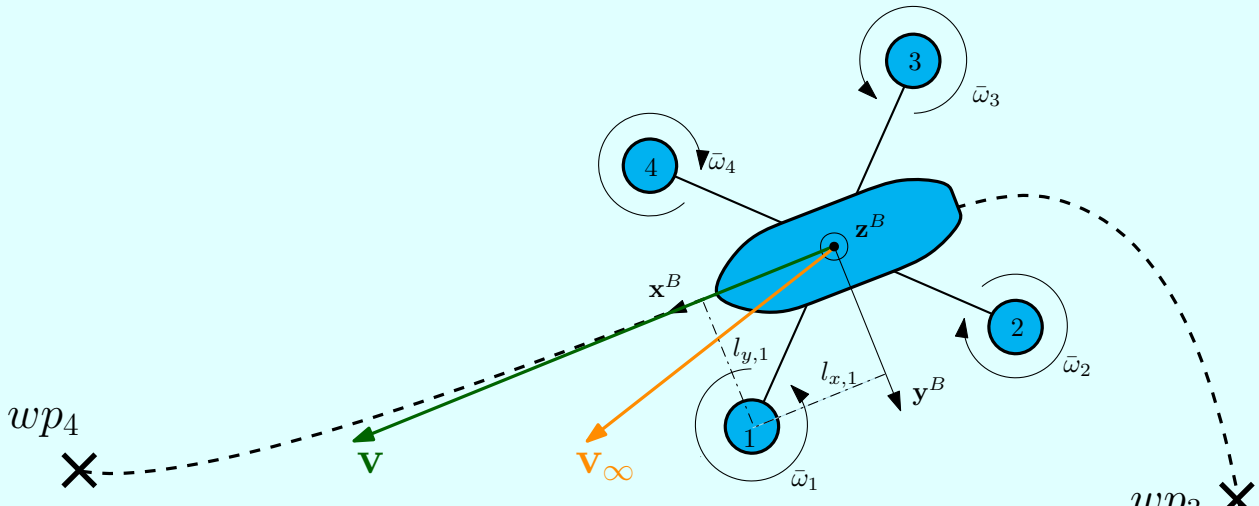
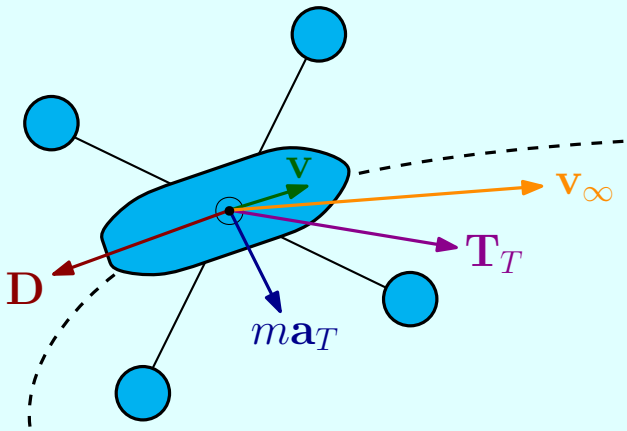
Faculty of Aerospace Engineering

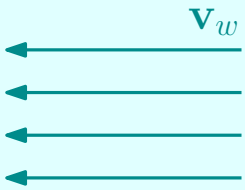


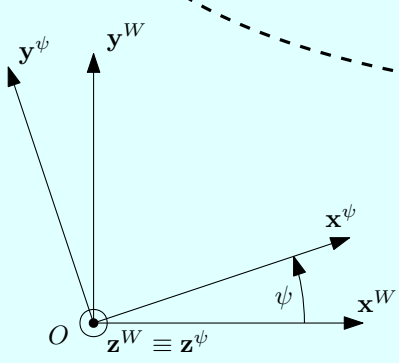
Quadrotor Thrust Vectoring Control with Time Optimal Trajectory Planning in Constant Wind Fields

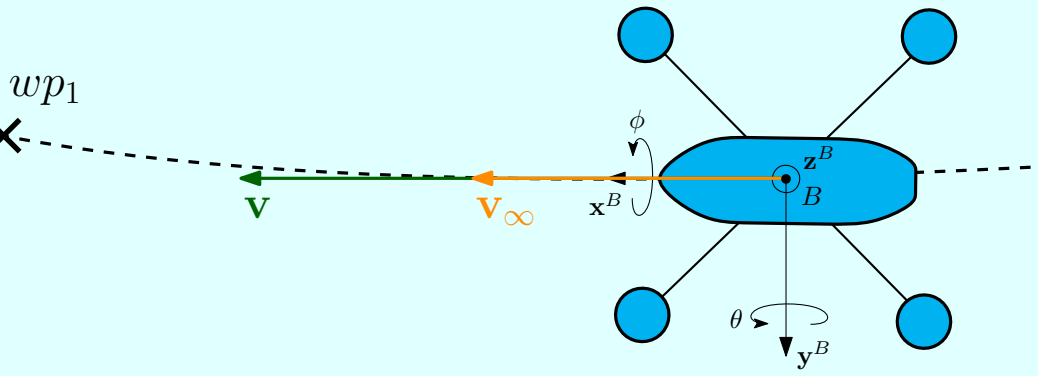
J. P. da Rocha Silva

August 9, 2016









TU Delft Delft University of Technology

Challenge the future

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

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

J. P. da Rocha Silva

August 9, 2016

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Quadrotor Thrust Vectoring Control with Time Optimal Trajectory Planning in Constant Wind Fields"** by **J. P. da Rocha Silva** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 9, 2016

Readers:

prof.dr.ir. M. Mulder

dr. G. de. Croon

ir. C. De Wagter

ir. M. Schuurman

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 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. 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, August 9, 2016

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 standard PID controller, showing an increase in performance by reducing the tracking delay and extending the flight envelope.

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
ζ	Filter damping ratio
θ	Pitch Euler angle
λ	Motor rotation direction
μ	Advancing ratio
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
$\mathbf{I}$	Identity matrix
J	Least squares cost
$\mathbf{J}$	Inertia matrix
k	Order of the derivative to minize
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
n	Polynomial order
p	Maximum derivative constraint
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	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
$\perp$	Perpendicular
$\top$	Transpose

List of Figures

List of Tables

Contents

Chapter 1

Introduction

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 and worsens when the wind is present. 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).

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 are visited, without restrictions in the visiting sequence. The total trajectory time has to be minimized while still maintaining feasibility, and the trajectory needs to be promptly computed to serve real-time applications. Wind influence on the quadrotors dynamics forces the inclusion of aerodynamic effects in the trajectory generation phase and to understand those effects a literature study is required. 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.

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 standard 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.

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 (w.r.t.) 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.

This work is structured in two Parts. Part I presents the scientific paper that is the final product of this work, and in Part II the appendices that support the scientific paper are shown.

Part I

Scientific Paper

Part II

Book of Appendices

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.

Figure A-1: Parrot Bebop with side bumpers.

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

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.

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

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

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 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.

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

Appendix B

Coding Implementation

In this Appendix the coding 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 then 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. These 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 standard 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 paper and are necessary in the coding 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()`;
- 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 w.r.t. 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.

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

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 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.

