p. 477-483).

Mellinger, D. W. (2012). *Trajectory generation and control for quadrotors.* Unpublished doctoral dissertation, University of Pennsylvania.

Mueller, M. W., Hehn, M., & D'Andrea, R. (2013, November). A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. In *Ieee/rsj international conference on intelligent robots and systems (iros)* (p. 3480-3486).

Mueller, M. W., Hehn, M., & D'Andrea, R. (2015, December). A computationally efficient motion primitive for quadrocopter trajectory generation. *Transactions on Robotics*, 1294-1310.

Nieuwstadt, M. J. van, & Murray, R. M. (1997, May). Real time trajectory generation for differential flat systems. *International Journal of Robust and Nonlinear Control*.

Omari, S., Hua, M.-D., Ducard, G., & Hamel, T. (2013, November). Nonlinear control of vtol uavs incorporing flapping dynamics. In *Ieee/rsj international conference on intelligent robots and systems (iros)* (p. 2419-2425).

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity.* Dover.

Pontryagin, L. S. (1987). *Mathematical theory of optimal processes.* CRC Press.

Prouty, R. W. (2002). *Helicopter performance, stability, and control.* Krieger Publishing Company.

Rego, C., Gamboa, D., Glover, F., & Osterman, C. (2011, June). Traveling salesman problem heuristics: leading methods, implementations and latest advances. *European Journal of Operational Research*.

Richter, C., Bry, A., & Roy, N. (2013). Polynomial trajectory planning for quadrotor flight. In *International symposium of robotics research (isrr).*

Ryll, M., Bülthoff, H. H., & Giordano, P. R. (2015, March). A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation. *IEEE Transactions on Control Systems Technology*, 540-556.

Silva, E. L. S. da. (2015). *Incremental nonlinear dynamic inversion for quadrotor control.* Unpublished master's thesis, Deflt University of Technology.

Smeur, E. J., Chu, Q., & Croon, G. C. de. (2016, January). Adaptive incremental nonlinear dynamic inversion for attitude control of micro aerial vehicles. In *Aiaa guidance, navigation, and control conference.*

Sydney, N., Smyth, B., & Paley, D. A. (2013, December). Dynamic control of autonomous quadrotor flight in an estimated wind field. In *52nd ieee conference on decision and control.*

Waslander, S., & Wang, C. (2009, April). Wind disturbance estimation and rejection for quadrotor position control. In *Aiaa infotech@aerospace conference.*

Zhou, D., & Schwager, M. (2014, June). Vector field following for quadrotors using differential flatness. In *Ieee international conference on robotics & automation (icra)* (p. 6569-6572).

Appendix A

Parrot Bebop, Paparazzi and CyberZoo

In this Appendix the three most important tools that allowed the fulfillment of the proposed work are described. The first tool is the main hardware, i.e., the quadrotor used in the experiments. The second tool is the main software framework, an open-source software platform for UAVs, used to override the two control outer-loops by including the time optimal reference generator and the thrust vectoring controller. The final tool is the physical arena where the experiments were performed, called CyberZoo, that is constructed at TUDelft.

A-1 Parrot Bebop

The Parrot Bebop (see Figure A-1) used in this dissertation is a commercially available quadrotor[1] widely used at TUDelft as a research platform. It uses a "×" motor configuration and has multiple advantages, such as: resistant structure, also protected by side bumpers to increase safety; lightweight; powerful dual-core CPU; and wide variety of sensors.

[1] `http://www.parrot.com/products/bebop-drone/`

Figure A-1: Parrot Bebop with side bumpers.

Important specifications of the Bebop are[2]:

- Structure - Glass fiber reinforced (15 [%]) ABS structure; 4 Brushless Outrunner motors; Three-blade auto-block propellers in Polycarbonate; 395 [g] with the side bumpers.
- Battery - Lithium Polymer 1200 [mAh]; Flight time of approximately 12 minutes.
- Processor and OS/Software - Parrot P7 dual-core CPU Cortex A9 processor; Quad core GPU; 8Gb flash memory; Linux operating system (kernel 3.4.11 3 SMP PREEMPT); MIMO dual-band WiFi antennas.
- Sensors - 3 axes magnetometer (AKM8963); 3 axes gyroscope (MPU6050); 3 axes accelerometer (MPU6050); Optical-flow sensor; Ultrasound sensor; Pressure sensor.
- Connectivity - MIMO dual-band Wi-Fi antennas with 2 double-set of dipole antennas for 2.4 and 5 [GHz]; Sending power up to 26 [dBm].

A-2 Paparazzi UAV

In the Paparazzi UAV website, it is written that *"Paparazzi UAV (Unmanned Aerial Vehicle) is an open-source drone hardware and software project encompassing autopilot systems and ground station software for multirotors, fixed-wing, helicopters and hybrid aircraft that was founded in* 2003*. Paparazzi UAV was designed with autonomous flight as the primary focus and manual flying as the secondary. From the beginning it was designed with portability in mind and the ability to control multiple aircraft within the same system. Paparazzi features a dynamic flight plan system that is defined by mission states and using way points as "variables". This makes it easy to create very complex fully automated missions without the operators intervention."*[3]

Figure A-2: Paparazzi UAV logo.

[2] http://wiki.paparazziuav.org/wiki/Bebop
[3] http://wiki.paparazziuav.org/wiki/Main_Page

The mentioned reasons suggest that the Paparazzi UAV platform is a suitable framework for this work. Other important features that confirm this are: there is a strong community currently developing Paparazzi, with multiple members of that community working at TUDelft; there is code developed with focus on the Parrot Bebop quadrotor; Paparazzi is written in *C* and it works by subsystems and modules, which are very flexible, allowing implementation of new code and improvement of old code, for instance to override control loops.

Paparazzi UAV was installed in the same computer in which the experiments were conducted, with an Intel i5-2430M 2.40 GHz CPU processor, and in a Virtual box running Ubuntu 14.04 32-bit operating system with 1.50 GB of dedicated memory.

The modules and subsystems are a key feature in Paparazzi. The most important subsystem are: the telemetry subsystem, which is adaptable to easily add new telemetry variables; the guidance and stabilization subsystems, that contain the control loops; and the auto-pilot subsystem, the main loop that runs at 512 [Hz] and is responsible for the flux of code. The modules are usually written by the individual developers that want to create code for specific applications, as the communication module that will be implemented in this work in Section B-2.

There is also a Ground Control Station (GCS) that allows for a visual interface of the quadrotor in the computer monitor. Through this interface, it is possible to manually fly the quadrotor, with a joystick, and to automatically fly it by means of flight plans. Those flight plans are also flexible and allow the introduction of new functions.

Other important aspect is the interchange of information by datalink communication between the quadrotor and the Paparazzi platform. This communication is implemented in Transmission Control Protocol (TCP) and there are *uplink* and *downlink* channels. The *uplink* is mainly used to control the quadrotor while the *downlink* is mainly used to store the telemetry data and to give information to be displayed in the GCS. The storage of telemetry data is obtained by the server, another key feature of the Paparazzi UAV platform.

A-3 CyberZoo

The CyberZoo is the flight test arena where all the flights were performed and can be seen in Figure A-3. It consists of an open space with 5×5 [m^2] area, and surrounded by a grid to protect both the quadrotor and the developers. The CyberZoo allows that several people work at the same time, allowing for team work support. On the roof of the arena (see Figure A-3(c)) a Motive[4] tracking system developed by OptiTrack is installed, which allows to get the pose of the quadrotors. The

[4] `https://www.optitrack.com/products/motive/tracker`

tracking system consists of 24 precision cameras, which serve to track the rigid body, and fake a Global Navigation Satellite System (GNSS). The system precision is greater than millimeters for position, and greater than tenths of degree for Euler angles.

(a) CyberZoo flight arena. (b) Bebop flying in the arena. (c) Several OptiTrack cameras.

Figure A-3: CyberZoo flight arena.

Appendix B

Code Implementation

In this Appendix the code implementation will be analyzed in detail, in particular the code written in C that serves two main goals.

The first goal is to construct an external program used to generate the time optimal trajectory. An external program is used because the reference generator uses a QP solver program, written in $C++$, as it will be discussed in Section B-1. The external program communicates the desirable trajectory to the Paparazzi main program using a communication module, that will be discussed in Section B-2. In order to safely communicate the trajectory to the quadrotor, a series of finite-state machines were implemented. This finite-state machines will be explained in Section B-3.

The second goal of the code is to receive the communicated information inside Paparazzi, as well as to override the PID control loops by the thrust vectoring controller. To do so, the Paparazzi code was altered and extended, as it will be seen in Section B-4. Finally, to implement the thrust vectoring controller it is necessary to give commands of thrust in Newton units. However, the Paparazzi auto-pilot is built to use integer thrust units ranging from 0 to 9600. Thus, in order to get the units conversion a static thrust mapping test was performed, with the results being presented in Section B-5.

B-1 Reference Generator

The reference generator files are the C code files necessary to create the external (to Paparazzi) program used to communicate to Paparazzi the desirable trajectory. The files can be found in the Folder `Ref_gen_files`. The reasons that support the adoption of an external program will be seen in Section B-2. The reference generator program is the `trajectory_generation` file and it should be executed in the terminal with `./trajectory_generation`. The program uses standard C libraries, as well as the GSL - GNU Scientific Library[1] and it is compiled in the terminal with `make`. The program `main()` function is divided in three parts:

- Problem definitions;
- Trajectory generation;
- Trajectory server.

The problem definitions are written either in the `main()` preamble or in its header file and consist of basic definitions that are discussed in the main Chapters and are necessary in the code implementation. The basic definitions are:

- Number of waypoints `N_POINTS` $= m$;
- Wind velocity `v_w` $= V_w =$ and wind angle `angle_w` $= \alpha_1$;
- Limit velocity `v_lim` $= V_{lim}$;
- Quadratic problem specifications `PROB.K` $= k$, `PROB.N` $= n$, and `PROB.P` $= p$;
- Time optimal trajectory time `T_opt` $= T$;
- Cost function objective vector `x` $= \mathbf{c}$;
- Number of iterations `n_it_qp` and `n_it_cf`;
- Waypoints position `r_T` $= \mathbf{wp}$;
- Others auxiliary variables.

The trajectory generation is composed by three functions that correspond to the three steps of the proposed time optimal trajectory generation approach:

- Optimal Sequencing: the optimal sequence is achieved by solving the TSP with a heuristic search method (heuristic h_2) and uses information of the time intervals using the Zermelo's problem theory. The name of the function is `zermelo_search_h2()`;

[1] `https://www.gnu.org/software/gsl/`

- QP problem: the QP optimization problem is structured and solved using the function `quadratic_programming()`. To solve the QP problem, an external program written in $C++$ is necessary. This approach is discussed in Section B-2;
- Constraints on the limit velocity: to make sure that the velocity with respect to the airstream respects the limit imposed by V_{lim}, the function `check_feasibility()` is necessary.

The trajectory server is implemented to communicate the trajectory to the quadrotor in the function `perform_as_server()`, and it will be explained in detail in Sections B-2 and B-3.