[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/

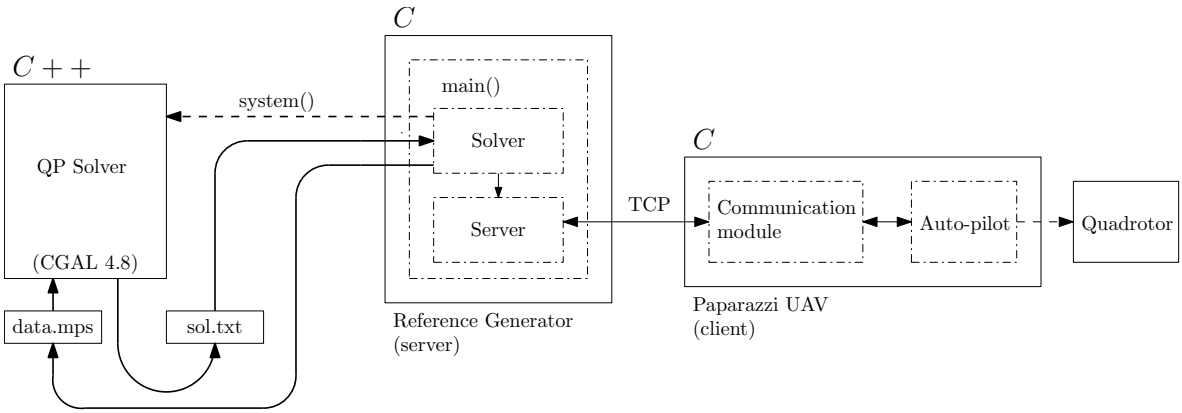


Figure B-1: Program Scheme containing the Solver and Client/Server implementations.

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.

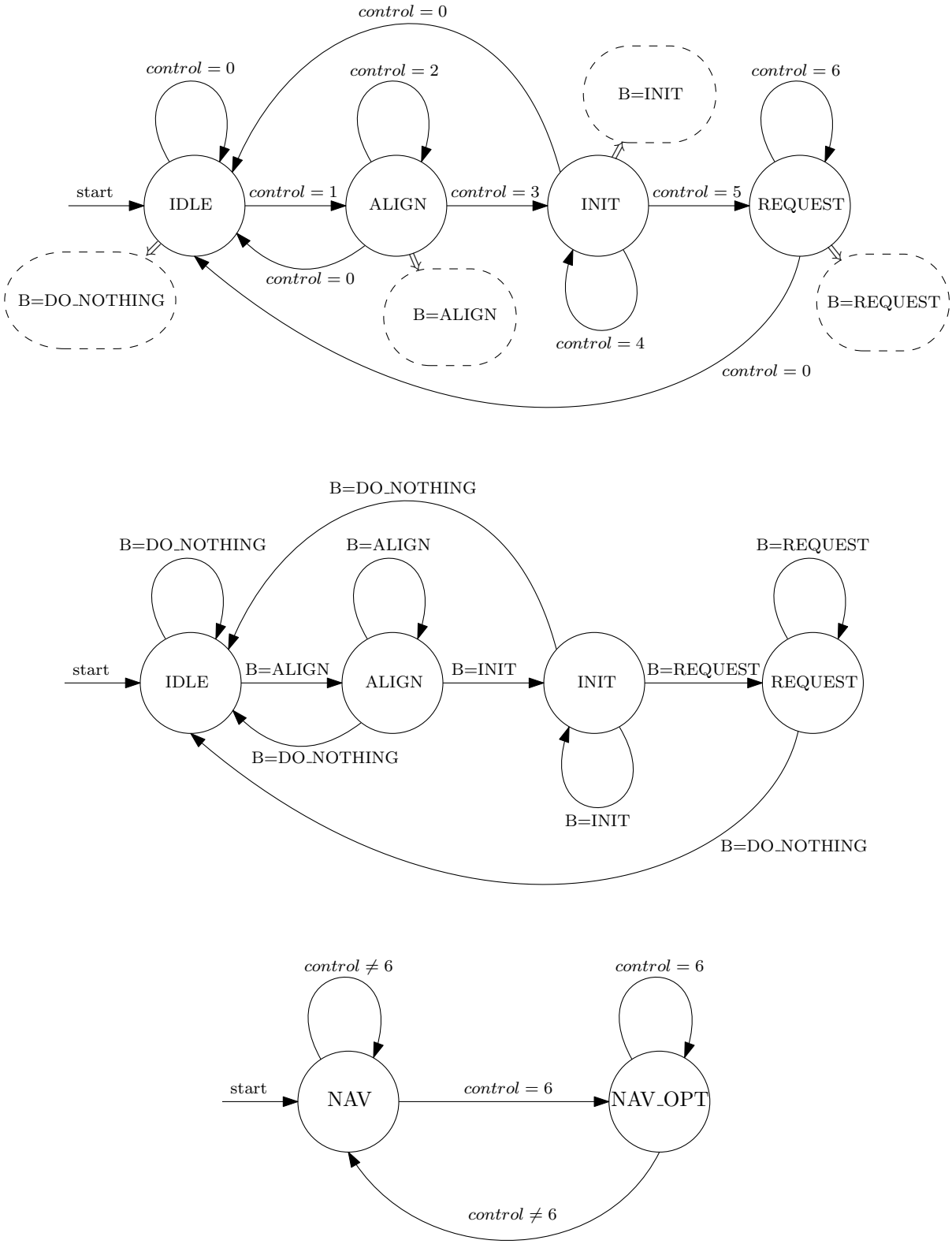


Figure B-2: Finite-State Machines - Top to bottom: Client, Server and Mode.

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.

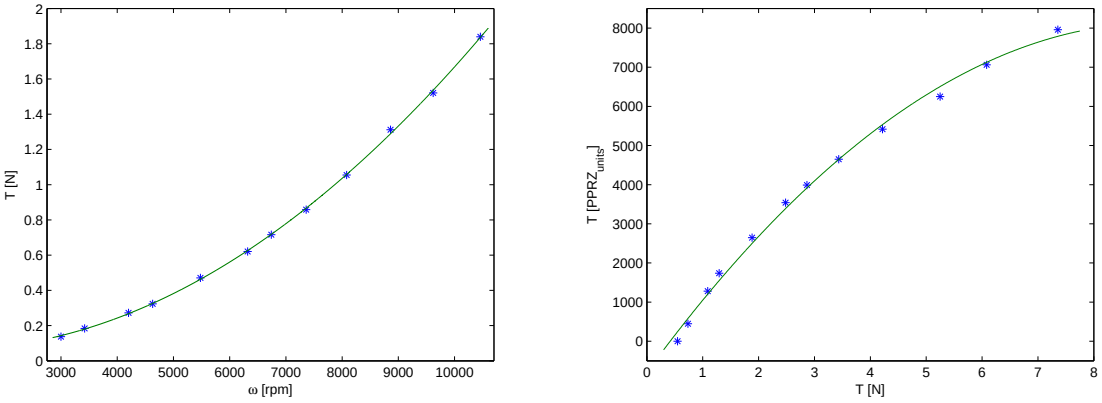
B-5 Thrust Mapping

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

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

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

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

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

Comparison with the Paparazzi Approach

In this Appendix a comparison is made between a standard reference generator and the proposed time optimal reference generator. The standard reference generators, such as the one embedded in Paparazzi, produce discrete point to point trajectory steps followed by a low-pass filter. This can be seen as a black-box that translates position requests into target position, velocity and acceleration. Therefore, the resulting trajectories will have an expected behavior, which will be referred to as the "Paparazzi approach". The Paparazzi approach is described here, in order to compare it with the Quadratic Programming approach that was implemented. In Paparazzi, a trajectory is not "generated" and instead the waypoints are fed to the autopilot controller. In the main outer-loop, there exists the already mentioned low-pass filter and the two inner-loops that control both the velocity and attitude of the quadrotor. Nevertheless, the trajectory that the quadrotor will achieve can be approximated by line segments in steady-state and circles in steady turns to represent overshoot between the waypoints. An hypothetical trajectory can be seen in Figure C-1.

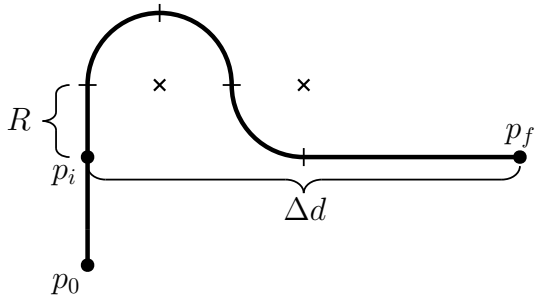


Figure C-1: Scheme of a general Paparazzi maneuver from p_i to p_f.

One can see an initial segment with length R, three quarter circles with radius R, and the final segment with length $\Delta d - 3R$, with Δd the distance between the two waypoints and

$$R = k_o \frac{\Delta d}{3} \tag{C-1}$$

with $k_o \in]0, 1]$ a constant which is related to the aggressiveness of the turn. For each quarter circle, the angular and linear velocities are

$$\omega_o = \frac{\pi}{2}\frac{1}{\Delta t_o} \qquad V_o = k_o \frac{\pi \Delta d}{6 \Delta t_o}, \tag{C-2}$$

and in the linear segments the quadrotor will travel at the limit velocity V_{lim}.

Being so, the total time from p_i to p_f is

$$\begin{aligned} T_{if} = \Delta t_i + 3\Delta t_o + \Delta t_f &= \frac{R}{V_{lim}} + k_o \frac{\pi \Delta d}{2 V_o} + \frac{\Delta d - 3R}{V_{lim}} \\ &= k_o \frac{\Delta d}{3 V_{lim}} + k_o \frac{\pi \Delta d}{2 V_o} + \frac{\Delta d - k_o \Delta d}{V_{lim}} \end{aligned} \tag{C-3}$$

Since

$$T_{if} = \frac{\Delta d}{V_{lim}}, \tag{C-4}$$

solving Equation (C-3) results in a linear velocity only dependent on the limit velocity

$$V_o = \frac{3\pi}{4} V_{lim}, \tag{C-5}$$

with the correspondent angular acceleration and jerk given by

$$a_o = \frac{V_o^2}{R} \qquad j_o = \frac{V_o^3}{R^2}. \tag{C-6}$$

From these results one can state the following:

- The velocity in the curve segments is more than twice the limit velocity;
- The acceleration and jerk are inversely related with k_o due to the R term;
- There are discontinuities in the velocity, acceleration and jerk at the start and at the end of a curve.

C-1 Velocity, Acceleration and Jerk

The previously mentioned aspects can be seen in Figures C-2-C-5 for $k_0 \in \{0.2,\ 0.5,\ 1\}$. They show the case when there is no wind and $V_{lim} = 3$ [m/s], for a small and a big trajectory. In order to obtain the same trajectory time between the Paparazzi and the QP approaches, the last step of the time optimal trajectory generation, that assures the velocity w.r.t. the airstream is below V_{lim}, is not performed. This is the case, since it would result always in a worst trajectory time for the Paparazzi approach, so it is a conservative approach.

Figures C-2 and C-3 qualitatively validate the preference of the QP approach over the Paparazzi approach, since the trajectories appear to be more natural in the first approach. It

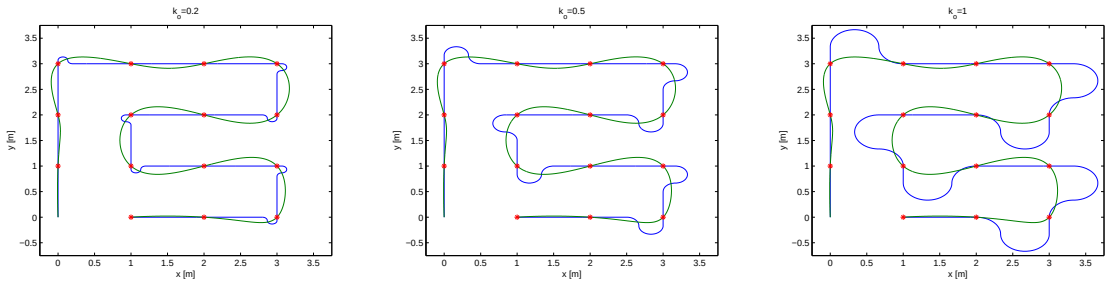


Figure C-2: Comparison between the small trajectory generated with the Paparazzi (blue) and QP (green) approaches for $k_o = 0.2$ (left), $k_o = 0.5$ (center), $k_o = 1$ (right).

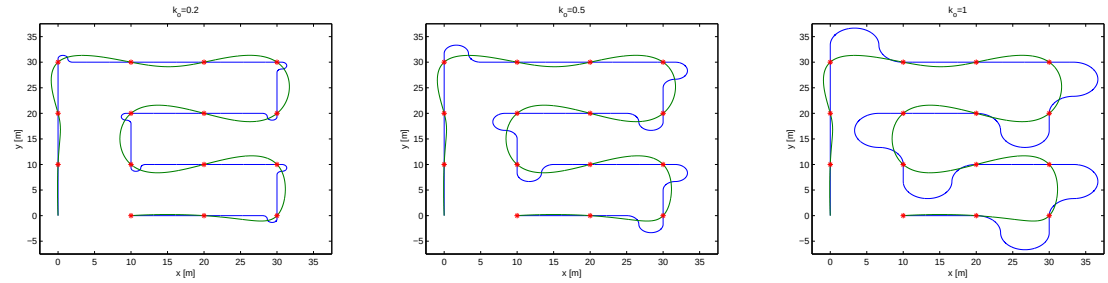


Figure C-3: Comparison between the big trajectory generated with the Paparazzi (blue) and QP (green) approaches for $k_o = 0.2$ (left), $k_o = 0.5$ (center), $k_o = 1$ (right).

also allows to see the influence of the aggressiveness parameter k_o in the shape of the trajectory. Moreover, one can see that the trajectory can be formulated dimensionless in spatial coordinates, since both trajectory solutions are equivalent, aside from a scale factor.

Figures C-4 and C-5 quantitatively validate the preference of the QP approach over the Paparazzi approach, since the velocity, acceleration and jerk for the Paparazzi approach are almost always larger than in the QP approach. Moreover, the discontinuities are present, meaning that, when beginning and finishing a curve, there will be an instantaneous infinite acceleration and jerk.

Furthermore, Figures C-4 and C-5 also allow to see the influence of the dimension of the trajectory. The small and big trajectories are equivalent, apart from a scale of 10 in the dimension. For that reason, the difference in the acceleration and jerk between Figure C-4 and C-5 is a scale of 10 and 10^2 respectively.

C-2 Euler Angles

Due to the Differential Flatness property of the quadrotors, the trajectory has a direct correspondence with the Euler angles. The corresponding angles can can be seen in Figure C-6, where the time evolution of ϕ, θ and ψ is shown for the small and big trajectories, using a hard ($k_o = 0.2$) and a soft ($k_o = 1$) aggressiveness coefficient.