B-2 Communication - QP Programming Implementation

In order to obtain the optimal trajectory solution it is necessary to solve a quadratic minimization problem and, to do so, Optimization Toolboxes already exist. Those optimization toolboxes are usually developed in high-level mathematical programming languages such as Matlab[2], Maple[3] or Mathematica[4], or as simple programs in which the user has to specify each element of $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}_{in}$, $\mathbf{b}_{in}$, $\mathbf{A}_{eq}$ and $\mathbf{b}_{eq}$ manually, usually developed in Java[5], Python[6], Fortran[7] or $C++$[8].

For this work, however, it is desirable to feed into the program only the waypoints position and the time intervals, so that the program can build the referred matrices and vectors by itself and then solve the optimization problem. Ideally this program would be implemented on-board to increase the full autonomy of the quadrotor, but in order to do so, it is necessary to have the program written in C, as Paparazzi is. After a wide search for toolboxes or programs that solve QP problems in C, it was concluded that none exist that satisfies the goals, and a different approach was opted.

The implemented approach uses an external program running on Ubuntu communicating with Paparazzi and the drone at 20 [Hz], via a wireless communication protocol. The programs scheme can be seen in Figure B-1. The main external program is written in C and, knowing the waypoints, is able to generate the matrices and vectors for the QP problem. It then writes that data to a text file in a Mathematical Programming System (MPS) format and calls a toolbox[9] written in $C++$ that reads that same file, produces another file with the solution and signals the first one

[2] http://nl.mathworks.com/help/optim/ug/quadprog.html
[3] http://www.maplesoft.com/support/help/Maple/view.aspx?path=Optimization/QPSolveMatrixForm
[4] http://reference.wolfram.com/language/tutorial/ConstrainedOptimizationGlobalNumerical.html
[5] http://www.joptimizer.com/quadraticProgramming.html
[6] http://cvxopt.org/
[7] http://staffhome.ecm.uwa.edu.au/~00043886/software/quadprog.html
[8] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/
[9] http://doc.cgal.org/latest/QP_solver/

that the solution is available. Afterwards, the main external program reads that file, with the optimal coefficients, and produces the full trajectory in components of $\mathbf{r}_T(t)$, $\dot{\mathbf{r}}_T(t)$, $\ddot{\mathbf{r}}_T(t)$ and $\psi_T(t)$. Finally, the main external program starts a server that waits for messages requests from the client, the Bebop, asking for the time optimal trajectory (see Section B-3).

The communication protocol for this type of applications is usually UDP (User Datagram Protocol), or TCP (Transmission Control Protocol). The first one is faster when transmitting the messages, but since it has no ordering of messages nor tracking of the connections is less robust and the connection with the quadrotor may be lost. Since this is highly undesirable, the second, slower but safer protocol was chosen.

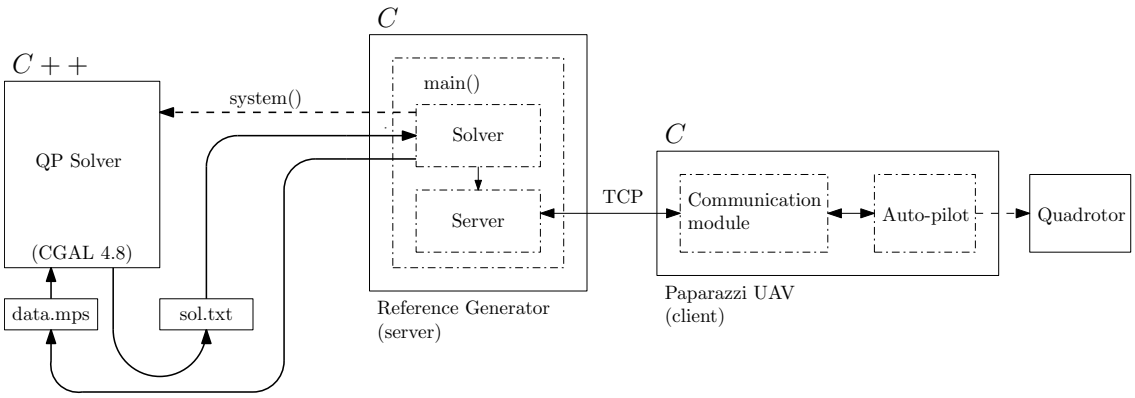


Figure B-1: Program Scheme containing the Solver and Client/Server implementations.

B-3 Finite-State Machines

In order to safely communicate the trajectory to the quadrotor and change between autopilot modes in Paparazzi, a series of finite-state machines were implemented. These finite-state machines correspond to the client (quadrotor), the server (external program) and the autopilot mode selected in the GCS. The three machines can be seen if Figure B-2.

Client

The client is the most important finite-state machine since its actions are responsible for the state changes in itself and the other finite-state machines. Its state flow is controlled by a global variable (global inside Paparazzi), `control`, which indirectly sets the control variable of the server machine. The client has four possible states:

1. `IDLE` - when the client starts, `control=0` and the reference generator is the same as in the usual Paparazzi UAV approach. The requested message to the server is `DO_NOTHING`. If

nothing happens then the state maintains. Every time that, in any other state, the user clicks to cancel, the state redirects to here with the message `DO_NOTHING`;

2. `ALIGN` - when the user clicks to start a time optimal trajectory, then the quadrotor starts aligning with the initial heading of the desired trajectory. It jumps from `IDLE` with `control=1` and then it maintains the state with `control=2`, until the heading is aligned within a certain margin. The requested message to the server is `ALIGN`;

3. `INIT` - when the heading is aligned with the desirable one, the flux goes to the trajectory initialization, by informing that to the server. So `control=3` and the requested message to the server is `INIT`. The client waits with `control=4` until the server reports back;

4. `REQUEST` - when everything is set, `control=5` and the main state is reached. This state loops with `control=6` and flies the optimal trajectory. When the quadrotor reaches the end of the trajectory it maintains its final position. The loop is only broken if the user clicks to cancel. In this state the requested message to the server is `REQUEST`.

Server

The server state-machine also has the four same states as the client. It is fully controlled by the client due to the communication of messages via TCP communication, by the message type, which can be: `B=DO_NOTHING`; `B=ALIGN`; `B=INIT`; and `B=REQUEST`.

Autopilot Mode

The autopilot mode state machine only has two states that correspond to the two autopilot modes. The usual state is `NAV` that uses the standard reference generator embedded in Paparazzi, while the other state is `NAV_OPT` that uses the time optimal reference generator. The control flux is set by the same variable `control` as in the client state-machine, since it is a global variable within Paparazzi. The state on the right is only achieved when `control=6`, i.e., both client and server are in the `REQUEST` state.

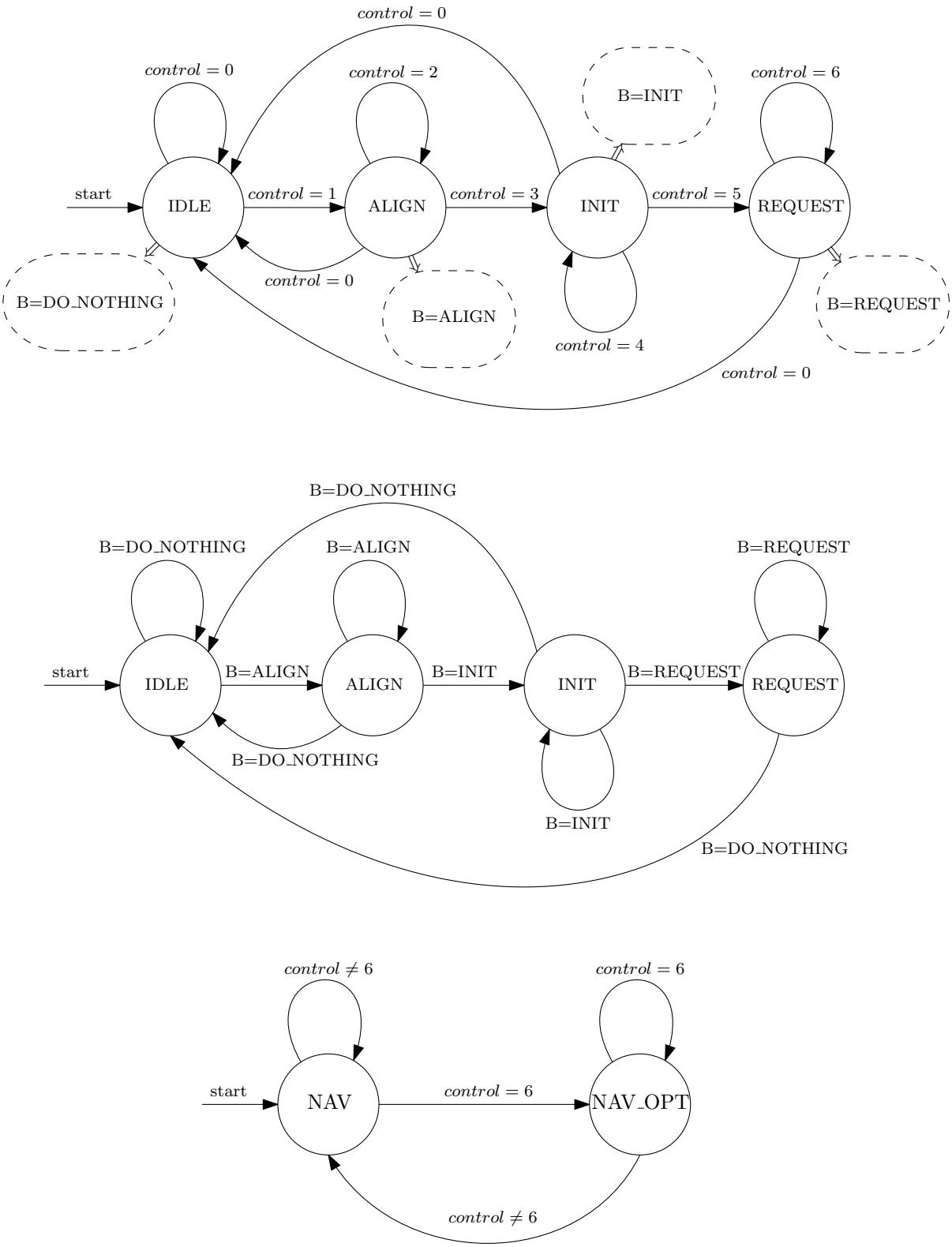


Figure B-2: Finite-State Machines - Top to bottom: Client, Server and Mode.

B-4 Paparazzi Files

The Paparazzi files are the C code files mainly used to override the two outer loops embedded in Paparazzi, which correspond to the reference generator and the thrust vectoring controller. The files can be found in the Folder `Paparazzi_files`. Besides changing the Paparazzi files, further referred as "old files", some new files have been created, named with the prefix `jp`, and further referred as "new files". The prefix or suffix `jp` is also used to add several variables within the old files, to allow a clear distinction between the already existent variables.