One can see that, in the Paparazzi approach, the angles in the steady state have a constant evolution, but during turns there is a big shift, almost instantaneous, to limit values. More-

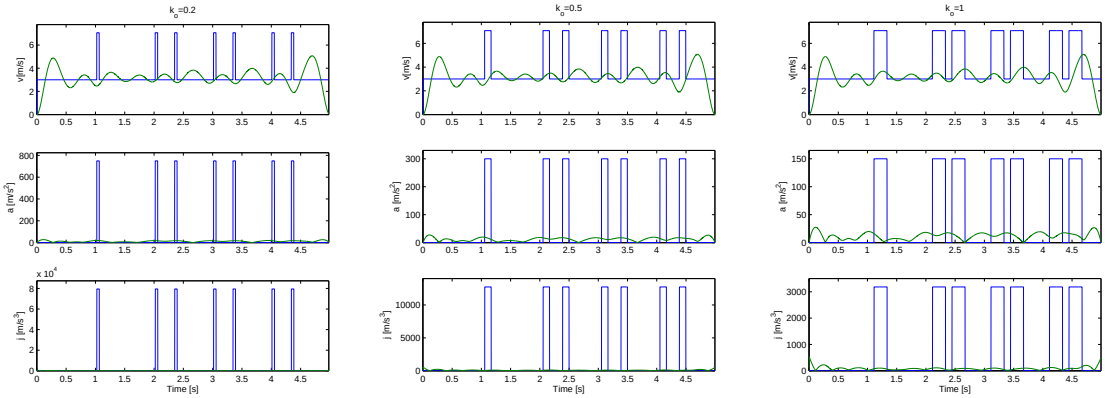


Figure C-4: Comparison between the velocity (top), acceleration (middle) and jerk (bottom) with the Paparazzi (blue) and QP (green) approaches for $k_o = 0.2$ (left), $k_o = 0.5$ (center), $k_o = 1$ (right) for the small trajectory.

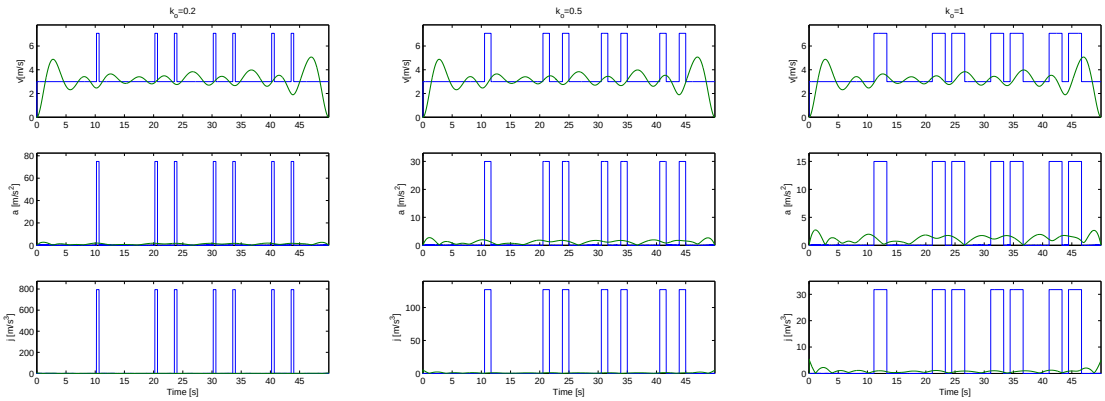


Figure C-5: Comparison between the velocity (top), acceleration (middle) and jerk (bottom) with the Paparazzi (blue) and QP (green) approaches for $k_o = 0.2$ (left), $k_o = 0.5$ (center), $k_o = 1$ (right) for the big trajectory.

over, the pitch angle shifts to ± 90 [^{o}] due to the high lateral accelerations, meaning that the quadrotor is supposed to turn completely sidewise, which obviously is not possible. In the QP approach, the evolution of the angles is smooth, which qualitatively validates the QP approach over the Paparazzi approach. Moreover, there is a usual tradeoff between the roll and pitch angles, all along the trajectory. Finally, there are still high values of pitch angle for the small trajectory (higher than 45 [^{o}]), which are not feasible in real life. However, the small trajectory is very ambitious, since it aims at turns of only 1 [m] at 3 [m/s].

Another important aspect is that in the full controller, the angles are not directly fed into the attitude stabilization loop. Instead, they are passed through a second order low pass filter, with saturations in the angular accelerations and rates, so that extreme angle changes are discarded. The results can be seen in Figure C-7.

It is possible to see that the narrow peaks are filtered, particularly in the small Paparazzi trajectory, where the reference angles are totally disfigured. The same happens for the big

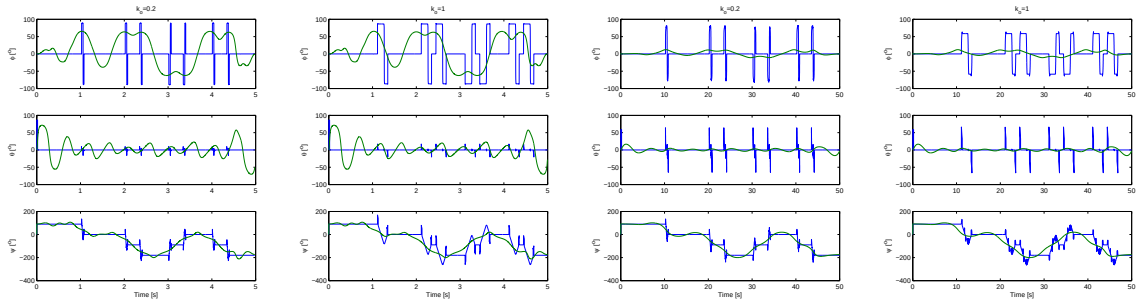


Figure C-6: Comparison between the Euler angles generated with the Paparazzi (blue) and QP (green) approaches for the small (two on the left) and big (two on the right) trajectories.

Paparazzi trajectory, but in a smaller scale. For the QP approach, the angles also suffer changes in the small trajectory, but usually implying a phase delay. For the big trajectory, one can see a perfect following.

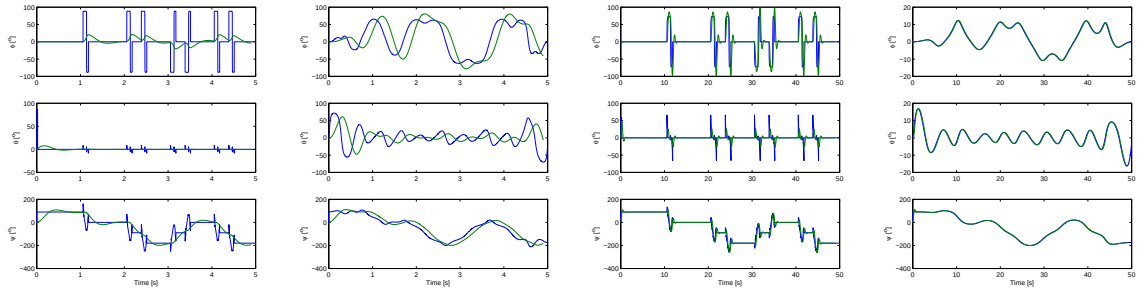


Figure C-7: Comparison between the Euler angles generated (blue) and followed (blue) for the Paparazzi/QP approaches with small/big trajectories. From left to right: Paparazzi small; QP small; Paparazzi big; QP big.

Appendix D

Quadratic Programming Structure

In this Appendix the structure of the Quadratic Programming (QP) problem is explained. The derivation will be made with focus on the specifications given in the paper, 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.