The most important files created are related to the trajectory communication module and are `jp_communicate.c,h,xml`. This communication module is the client inside Paparazzi that communicates with the external program (see Figure B-1), the server, that gives the optimal trajectory when requested. Furthermore, this module also allows to change the gains used in the thrust vectoring controller. Those gains are `GAIN_JP_PP_H,V` $= K_p$, `GAIN_JP_PI_H,V` $= K_{pi}$ and `GAIN_JP_V_H,V` $= K_v$, defined with vertical and horizontal precision. The gain K_a was not used since the measurements of the acceleration in the Bebop are overly noisy. Moreover, this module also allows to control the usage, or not, of the thrust vectoring controller with the variable `USE_CONTROLLER_JP`. The gains and the usage of the controller can be changed in real-time when a flight is being tested, through the GCS settings. The other files created are related with telemetry, flight plans or aircraft configuration files.

The most important files altered are related with the outer loops. From these, the main file is the `guidance_h.c`. In this file is implemented the mechanism to override the outer loops and to control the flux of the variable `control`, defined in Section B-3. The new controller equations are written in the function `guidance_jp_run()`, and initialized in `guidance_jp_init()`. The function `jp_guidance_set_stage()` changes `control`, although it is written as `jp_stage`, and the values $1 - 6$ are numbered differently due to the different moments in which the finite-state machine was developed. Since in Paparazzi the vertical and horizontal loops are separated, but in the thrust vectoring controller this is not the case, then the `guidance_v.c` is also altered to discard the vertical thrust commands. Moreover, in Paparazzi the horizontal commands are not only Euler angles, but also horizontal force commands. Therefore, the Euler angles computed with the thrust vectoring controller have to be extended to force commands, and fed to the INDI stabilizer. For that reason, the file `stabilization_indi.c` is also altered. The other files altered are related with variables declaration, telemetry and algebraic functions.

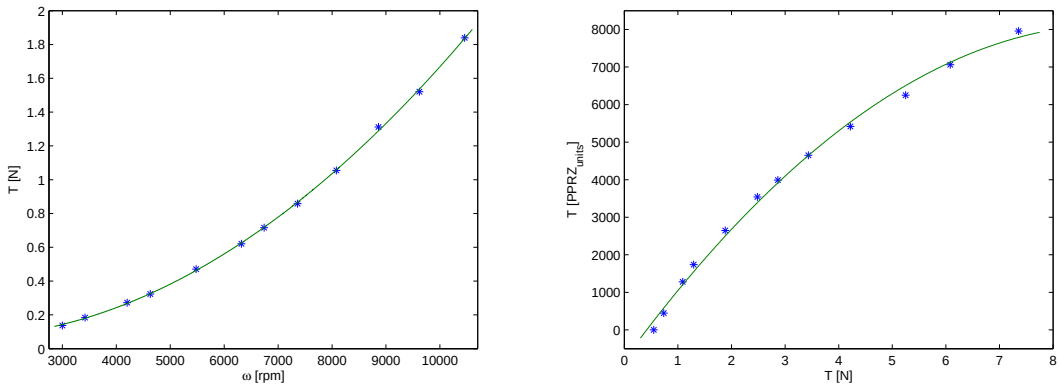
B-5 Thrust Mapping

The commanded thrust within the Paparazzi code is an integer number ranging from 0 to 9600, where 9600 corresponds to the maximum allowed thrust. Thus, in order to correctly compute the desirable thrust in Paparazzi units from the desirable thrust in Newton units, a mapping of units is necessary.

Silva (2015) has performed an experiment which allowed to determine the correspondence between a single propeller rotating speed and its produced thrust for the Parrot Bebop quadrotor. His procedure was based on the measuring of the weight on a weighing scale for multiple requests of propeller rotating speed. The results can be found in Figure B-3(a). The data was fitted to a second order degree polynomial

$$T_N = 8.001 \cdot 10^{-2} - 3.804 \cdot 10^{-5} \cdot \bar{\omega} + 1.969 \cdot 10^{-8} \cdot \bar{\omega}^2 \qquad [\mathrm{N}] \tag{B-1}$$

with $\bar{\omega}$ in [rpm]. The fitting possesses a relative predictive power of $R^2 = 0.9997$.



(a) Propeller Thrust *vs* Propeller Speed.

(b) Thrust in Newtons *vs* Thrust in Paparazzi units.

Figure B-3: Static mapping showing the obtained data (blue) and the fitting data (green).

A discrepancy may be seen since the thrust was expected to be only a quadratic function of the propellers rotating speed. Nevertheless, for nominal rotating speeds, which are around 7000 [rpm] and produce roughly 1 [N] each, the quadratic part accounts already 77 [%] on Equation (B-1). For larger thrusts, the quadratic term influence is even greater.

Knowing that there is a direct relation from the rotating speed to Paparazzi units given by

$$T_{PPRZ} = \frac{\bar{\omega} - 3000}{9000} \cdot 9600 \qquad [\mathrm{PPRZ}_{units}] \tag{B-2}$$

then the mapping from thrust in Newtons to thrust in Paparazzi units can be accomplished. The

results can be seen in Figure B-3(b). The data fitted to a second order degree polynomial results in

$$T_{PPRZ} = -7.936 \cdot 10^2 + 1.948 \cdot 10^3 \cdot T_N - 1.062 \cdot 10^2 \cdot T_N^2 \quad [PPRZ_{units}] \tag{B-3}$$

with a relative predictive power of $R^2 = 0.9963$.

Appendix C

Quadratic Programming Structure

In this Appendix the structure of the Quadratic Programming (QP) problem (see Section 3-2-2) is explained. The derivation will be made with focus on the specifications given in the referred section, in particular the values of k, n and p, but the derivation of the vectors and matrices is made general, so that other values can be chosen for other works.

C-1 Trajectory Definition

Using differential flatness (Fliess et al., 1992; Nieuwstadt & Murray, 1997), and using the quadrotors usual flat outputs (Zhou & Schwager, 2014; D. Mellinger & Kumar, 2011), the target time trajectory is defined as

$$\mathbf{r}_T(t) = [\ r_{T_x}(t) \quad r_{T_y}(t) \quad r_{T_z}(t) \quad \psi_T(t)\ ]^\top \tag{C-1}$$

Considering a single direction on $\mathbf{r}_T(t)$, furthermore referred to as $\sigma_T(t)$, then m trajectories between the origin and each one of the m waypoints in that single direction can be defined as a n order time polynomial, such that

$$\sigma_T(t) = \begin{cases} c_{10} + c_{11}t + c_{12}t^2 + \cdots + c_{1n}t^n & 0 \leq t \leq \Delta T_1 \\ c_{20} + c_{21}t + c_{22}t^2 + \cdots + c_{2n}t^n & \Delta T_1 \leq t\Delta \leq T_2 \\ \vdots & \\ c_{m0} + c_{m1}t + c_{m2}t^2 + \cdots + c_{mn}t^n & \Delta T_{m-1} \leq t \leq T \end{cases} \tag{C-2}$$

with $m(n+1)$ coefficients c_{ij}. In the previous definition, time is monotonically increasing, meaning that the c_{i0} do not represent the m waypoints coordinates. To do so, a time-shift can be applied to Equation (C-2) such that

$$t = \Delta T_{i-1} + t_i \qquad \Delta T_{i-1} \leq t \leq \Delta T_i \tag{C-3}$$

and

$$\sigma_T(t) = \begin{cases} c_{10} + c_{11}t_1 + c_{12}t_1^2 + \cdots + c_{1n}t_1^n & 0 \leq t_1 \leq T_1 \\ c_{20} + c_{21}t_2 + c_{22}t_2^2 + \cdots + c_{2n}t_2^n & 0 \leq t_2 \leq T_2 \\ & \\ \vdots & \\ c_{m0} + c_{m1}t_m + c_{m2}t_m^2 + \cdots + c_{mn}t_m^n & 0 \leq t_m \leq T_m \end{cases} \tag{C-4}$$

Each ith segment of the time trajectory of the position and its derivatives is then given by

$$\begin{aligned} \sigma_{T_i}(t_i) &= c_{i0} + c_{i1}t_i + \cdots + c_{in}t_i^n \\ \dot{\sigma}_{T_i}(t_i) &= c_{i1} + 2c_{i2}t_i + \cdots + nc_{in}t_i^{n-1} \\ \ddot{\sigma}_{T_i}(t_i) &= 2c_{i2} + 6c_{i3}t_i + \cdots + n(n-1)c_{in}t_i^{n-2} \\ &\vdots \\ \frac{\mathrm{d}^k\sigma_{T_i}(t_i)}{\mathrm{d}t^k} &= \sum_{j=0}^{n-k} \frac{(k+j)!}{j!} c_{i(k+j)} t_i^j \end{aligned} \tag{C-5}$$

C-2 Quadratic Programming Optimization Problem

The QP optimization problem is a special case of a nonlinear programming problem and is formulated to minimize or maximize the cost J of the vector $\mathbf{c}$ as

$$\begin{aligned} \min \ J(\mathbf{c}) &= \frac{1}{2}\mathbf{c}^\top \mathbf{H}\mathbf{c} + \mathbf{f}^\top \mathbf{c} \\ \text{subjected to } \mathbf{A}_{in}\mathbf{c} &\leq \mathbf{b}_{in} \\ \text{and } \mathbf{A}_{eq}\mathbf{c} &= \mathbf{b}_{eq} \end{aligned} \tag{C-6}$$

where $\mathbf{H}$ is a symmetrical matrix reflecting the quadratic form of the problem and $\mathbf{f}$ is a vector reflecting the linear one. The vector $\mathbf{c}$ corresponds to the concatenation of the $m(n+1)$ coefficients c_{ij} written in Equation (C-4).

If the objective is to minimize a general derivative of the position, then it can be formulated as

$$\sigma_T^*(t) = \min \int_0^T \sum_{i=1}^{4} K_i \left(\frac{\mathrm{d}^k \sigma_{T_i}(t)}{\mathrm{d}t^k} \right)^2 \mathrm{d}t \tag{C-7}$$

with K_i a constant to turn the integral dimensionless. Since the trajectories are decoupled in all directions, the above Equation (C-7) can be seen as four different optimization problems. By choosing to minimize a derivative of the position, the $\mathbf{H}$ matrix will result in a block diagonal of m independent $(n+1) \times (n+1)$ matrices as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_m \end{bmatrix} \tag{C-8}$$

and the $\mathbf{f}$ vector will be null. Nevertheless, the waypoints are still related by imposing constraints in the continuity up until the pth derivative of the position, constraints to be imposed by $\mathbf{A}_{eq}$ and $\mathbf{b}_{eq}$. Maximum or minimum values for acceptable velocities, accelerations, jerks or snaps using $\mathbf{A}_{in}$ or $\mathbf{b}_{in}$ can also be imposed. However, minimizing a derivative of the position is already indirectly imposing acceptable trajectories, and in this work the inequality equation is neglected.

C-3 Values of k, n and p

In this work, minimizing the power of jerk was defined as the goal, so that $k = 3$. The degree of the polynomial was set to $n = 5$, so that there are still sufficient coefficients when minimizing the jerk power. In the case that $n = 5$, the jerk is a second order time polynomial, which is sufficient to give reliable results. Moreover, continuity constraints were imposed until $p = 2$, in order to guarantee continuity at least until the second derivative of the position, the acceleration.