D-1 Trajectory Definition

Using differential flatness (Fliess, Lévine, Martin, & Rouchon, 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

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

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

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

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 (D-2) such that

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

and

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

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

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

D-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

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

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 (D-4).

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

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

with K_i a constant to turn the integral dimensionless. Since the trajectories are decoupled in all directions, the above Equation (D-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

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

and the **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.

D-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.

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

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

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

the jerk power can be obtained as

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

and the integral of the jerk power is given by

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

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

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

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. Thus, 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 (D-13) and $\mathbf{A}_{eq}$ as in Equation (D-14).

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

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

Appendix E

Identification of the Drag Coefficients

In this Appendix the flight tests that allowed 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 E-1 to E-4.

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 E-1.

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

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

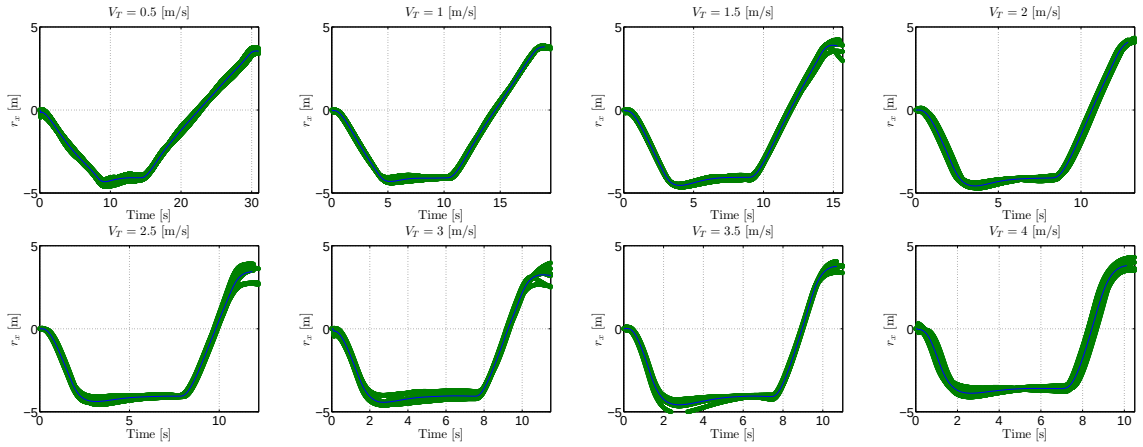


Figure E-1: Position.

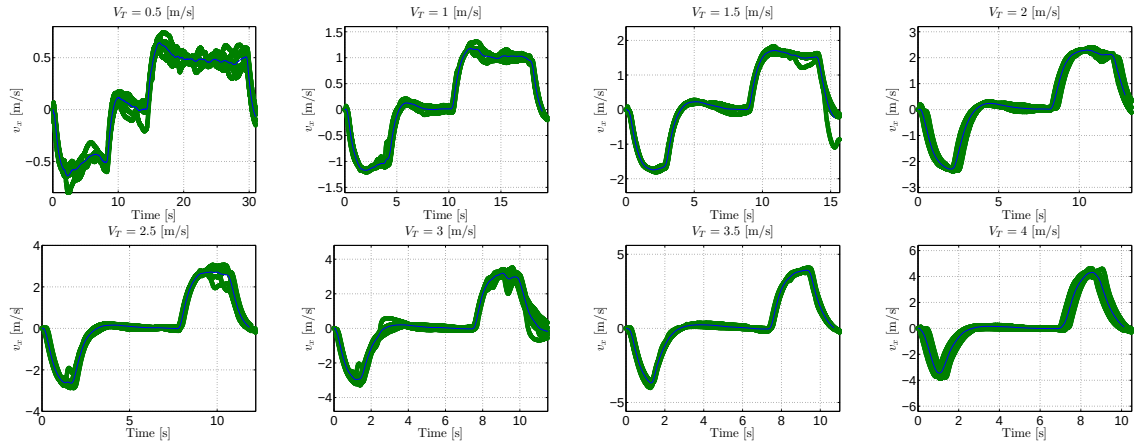


Figure E-2: Velocity.

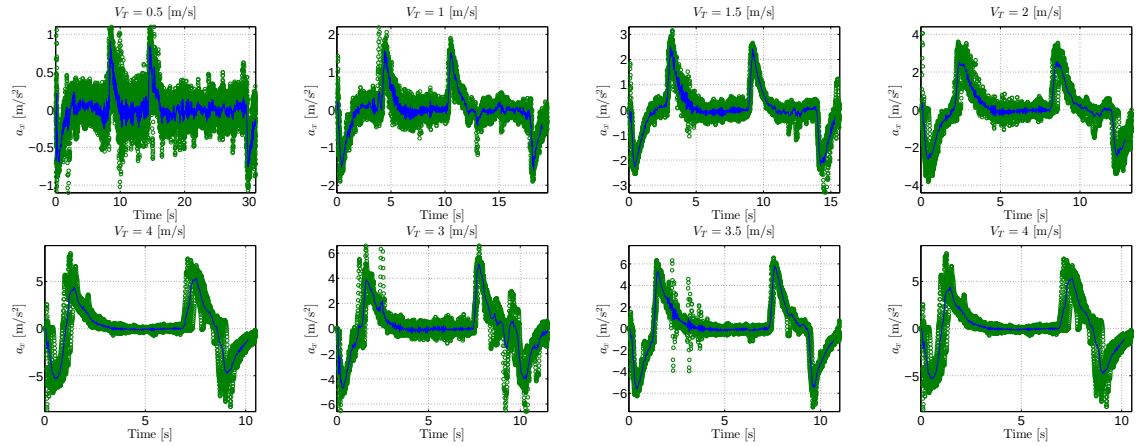


Figure E-3: Acceleration.

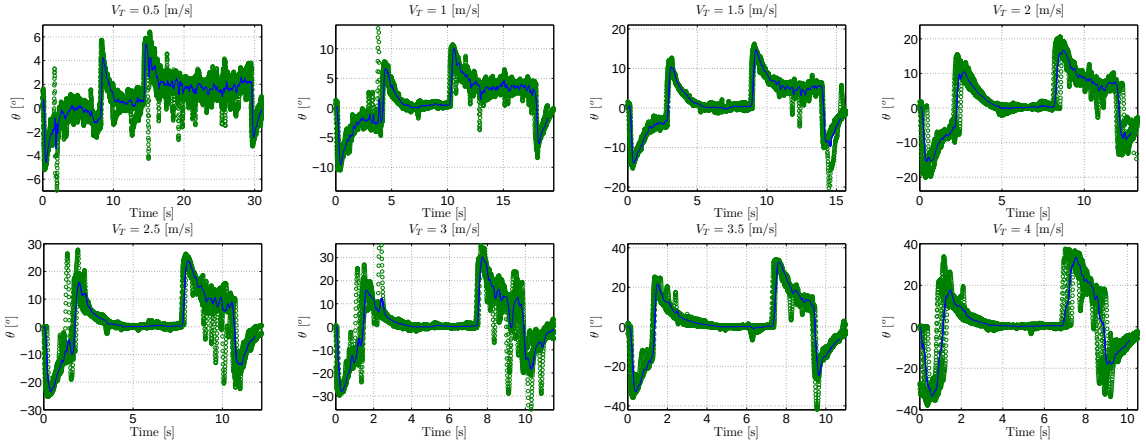


Figure E-4: Pitch angle.