C-4 Structure of H, $\mathbf{A}_{eq}$ and $\mathbf{b}_{eq}$

From Equation C-5, jerk in a general direction and for each time segment can be obtained as

$$j_i(t_i) = \frac{\mathrm{d}^3\sigma_{T_i}(t_i)}{\mathrm{d}t_i^3} = 6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2, \tag{C-9}$$

the jerk power can be obtained as

$$\begin{aligned} j_i^2(t_i) = \ & \left(6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2\right)^2 = \\ & (6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2)6c_{i3} + \\ & (6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2)24c_{i4}t_i + \\ & (6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2)60c_{i5}t_i^2, \end{aligned} \tag{C-10}$$

and the integral of the jerk power is given by

$$\begin{aligned} \int_0^{T_i} j_i^2\left(t_i\right)\ \mathrm{d}t_i = & \int_0^{T_i} \left(6c_{i3} + 24c_{i4}t_i + 60c_{i5}t_i^2\right)^2\ \mathrm{d}t_i = \\ & c_{i3}\left(36c_{i3}T_i + 72c_{i4}T_i^2 + 120c_{i5}T_i^3\right) + \\ & c_{i4}\left(72c_{i3}T_i^2 + 192c_{i4}T_i^3 + 360c_{i5}T_i^4\right) + \\ & c_{i5}\left(120c_{i3}T_i^3 + 360c_{i4}T_i^4 + 720c_{i5}T_i^5\right) \end{aligned} \tag{C-11}$$

Finally, the $\mathbf{H}$ matrix, as mentioned in equation (C-8), is composed by the m matrices $\mathbf{H}_i$ as in Equation (C-12).

$$\mathbf{H}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 36T_i & 72T_i^2 & 120T_i^3 \\ 0 & 0 & 0 & 72T_i^2 & 192T_i^3 & 360T_i^4 \\ 0 & 0 & 0 & 120T_i^3 & 360T_i^4 & 720T_i^5 \end{bmatrix} \tag{C-12}$$

The matrix $\mathbf{A}_{eq}$ and the vector $\mathbf{b}_{eq}$ result of the constraints in continuity of derivatives, and on the connection of the polynomial positions, since all trajectories start and end in a given waypoint.

Therefore, there are three types of constraints given by

1. Final and initial position for each waypoint correspond either to a waypoint or to a distance between waypoints:

$$
\begin{aligned}
r_1(t_1 = 0) &= wp_0 \\
&\vdots \\
r_m(t_m = 0) &= wp_{m-1} \\
r_1(t_1 = T_1) &= \Delta wp_1 \\
&\vdots \\
r_m(t_m = T_m) &= \Delta wp_m
\end{aligned}
$$

2. Initial and final velocity and acceleration ($p = 2$) are null:

$$
\begin{aligned}
v_1(t_1 = 0) &= 0 \\
a_1(t_1 = 0) &= 0 \\
v_m(t_m = T_m) &= 0 \\
a_m(t_m = T_m) &= 0
\end{aligned}
$$

3. Continuity in the velocity and acceleration ($p = 2$):

$$
\begin{aligned}
v_1(t_1 = T_1) &= v_2(t_2 = 0) \\
&\vdots \\
v_{m-1}(t_{m-1} = T_{m-1}) &= v_m(t_m = 0) \\
a_1(t_1 = T_1) &= a_2(t_2 = 0) \\
&\vdots \\
a_{m-1}(t_{m-1} = T_{m-1}) &= a_m(t_m = 0)
\end{aligned}
$$

Finally, it is possible to derive $\mathbf{b}_{eq}$ as in Equation (C-13) and $\mathbf{A}_{eq}$ as in Equation (C-14).

$$
\mathbf{b}_{eq} = \begin{bmatrix} wp_0 & 0 & 0 & \Delta wp_1 & |wp_1 & 0 & 0 & \Delta wp_2| & \cdots & \cdots & | & wp_{m-1} & 0 & 0 & \Delta wp_m| & 0 & 0 \end{bmatrix}^\top \tag{C-13}
$$

$$
A_{eq} = \left[\begin{array}{cccccc|cccccc|c|cccccc|cccccc}
1 & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & & & & & & & & & & & & & & & \\
 & & 2 & & & & & & & & & & & & & & & & & & & & & \\
0 & T_1 & T_1^2 & T_1^3 & T_1^4 & T_1^5 & & & & & & & & & & & & & & & & & & \\
\hline
 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
0 & -1 & -2T_1 & -3T_1^2 & -4T_1^3 & -5T_1^4 & & 1 & & & & & & & & & & & & & & & & \\
0 & 0 & -2 & -6T_1 & -12T_1^2 & -20T_1^3 & & & 2 & & & & & & & & & & & & & & & \\
 & & & & & & 0 & T_2 & T_2^2 & T_2^3 & T_2^4 & T_2^5 & & & & & & & & & & & & \\
\hline
 & & & & & & & & & & & & \ddots & & & & & & & & & & & \\
\hline
 & & & & & & & & & & & & & & & & & & 1 & & & & & \\
 & & & & & & & & & & & & & 0 & -1 & -2T_{m-1} & -3T_{m-1}^2 & -4T_{m-1}^3 & -5T_{m-1}^4 & & 1 & & & \\
 & & & & & & & & & & & & & 0 & 0 & -2 & -6T_{m-1} & -12T_{m-1}^2 & -20T_{m-1}^3 & & & 2 & & \\
 & & & & & & & & & & & & & & & & & & 0 & T_m & T_m^2 & T_m^3 & T_m^4 & T_m^5 \\
\hline
 & & & & & & & & & & & & & & & & & & 0 & -1 & -2T_m & -3T_m^2 & -4T_m^3 & -5T_m^4 \\
 & & & & & & & & & & & & & & & & & & 0 & 0 & -2 & -6T_m & -12T_m^2 & -20T_m^3
\end{array}\right]
\tag{C-14}
$$

Appendix D

Identification of the Drag Coefficients

In this Appendix the flight tests that allowed to obtain the results (see Section 5-1) of the identification of the drag coefficients are shown. The test velocity setpoint V_T was varied in the set $V_T \in \{0.5,\ 1.0,\ 1.5,\ \cdots,\ 4.0\}$ [m/s], which is represented from the top-left to the bottom-right in the Figures D-1 to D-4. The total number of points that represent the collected data in the Figures correspond only to a small portion of the used points, for the sake of readability.

In the end of this Appendix, a sensitivity analysis is also performed on the filtering aggressiveness, by varying the filter cut-off frequency ω_n, which shows that the results of this work are consistent for every tested filter type. In the sensitivity analysis, a measure of coherence given by the VAF (variance accounted for) is performed, as in Equation D-1.

$$VAF = \left(1 - \frac{\sum_{i=1}^{N} \|\mathbf{u}(i) - \hat{\mathbf{u}}(i)\|^2}{\sum_{i=1}^{N} \mathbf{u}^2(i)}\right) \cdot 100\ [\%] \tag{D-1}$$

where $\|\mathbf{a}\|$ is the Euclidean norm of the vector $\mathbf{a}$, and N is the number of samples.

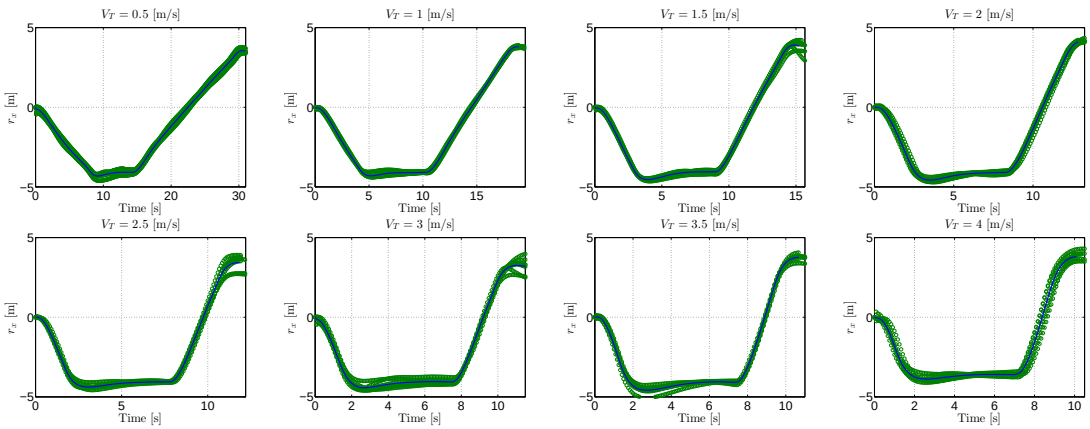


Figure D-1: Position.

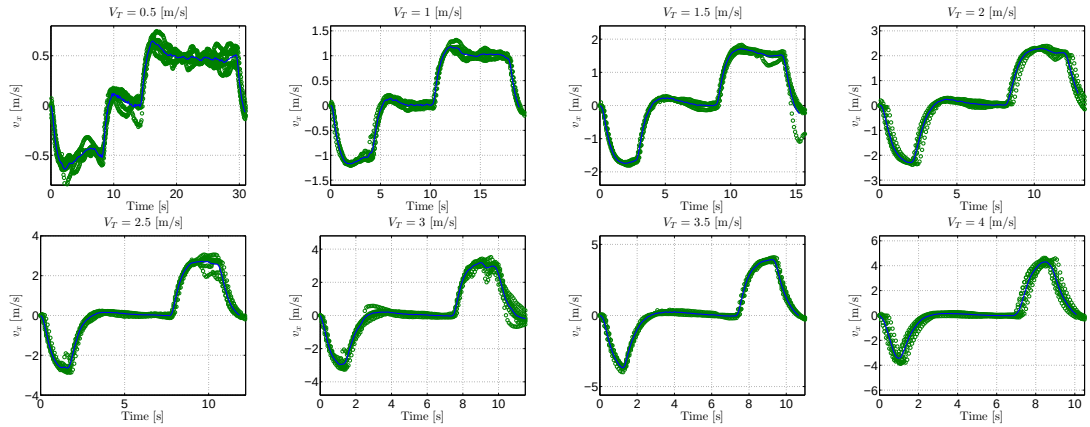


Figure D-2: Velocity.

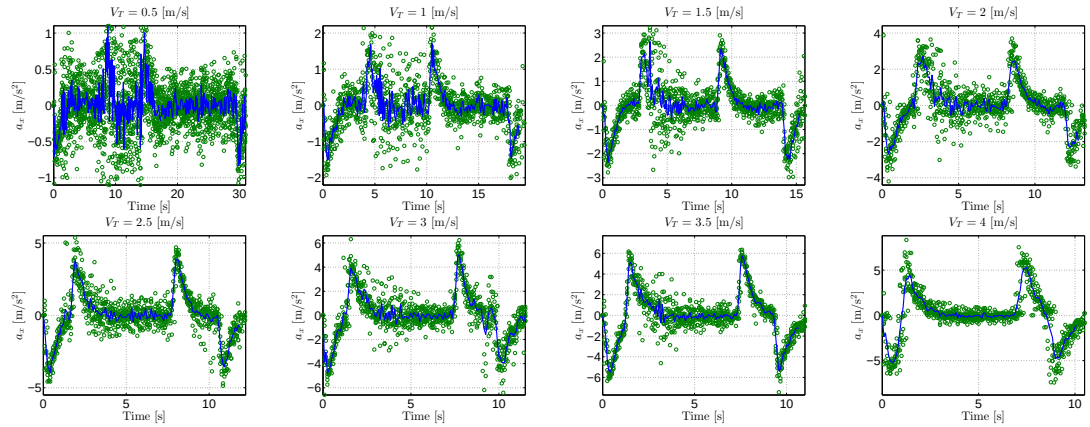


Figure D-3: Acceleration.

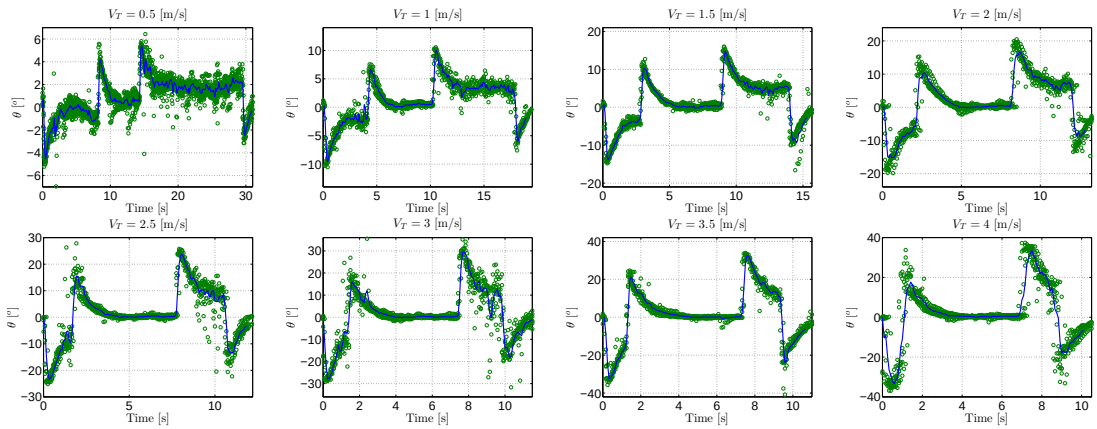


Figure D-4: Pitch angle.

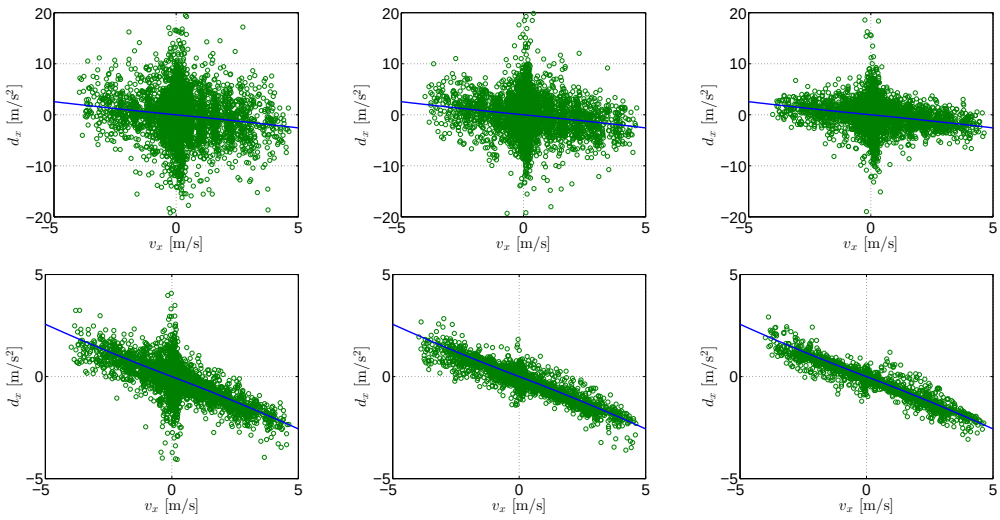


Figure D-5: Least squares fitting to obtain the drag coefficients. On the top with relatively small filtering, from left to right: no filtering, $\omega_n = 0.8$, $\omega_n = 0.6$. On the bottom with relatively big filtering, from left to right: $\omega_n = 0.3$, $\omega_n = 0.2$, $\omega_n = 0.1$.

ω_n [-]	$\hat{a}_{flap}$ [s/m]	$\hat{k}_{par}$ [kg/m]	VAF [%]
-	-0.04754	-0.00376	1.65
0.8	-0.04762	-0.00368	2.73
0.6	-0.04762	-0.00364	5.27
0.3	-0.04764	-0.00364	50.05
0.2	-0.04766	-0.00360	80.96
0.1	-0.04768	-0.00360	89.33
	-0.0476	-0.0036	

Table D-1: Influence of the cut-off frequency on the drag coefficients determination.

Appendix E

Influence of the Lumped Drag Coefficient

In this Appendix the flight tests that allowed to obtain the results (see Section 5-2) to see the influence of the lumped drag coefficient in the controller are shown. The test velocity setpoint V_T was varied in the set $V_T \in \{0.5,\ 1.0,\ 1.5,\ \cdots,\ 4.0\}$ [m/s], which is represented respectively from the top-left to the bottom-right in the Figures E-1 to E-4. Figures E-1 and E-2 show the results when the drag coefficient is neglected, while Figures E-3 and E-4 show the results when it is accounted. The total number of points that represent the collected data in the Figures correspond only to a small portion of the used points, for the sake of readability.

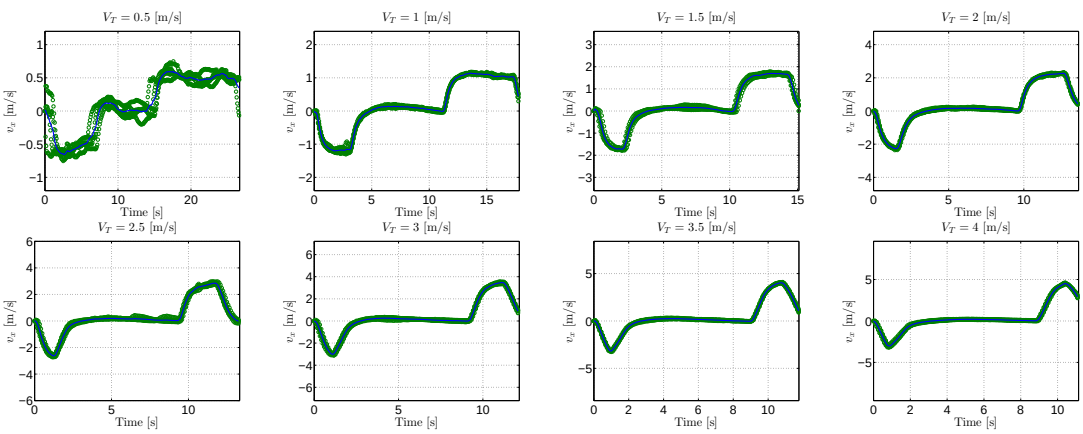


Figure E-1: Without drag coefficient - Velocity.

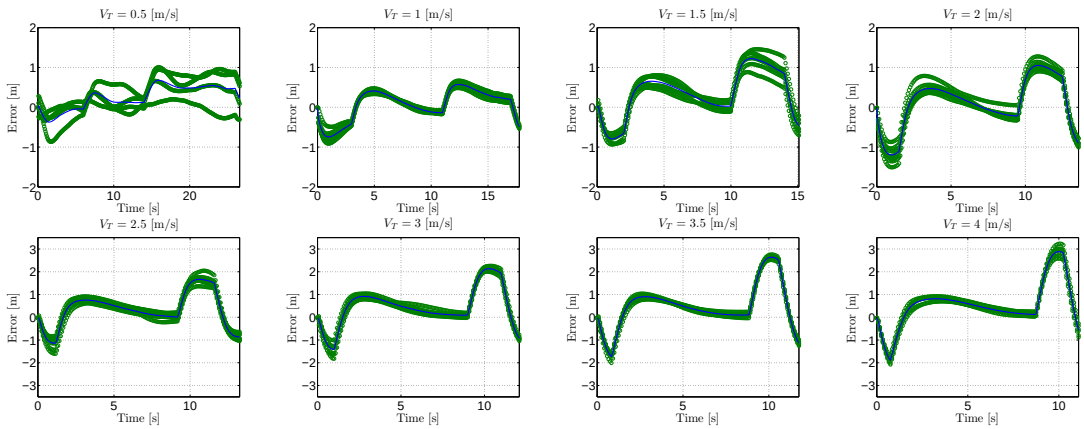


Figure E-2: Without drag coefficient - Position error.

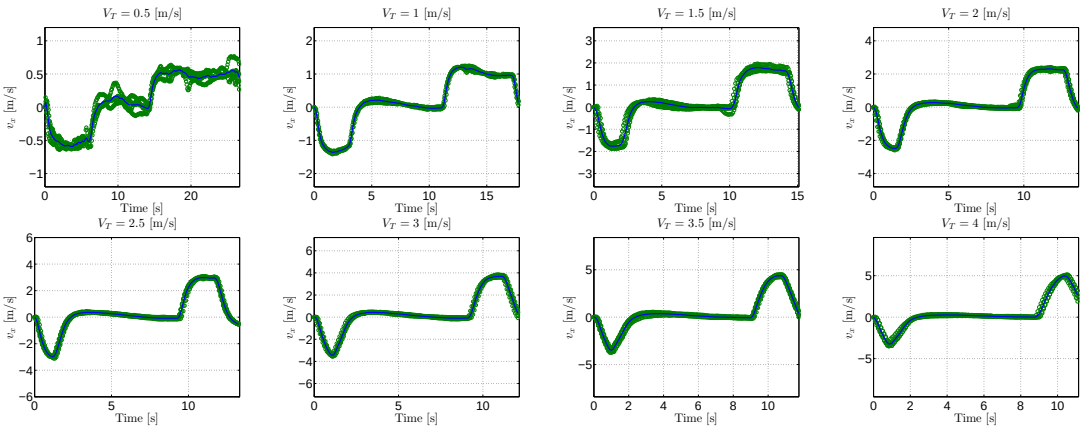


Figure E-3: With drag coefficient - Velocity.