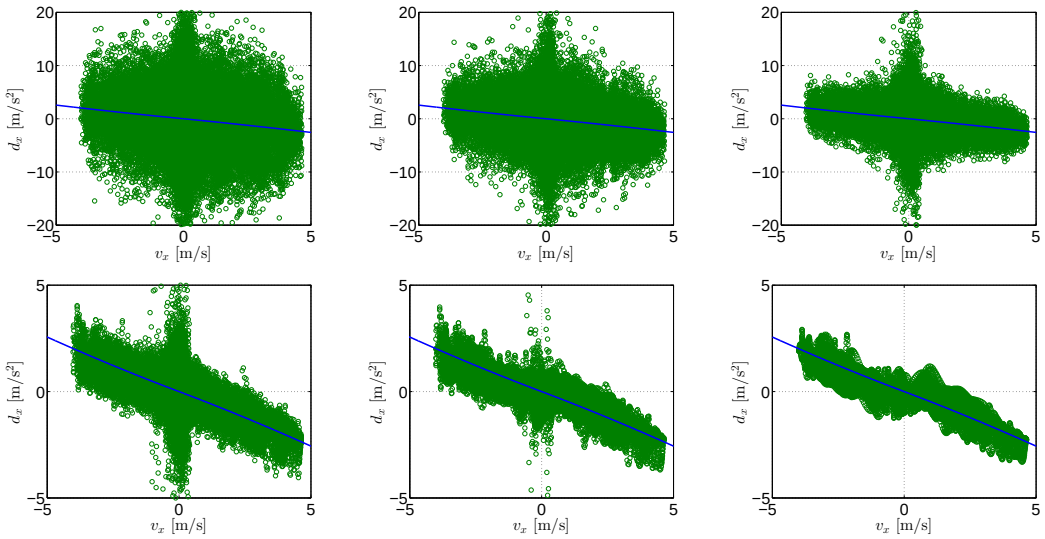


Figure E-5: Least squares fitting to obtain the drag coefficients. On the top with relatively small filtering, from left to right: no filtering, $\omega_n = 0.8$, $\omega_n = 0.6$. On the bottom with relatively big filtering, from left to right: $\omega_n = 0.3$, $\omega_n = 0.2$, $\omega_n = 0.1$.

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

Appendix F

Influence of the Lumped Drag Coefficient

In this Appendix the flight tests that allowed 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 F-1 to F-4. Figures F-1 and F-2 show the results when the drag coefficient is neglected, while Figures F-3 and F-4 show the results when it is accounted.

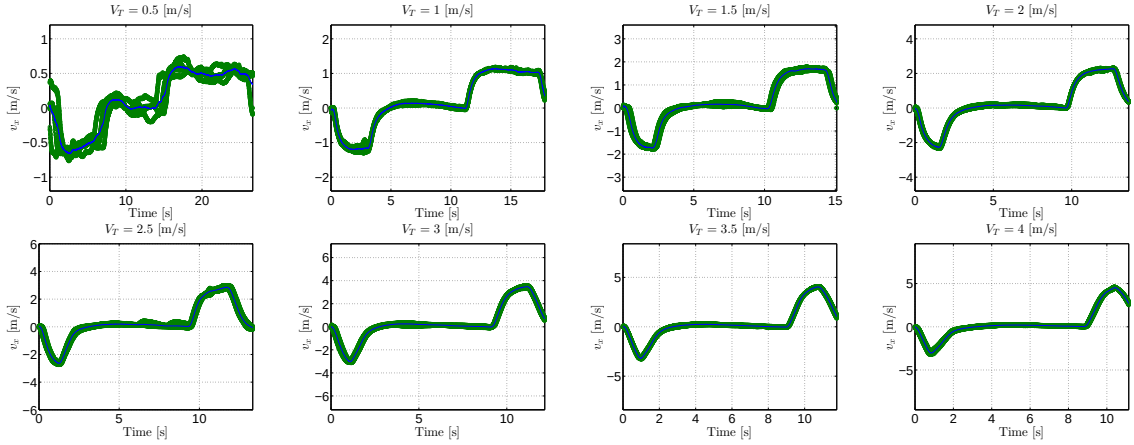


Figure F-1: Without drag coefficient - Velocity.

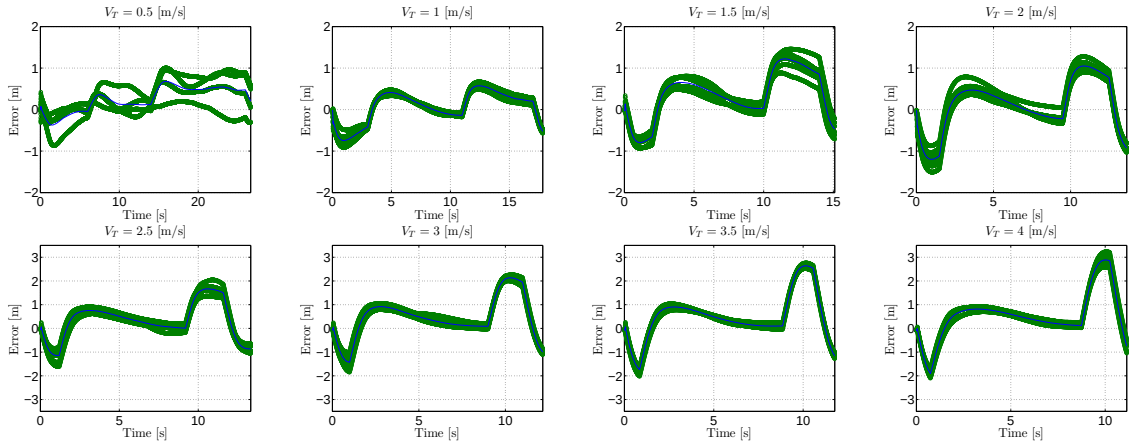


Figure F-2: Without drag coefficient - Position error.

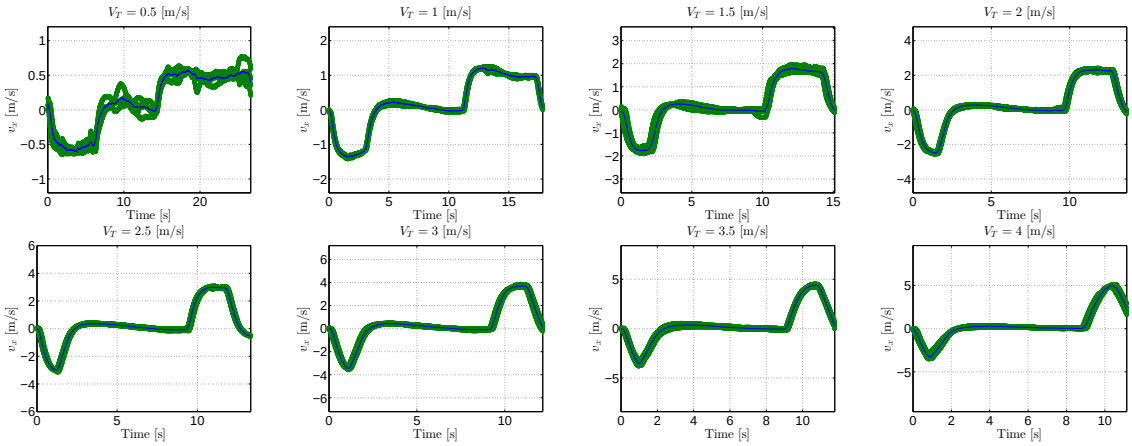


Figure F-3: With drag coefficient - Velocity.

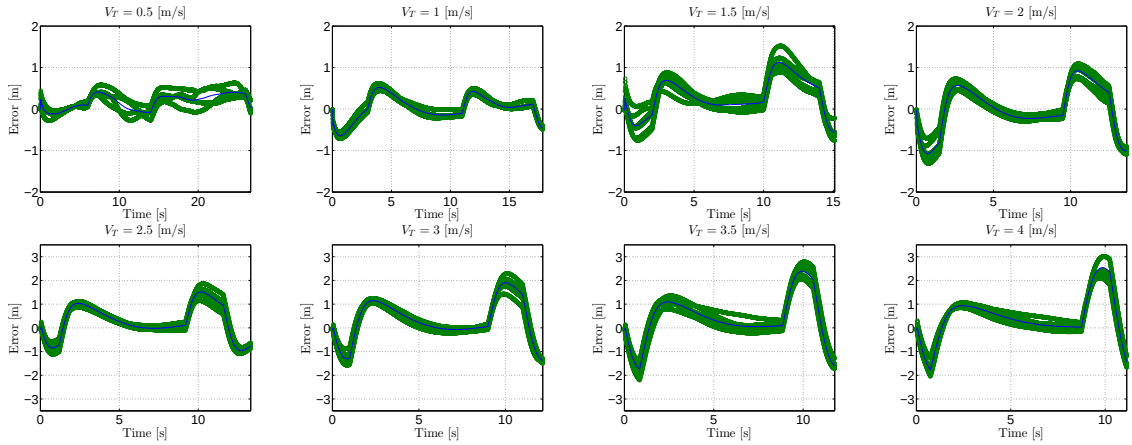


Figure F-4: With drag coefficient - Position error.

Appendix G

Reference Generator Comparison

In this Appendix the flight tests that allowed the comparison of the two implemented reference generators are shown. Both reference generators 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.

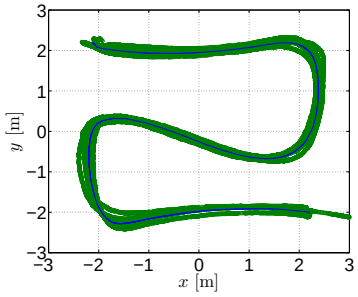


Figure G-1: Optimal - Position.

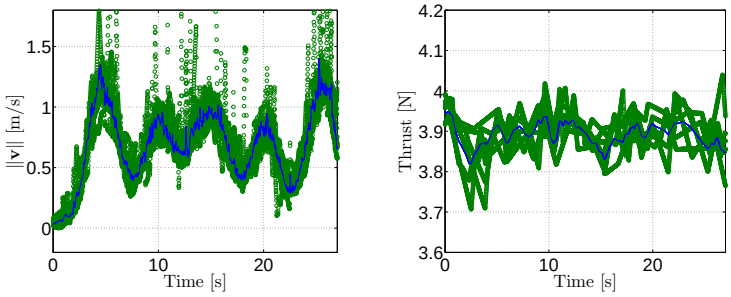


Figure G-2: Optimal - Velocity and thrust.

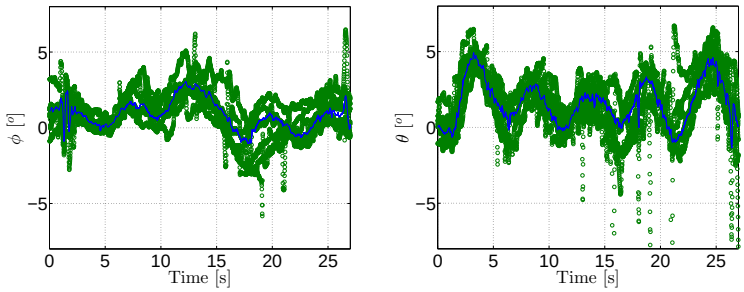


Figure G-3: Optimal - Euler angles.

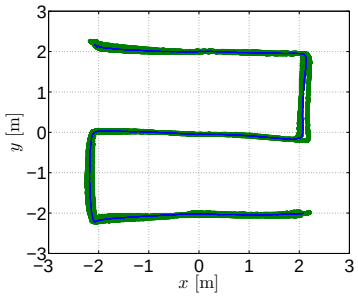


Figure G-4: Low-pass filter - Position.

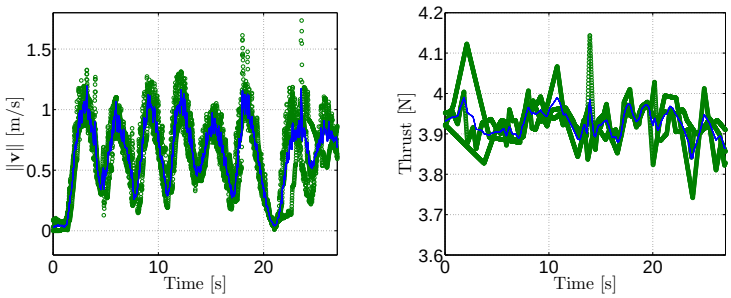


Figure G-5: Low-pass filter - Velocity and Thrust.

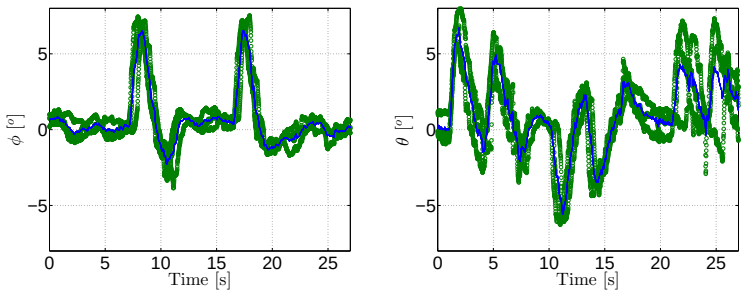


Figure G-6: Low-pass filter - Euler angles.

Appendix H

Controllers Comparison

In this Appendix the flight tests that allowed the comparison of the two implemented controllers are shown. 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.

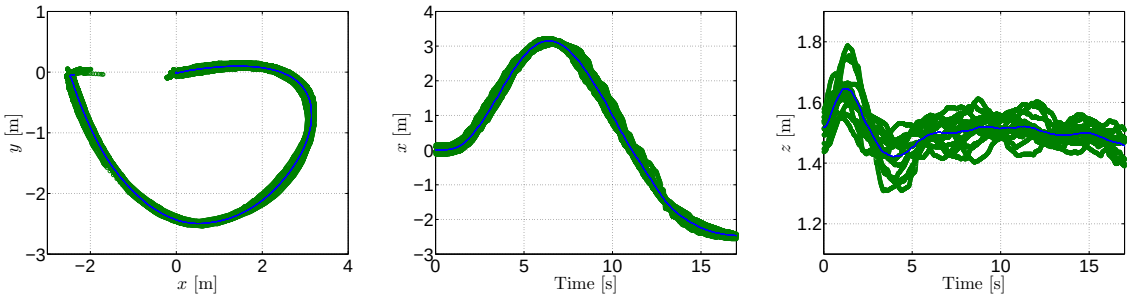


Figure H-1: Thrust vectoring controller, $V_{lim} = 1$ [m/s] - Position.