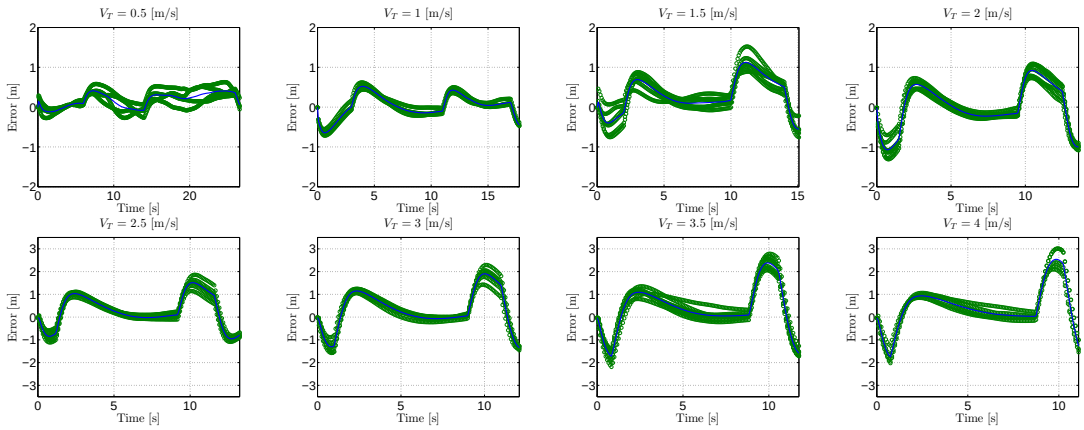


Figure E-4: With drag coefficient - Position error.

Appendix F

Reference Generator Comparison

In this Appendix the flight tests that allowed to obtain the results (see Section 5-6) that compare the two implemented reference generators are shown. One reference generator uses a discrete point to point followed by low-pass filtering approach, as discussed in Section 4-3, and the other uses the time optimal trajectory as discussed in Section 3-2-1. Both trajectories are forced to pass by the same eight waypoints, being that the trajectory starts at $(x, y) = (-2, 2)$ [m]. The Figures show the overall 2D trajectory, the velocity magnitude, the thrust and the Euler angles. The velocity in body coordinates can be obtained with the velocity magnitude and the Euler angles. The total number of points that represent the collected data in the Figures correspond only to a small portion of the used points, for the sake of readability.

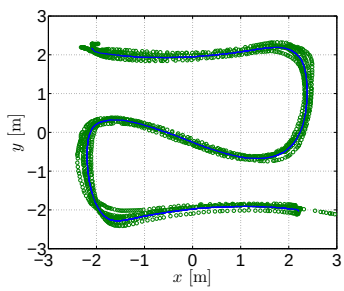


Figure F-1: Optimal - Position.

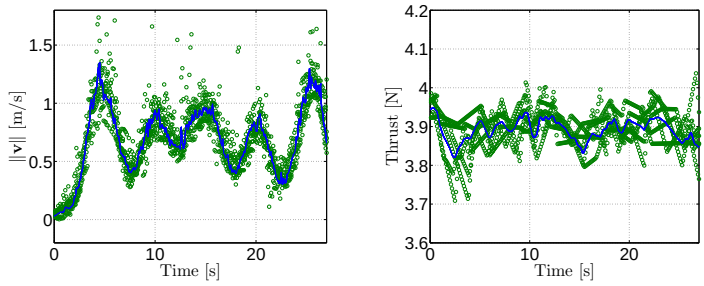


Figure F-2: Optimal - Velocity and thrust.

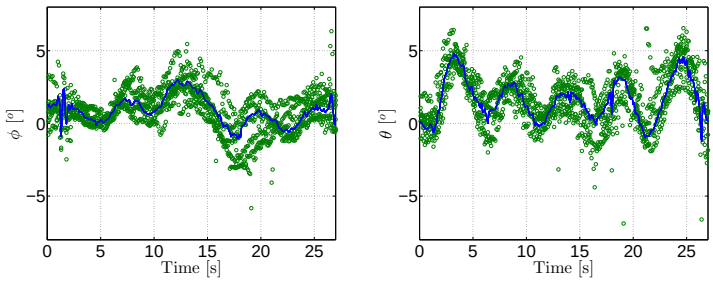


Figure F-3: Optimal - Euler angles.

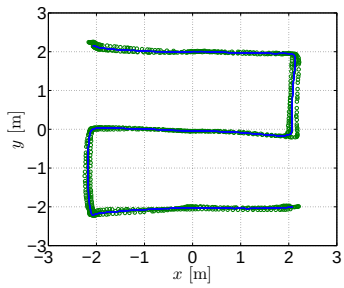


Figure F-4: Low-pass filter - Position.

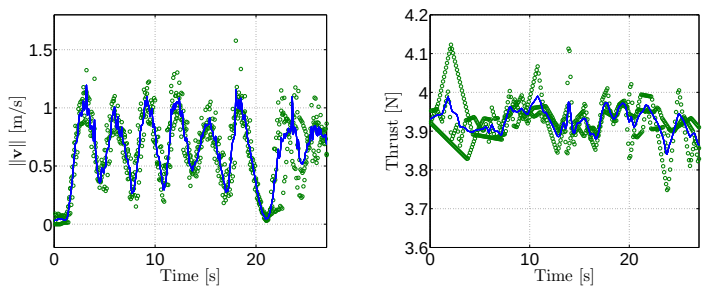


Figure F-5: Low-pass filter - Velocity and Thrust.

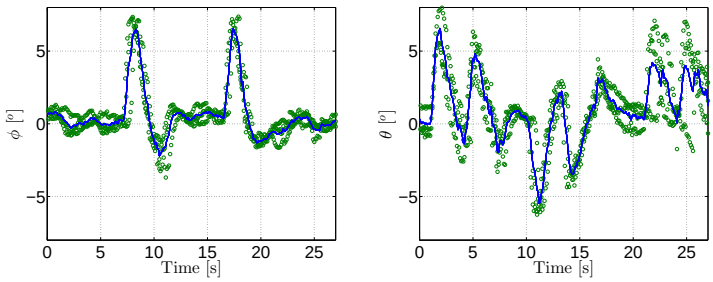


Figure F-6: Low-pass filter - Euler angles.

Appendix G

Controllers Comparison

In this Appendix the flight tests that allowed to obtain the results (see Section 5-7) that compare the implemented controllers are shown. One controller uses a PID approach, as discussed in Section 4-3, and the other uses a thrust vectoring approach, as discussed in Section 4-4. For each controller two different limit velocities are tested, $V_{lim} = 1 \ \vee \ V_{lim} = 2$ [m/s]. The Figures show the overall 2D trajectory, the position in x, the position in z, the roll angle and the pitch angle. The total number of points that represent the collected data in the Figures correspond only to a small portion of the used points, for the sake of readability.

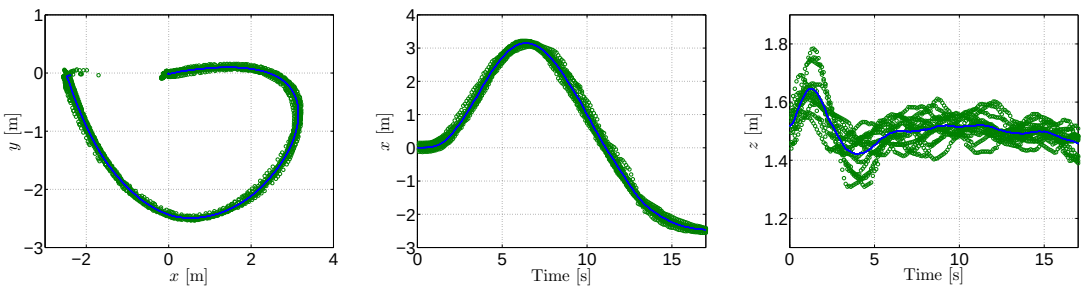


Figure G-1: Thrust vectoring controller, $V_{lim} = 1$ [m/s] - Position.

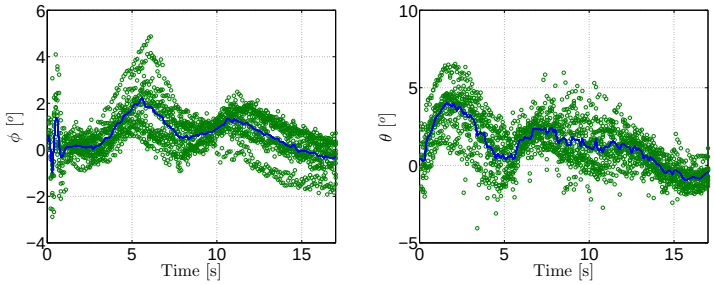


Figure G-2: Thrust vectoring controller, $V_{lim} = 1$ [m/s] - Euler angles.

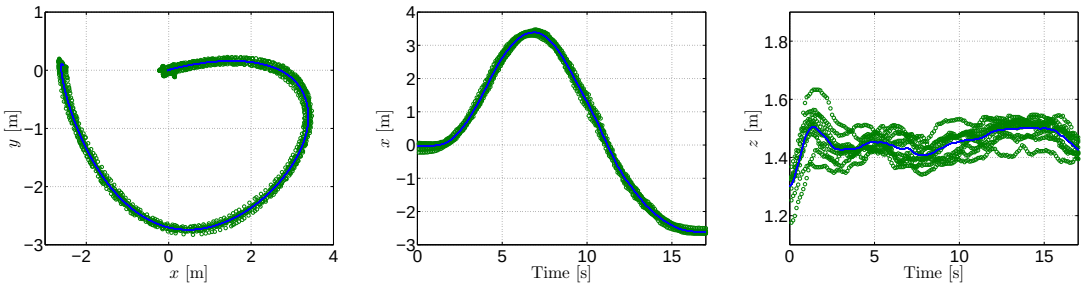


Figure G-3: PID controller, $V_{lim} = 1$ [m/s] - Position.

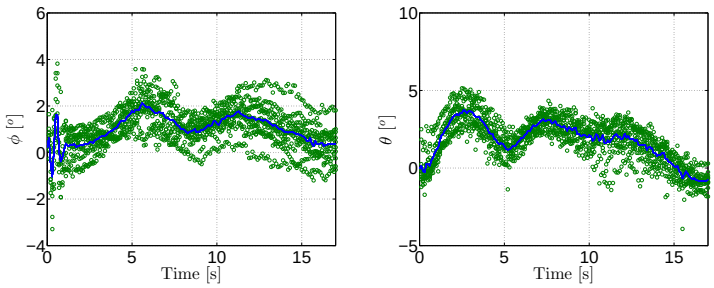


Figure G-4: PID controller, $V_{lim} = 1$ [m/s] - Euler angles.

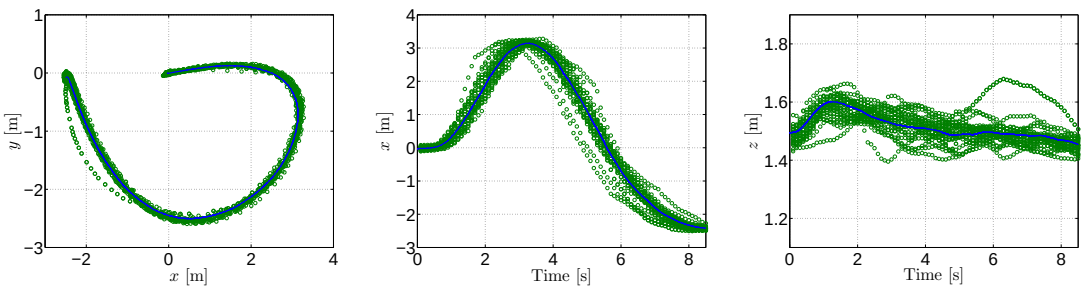


Figure G-5: Thrust vectoring controller, $V_{lim} = 2$ [m/s] - Position.