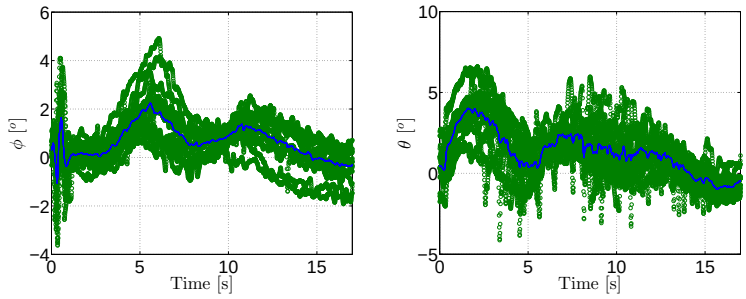


Figure H-2: Thrust vectoring controller, $V_{lim} = 1$ [m/s] - Euler angles.

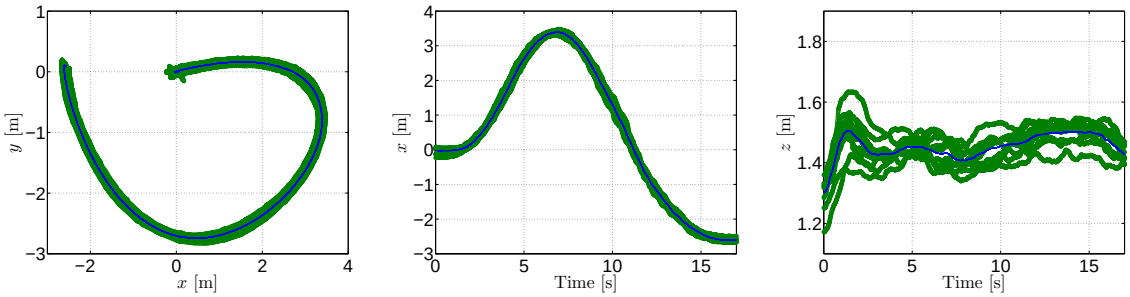


Figure H-3: Standard PID controller, $V_{lim} = 1$ [m/s] - Position.

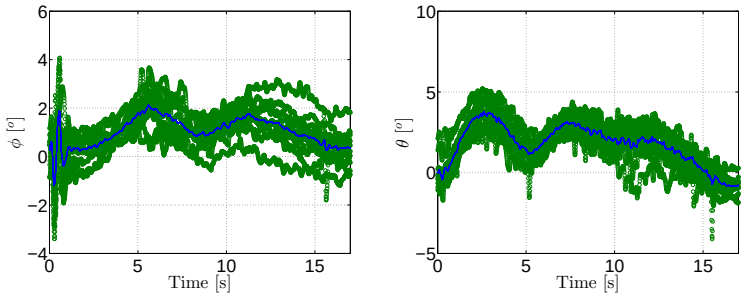


Figure H-4: Standard PID controller, $V_{lim} = 1$ [m/s] - Euler angles.

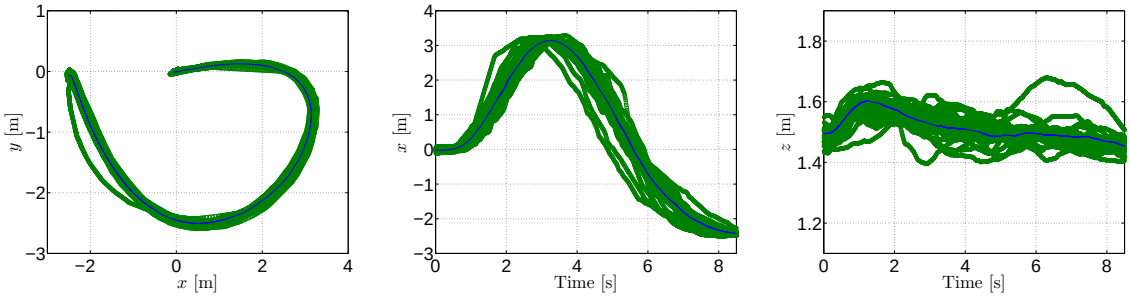


Figure H-5: Thrust vectoring controller, $V_{lim} = 2$ [m/s] - Position.

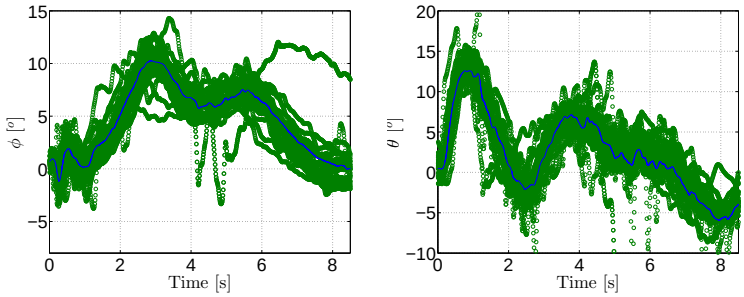


Figure H-6: Thrust vectoring controller, $V_{lim} = 2$ [m/s] - Euler angles.

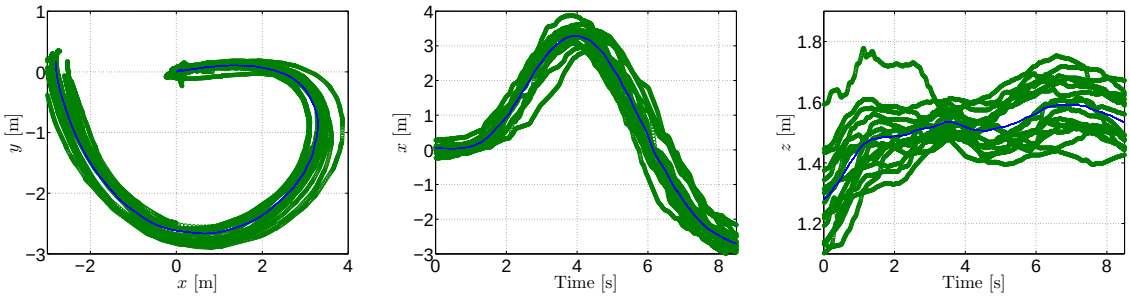


Figure H-7: Standard PID, $V_{lim} = 2$ [m/s] - Position.

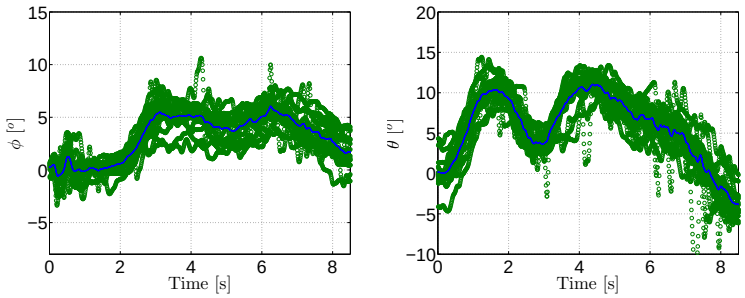


Figure H-8: Standard PID, $V_{lim} = 2$ [m/s] - Euler angles.

Appendix I

Optimal Sequencing Analysis

In this Appendix the results 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 I-1-I-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 I-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 I-4.

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

$\frac{V_w}{V_{lim}}$ [%]	m = 1			m = 2			m = 3			m = 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} [-]
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

$\frac{V_w}{V_{lim}}$ [%]	m = 5			m = 6			m = 7			m = 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} [-]
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 I-4: Heuristic h_2 - Trajectory time, CPU time and problem dimension.

$\frac{V_w}{V_{lim}}$ [%]	m = 1			m = 2			m = 3			m = 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} [-]
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

$\frac{V_w}{V_{lim}}$ [%]	m = 5			m = 6			m = 7			m = 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} [-]
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 I-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 I-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 I-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

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).

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

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

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).