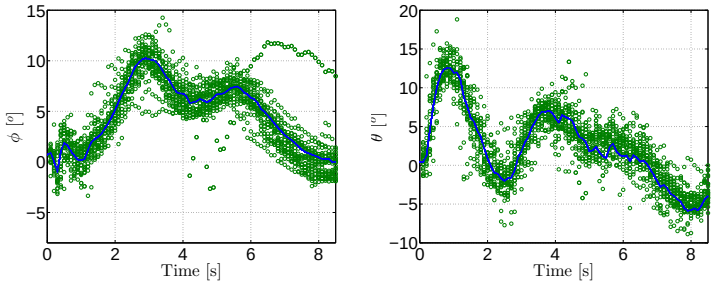


Figure G-6: Thrust vectoring controller, $V_{lim} = 2$ [m/s] - Euler angles.

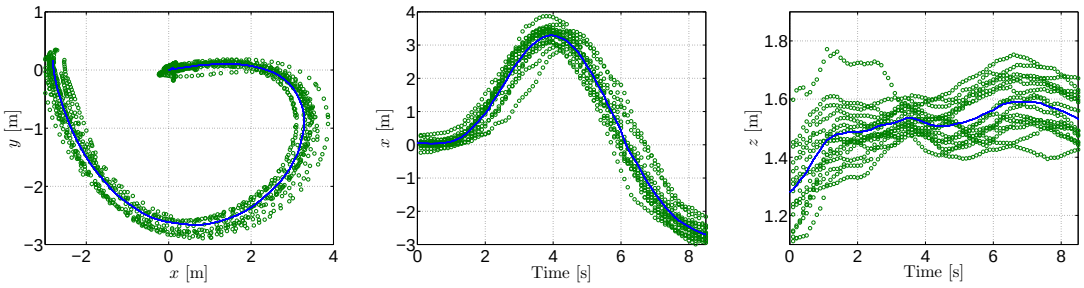


Figure G-7: PID, $V_{lim} = 2$ [m/s] - Position.

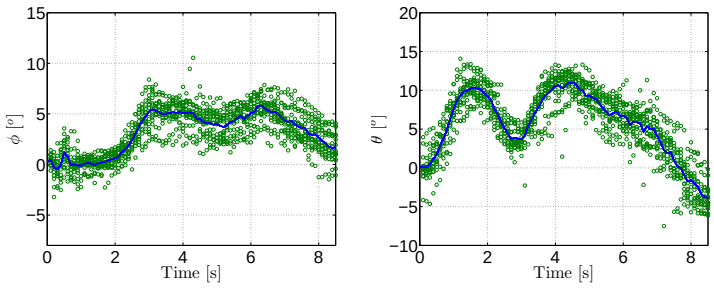


Figure G-8: PID, $V_{lim} = 2$ [m/s] - Euler angles.

Appendix H

Optimal Sequencing Analysis

In this Appendix the results (see Section 5-3) that allowed the analysis of the time optimal trajectory sequence are shown. The experiments were performed in a 64 bits Windows 7 OS, running MATLAB 2013a, with an Intel i5-2430M 2.40 GHz CPU processor and 4.00 GB of memory. Tables H-1-H-4 show the trajectory time, CPU time and problem dimension obtained for the four possible approaches: permutations; uniform search; heuristic h_1 search and; heuristic h_2 search. The waypoints were randomly generated in a $50 \times 50 \times 50$ [m^3] box, while randomizing the wind velocity as fractions of the limit velocity, with a random angle between 0 [^{o}] to 360 [^{o}]. The maximum number of waypoints is eight, since the first three methods can not support more than that. The value of K_{heu} is set to $K_{heu} = 0.9$ Table H-5 shows the optimal trajectory time, as well as the trajectory time resulting from the heuristic h_2. The optimal time is obtained by averaging the trajectory time of the three first approaches, while the h_2 trajectory time is directly obtained from Table H-4. Table H-6 shows the results of a sensitivity analysis performed to obtain the optimal value of K_{heu}, which represents a trade-off between the optimal solution and the computational time. The waypoints are generated in the same $50 \times 50 \times 50$ [m^3] box, but the comparison is being done with an average wind speed of 50 [%] the limit velocity since it is already a considerable amount of wind. The wind direction is still randomized in the 0 [^{o}] to 360 [^{o}] interval. Table H-7 shows the results of four waypoints, either symmetrically or asymmetrically distributed around the origin. It allows to see the time optimal trajectory time estimate of the TSP solution, and the exact optimal trajectory time after all steps of the trajectory generation process are completed. Wind is tested for two directions, either $\alpha_1 = 20$ [^{o}] or $\alpha_1 = 110$ [^{o}].

Table H-1: Permutations - Trajectory time, CPU time and problem dimension.

		m											
		1			2			3			4		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	4,80	0,19	1,00	10,67	0,34	2,00	15,72	0,54	6,00	19,87	1,43	24,00
	20	4,93	0,16	1,00	11,03	0,32	2,00	15,70	0,48	6,00	20,24	1,59	24,00
	30	5,16	0,17	1,00	11,03	0,35	2,00	16,23	0,50	6,00	20,47	1,26	24,00
	40	5,45	0,18	1,00	11,58	0,26	2,00	16,88	0,51	6,00	21,48	1,33	24,00
	50	5,93	0,16	1,00	12,46	0,28	2,00	18,11	0,48	6,00	22,95	1,30	24,00
	60	6,56	0,18	1,00	13,58	0,27	2,00	19,93	0,50	6,00	24,86	1,28	24,00
	70	7,56	0,19	1,00	15,89	0,28	2,00	22,79	0,49	6,00	28,75	1,29	24,00
	80	10,10	0,18	1,00	19,70	0,28	2,00	28,11	0,52	6,00	35,91	1,24	24,00
	90	17,74	0,16	1,00	32,71	0,27	2,00	46,25	0,50	6,00	59,10	1,33	24,00

		m											
		5			6			7			8		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	23,54	6,14	120,00	27,37	38,76	720,00	29,97	312,55	5040,00	33,94	3224,36	40320,00
	20	24,19	5,94	120,00	27,48	39,05	720,00	30,55	439,71	5040,00	32,92	3044,34	40320,00
	30	24,81	5,95	120,00	28,68	37,48	720,00	31,13	419,69	5040,00	37,50	4240,29	40320,00
	40	25,46	6,23	120,00	28,30	38,18	720,00	32,09	400,49	5040,00	36,72	3599,49	40320,00
	50	27,20	6,00	120,00	31,37	36,24	720,00	35,17	360,19	5040,00	35,92	3607,99	40320,00
	60	30,19	6,18	120,00	33,98	39,69	720,00	39,53	439,73	5040,00	40,34	2985,75	40320,00
	70	34,02	6,17	120,00	39,28	53,43	720,00	43,08	439,40	5040,00	46,42	2563,90	40320,00
	80	41,94	5,85	120,00	49,91	55,56	720,00	54,27	454,65	5040,00	58,83	2523,39	40320,00
	90	67,55	6,09	120,00	78,65	45,18	720,00	80,11	456,63	5040,00	90,16	2638,85	40320,00

Table H-2: Uniform - Trajectory time, CPU time and problem dimension

		m											
		1			2			3			4		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	4,79	0,30	1,00	10,74	0,73	2,00	15,57	1,94	6,00	19,65	7,57	23,92
	20	4,98	0,28	1,00	10,84	0,76	2,00	15,96	1,95	6,00	20,31	7,86	23,89
	30	5,10	0,34	1,00	11,20	0,74	2,00	16,02	1,88	6,00	20,73	7,50	23,85
	40	5,40	0,34	1,00	11,77	0,71	2,00	16,82	1,87	6,00	21,63	7,24	23,83
	50	5,80	0,32	1,00	12,45	0,74	2,00	18,23	1,89	6,00	22,84	7,13	23,68
	60	6,59	0,30	1,00	13,60	0,74	2,00	19,64	1,86	6,00	25,12	7,44	23,52
	70	7,88	0,29	1,00	15,80	0,70	2,00	22,80	1,80	6,00	29,09	7,22	23,29
	80	10,07	0,28	1,00	20,19	0,70	2,00	28,34	1,82	6,00	36,02	6,27	22,91
	90	17,55	0,30	1,00	33,47	0,75	2,00	46,88	1,68	6,00	58,98	5,64	22,08

		m											
		5			6			7			8		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	23,56	41,44	114,82	26,39	274,82	590,70	31,30	8534,61	3507,45	35,94	84766,43	12435,00
	20	23,97	40,57	114,30	26,84	281,02	599,08	29,96	4315,46	3120,15	34,11	206336,00	17980,00
	30	24,51	36,79	113,41	27,87	389,53	582,16	31,13	6354,86	3280,50	32,12	134555,85	15378,00
	40	25,73	39,07	112,40	28,91	283,15	572,64	31,47	3928,97	2883,45	32,07	289832,37	19634,00
	50	27,71	34,11	111,24	30,83	230,34	550,07	37,51	7868,09	3174,25	41,19	268831,04	22752,00
	60	29,71	35,79	108,32	34,01	225,47	541,72	37,37	5016,41	2741,50	44,06	24098,30	9681,00
	70	34,18	33,04	104,59	40,07	259,67	514,64	45,27	4103,31	2619,75	53,33	139465,44	17821,00
	80	42,19	29,65	99,05	48,75	251,96	475,06	55,32	3274,14	2330,85	53,39	41134,01	11033,00
	90	67,04	24,10	90,57	75,04	161,65	395,53	96,16	2511,87	1967,55	118,59	2352,56	3553,00

Table H-3: Heuristic h_1 - Trajectory time, CPU time and problem dimension.

		m											
		1			2			3			4		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	4,81	0,33	1,00	10,73	0,83	2,00	15,62	1,89	5,93	19,92	5,69	21,87
	20	4,89	0,34	1,00	10,97	0,79	2,00	16,13	1,83	5,94	20,23	5,62	22,10
	30	5,15	0,35	1,00	11,39	0,83	2,00	16,35	1,88	5,95	20,69	5,71	22,30
	40	5,39	0,32	1,00	11,68	0,70	2,00	16,87	1,88	5,96	21,67	5,27	22,40
	50	5,89	0,33	1,00	12,50	0,73	2,00	18,11	1,90	5,98	22,77	5,29	22,26
	60	6,56	0,31	1,00	13,77	0,78	2,00	19,78	1,85	5,98	25,38	5,44	22,23
	70	7,74	0,34	1,00	15,72	0,74	2,00	22,43	1,89	5,99	28,62	5,23	22,07
	80	10,33	0,34	1,00	20,02	0,78	2,00	27,80	1,85	5,98	35,57	5,33	21,87
	90	18,85	0,30	1,00	34,17	0,69	2,00	46,50	1,93	6,00	59,39	5,13	21,49

		m											
		5			6			7			8		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	23,75	22,42	92,46	26,49	120,98	418,75	30,19	1417,63	2015,83	38,57	123925,29	15435,00
	20	24,08	21,51	92,68	27,27	130,81	426,76	30,78	1463,11	2024,70	33,14	15569,59	6860,00
	30	24,73	22,61	93,93	28,10	123,87	417,31	31,87	1760,18	2080,76	34,59	19282,16	7568,00
	40	25,34	28,62	93,51	29,23	153,00	430,92	33,46	2580,87	2146,50	37,38	79952,31	12628,00
	50	27,41	28,81	94,81	30,94	149,11	435,50	35,57	2162,42	2064,46	31,55	2613,19	2622,00
	60	29,83	30,85	92,86	34,53	125,80	416,13	37,29	1830,07	1890,90	41,56	26993,50	7617,00
	70	33,96	26,17	92,34	39,16	163,94	408,24	44,89	2171,90	2045,85	50,51	39590,62	9381,00
	80	42,42	25,19	89,54	47,86	144,33	389,85	53,82	1684,58	1821,50	62,84	31615,10	8700,00
	90	69,22	24,41	85,17	78,32	119,14	358,75	88,49	1277,01	1577,11	80,55	6311,58	4565,00

Table H-4: Heuristic h_2 - Trajectory time, CPU time and problem dimension.

		m											
		1			2			3			4		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	4,82	0,44	1,00	10,68	1,07	2,00	15,78	2,18	5,16	20,01	3,48	11,25
	20	4,93	0,53	1,00	10,88	1,04	2,00	16,10	2,11	5,17	20,38	3,55	11,48
	30	5,08	0,50	1,00	11,15	1,13	2,00	16,58	2,15	5,13	21,55	3,79	11,72
	40	5,40	0,41	1,00	11,80	1,19	2,00	17,37	1,80	5,14	22,29	3,70	11,86
	50	5,89	0,40	1,00	12,51	1,28	2,00	18,62	1,78	5,12	24,34	3,89	12,29
	60	6,55	0,43	1,00	13,79	1,14	2,00	20,63	1,85	5,13	26,69	4,04	12,27
	70	7,81	0,42	1,00	16,06	1,01	2,00	23,63	1,95	5,12	31,10	4,53	12,87
	80	10,09	0,45	1,00	20,72	1,04	2,00	30,47	1,98	5,12	40,24	4,68	13,27
	90	18,85	0,44	1,00	35,51	1,09	2,00	50,50	1,93	5,13	67,36	4,54	13,25

		m											
		5			6			7			8		
		T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]	T [s]	CPU Time [ms]	N_{OL} [-]
$\frac{V_w}{V_{lim}}$ [%]	10	24,03	6,46	19,07	28,16	7,44	27,90	31,66	8,84	37,13	35,52	12,54	46,77
	20	25,11	6,11	20,07	28,78	7,08	28,48	32,67	8,83	37,10	35,01	10,29	44,27
	30	25,75	7,09	21,04	29,61	6,75	29,11	33,52	9,53	38,32	37,53	11,61	47,07
	40	27,16	6,26	22,67	31,53	8,11	32,56	34,89	9,86	39,59	39,04	13,64	52,17
	50	29,14	5,78	23,68	33,88	8,90	37,06	38,59	11,94	47,65	42,83	15,08	57,48
	60	32,70	6,66	27,10	37,94	11,27	43,50	42,72	13,18	54,72	47,01	18,99	66,63
	70	37,83	7,54	28,75	44,23	13,92	52,02	50,25	21,54	73,42	55,93	20,44	77,87
	80	49,51	8,71	32,35	58,42	19,05	65,31	65,28	31,12	90,62	73,07	35,43	118,62
	90	83,34	10,15	34,14	98,39	24,45	79,45	111,39	59,36	129,20	126,41	40,57	120,58

Table H-5: Trajectory time - Optimal at the top and with heuristic h_2 at the bottom.

		m							
		1	2	3	4	5	6	7	8
		T [s]							
$\frac{V_w}{V_{lim}}$ [%]	10	4,80	10,71	15,64	19,81	23,61	26,75	30,49	36,15
	20	4,94	10,95	15,93	20,26	24,08	27,19	30,43	33,39
	30	5,14	11,21	16,20	20,63	24,68	28,22	31,38	34,73
	40	5,41	11,68	16,86	21,59	25,51	28,81	32,34	35,39
	50	5,87	12,47	18,15	22,86	27,44	31,05	36,09	36,22
	60	6,57	13,65	19,78	25,12	29,91	34,17	38,06	41,99
	70	7,73	15,80	22,67	28,82	34,05	39,51	44,41	50,09
	80	10,17	19,97	28,09	35,83	42,18	48,84	54,47	58,35
	90	18,05	33,45	46,54	59,16	67,94	77,34	88,25	96,44
	mean	7,63	15,54	22,21	28,23	33,27	37,99	42,88	46,97

		m							
		1	2	3	4	5	6	7	8
		T [s]							
$\frac{V_w}{V_{lim}}$ [%]	10	4,82	10,68	15,78	20,01	24,03	28,16	31,66	35,52
	20	4,93	10,88	16,10	20,38	25,11	28,78	32,67	35,01
	30	5,08	11,15	16,58	21,55	25,75	29,61	33,52	37,53
	40	5,40	11,80	17,37	22,29	27,16	31,53	34,89	39,04
	50	5,89	12,51	18,62	24,34	29,14	33,88	38,59	42,83
	60	6,55	13,79	20,63	26,69	32,70	37,94	42,72	47,01
	70	7,81	16,06	23,63	31,10	37,83	44,23	50,25	55,93
	80	10,09	20,72	30,47	40,24	49,51	58,42	65,28	73,07
	90	18,85	35,51	50,50	67,36	83,34	98,39	111,39	126,41
	mean	7,71	15,90	23,30	30,44	37,17	43,44	49,00	54,71

Table H-6: Determination of the coefficient K_{heu} - Trajectory time and CPU.

		K_{heu}					
		0.75	0.80	0.85	0.90	0.95	1.00
		T [s]					
m	1	5,884801	5,790527	5,851365	5,902371	5,789215	5,913984
	2	12,44196	12,46553	12,49656	12,45358	12,49784	12,62785
	3	18,06866	18,31328	18,43443	18,52131	18,7223	18,77173
	4	23,32552	23,42808	23,84668	24,22579	24,66889	24,68221
	5	28,39334	28,95835	28,57322	29,08222	29,20719	29,92069
	6	32,70283	33,40888	33,59603	34,10978	34,21422	35,01023
	7	37,28239	37,58863	38,15382	38,59284	38,54745	39,10977
	8	41,28914	41,51546	41,94088	42,69273	43,0779	43,33393

		K_{heu}					
		0.75	0.80	0.85	0.90	0.95	1.00
		CPU Time [ms]					
m	1	0,217419	0,218366	0,203417	0,212814	0,214504	0,221709
	2	0,52647	0,488056	0,481042	0,491817	0,509053	0,532718
	3	1,222492	1,067142	1,014925	1,028988	1,092751	0,989235
	4	2,680019	2,324996	2,332284	1,845469	1,839818	1,63438
	5	6,432114	5,178651	3,986635	3,204626	2,522151	2,227589
	6	13,85087	9,887837	5,878778	4,642328	3,203518	2,654877
	7	35,25819	15,65874	9,286663	5,528918	4,007238	3,413385
	8	127,6312	34,02294	13,00827	6,563606	3,974498	3,599298

Table H-7: Comparison of the TSP optimal sequence with the overall optimal sequence.

Symmetrical								Asymmetrical							
$\alpha_1 = 20$ [^{o}]				$\alpha_1 = 110$ [^{o}]				$\alpha_1 = 20$ [^{o}]				$\alpha_1 = 110$ [^{o}]			
ΔTSP [%]	T_{TSP} [s]	T_{OPT} [s]	ΔOPT [%]	ΔTSP [%]	T_{TSP} [s]	T_{OPT} [s]	ΔOPT [%]	ΔTSP [%]	T_{TSP} [s]	T_{OPT} [s]	ΔOPT [%]	ΔTSP [%]	T_{TSP} [s]	T_{OPT} [s]	ΔOPT [%]
0	3,55	6,49	3	0	3,55	6,54	2	0	2,66	4,36	0	0	2,8	4,2	0
1	3,59	8,7	39	1	3,59	9,14	43	7	2,84	4,98	14	13	3,15	5,61	34
5	3,73	6,28	0	5	3,73	6,4	0	7	2,85	4,63	6	14	3,19	6,05	44
6	3,77	14,13	125	6	3,77	13,65	113	9	2,91	7,45	71	16	3,25	8,69	107
10	3,9	12,6	101	10	3,9	12,03	88	19	3,16	5,48	26	23	3,43	6,26	49
11	3,94	6,99	11	11	3,94	7,28	14	22	3,24	8,06	85	23	3,45	6,63	58
12	3,96	7,36	17	12	3,96	7,34	15	24	3,3	5,92	36	26	3,53	9,24	120
13	4,02	8,26	32	13	4,02	8,54	33	29	3,42	5,69	31	30	3,63	7,84	87
14	4,03	10,05	60	14	4,03	9,6	50	32	3,51	5,71	31	30	3,64	6,19	47
14	4,06	7,74	23	14	4,06	8,5	33	33	3,53	5,28	21	30	3,65	6,82	62
17	4,14	7,25	15	17	4,14	7,24	13	33	3,55	6,52	50	32	3,7	9,66	130
19	4,22	11,56	84	19	4,22	11,6	81	36	3,62	10,64	144	33	3,72	7,06	68
21	4,31	10,45	66	21	4,31	10,48	64	41	3,74	7,7	77	37	3,84	7,69	83
24	4,39	9,46	51	24	4,39	9,35	46	41	3,75	9,63	121	43	4	6,74	60
24	4,4	10,97	75	24	4,4	9,97	56	43	3,8	7,31	68	45	4,06	7,08	69
25	4,43	13,98	123	25	4,43	13,37	109	43	3,81	8,57	97	46	4,09	7,3	74
25	4,44	9,28	48	25	4,44	9,3	45	43	3,81	7,26	67	46	4,1	7,55	80
27	4,5	13,78	119	27	4,5	13,3	108	45	3,86	8,01	84	48	4,13	7,38	76
29	4,59	14,54	132	29	4,59	15,4	141	54	4,09	7,55	73	50	4,21	8,33	98
30	4,62	8,72	39	30	4,62	8,81	38	56	4,16	8,49	95	53	4,28	8,62	105
34	4,76	8,61	37	34	4,76	8,64	35	59	4,22	7,09	63	55	4,35	9,02	115
35	4,8	16,1	156	35	4,8	15,64	144	61	4,28	7,19	65	58	4,41	7,62	81
37	4,87	9,02	44	37	4,87	9,03	41	62	4,32	8,74	100	58	4,41	8,43	101
38	4,91	11,31	80	38	4,91	11,44	79	69	4,5	9,46	117	70	4,76	9,01	115