Time-Optimal Control

for Tiny Quadcopters

J.M. Westenberger

Faculty of Aerospace Engineering

TU Delft

Time-Optimal Control

for Tiny Quadcopters

by

J.M. Westenberger

to obtain the degree of Master of Science at the Delft University of Technology, to be defended publicly on
Tuesday March 23, 2021 at 14:00

Student number:	4233425	
Project duration:	June, 2019 – March, 2021	
Thesis committee:	prof. dr. G.C.H.E. de Croon	Control and Simulation, TU Delft, supervisor
	ir. C. de Wagter	Control and Simulation, TU Delft, supervisor
	dr. E.J.J. Smeur	Control and Simulation, TU Delft
	dr. A. Menicucci	Space Engineering, TU Delft

TU Delft

Acknowledgements

I would like to start off by thanking my supervisors, Guido and Christophe, for their precious guidance and support. Thank you for giving me the freedom to explore my own ideas whilst making sure I stayed in the right direction.
Furthermore, I am immensely grateful for being given the opportunity to be part of the MAVLab Alphapilot team. The months in this team have given me a lot of insight on software development, embedded systems, deep learning and debugging. The level of proficiency, focus and perseverance that Christophe, Federico and Nilay displayed during the final weeks of the competition inspired me greatly.

I would also like to thank my fellow researchers of the MAVLab: Rohan, Nilay, Anoosh, Karan, Dennis and Bart for their friendship and support. Your continued presence and humor were invaluable to me and made me keep my motivation and high spirit in a time of a pandemic.

Finally, I would like thank my parents for their patience and my grandfather, whose lack of patience pushed me at the right moments.

J.M. Westenberger
Delft, March 2021

Abstract

Time-optimal model-predictive control is essential in achieving fast and adaptive quadcopter flight. Due to the limited computational performance of onboard hardware, aggressive flight approaches have relied on off-line trajectory optimization processes or non time-optimal methods. In this work we propose a computational efficient model predictive controller (MPC) that approaches time-optimal flight and runs onboard a consumer quadcopter. The proposed controller is built on the principle that constrained optimal control problems (OCPs) have a so-called 'bang-bang' solution. Our solution plans a bang-bang maneuver in the critical direction while aiming for a 'minimum-effort' approach in non-critical direction. Control parameters are computed by means of a bisection scheme using an analytical path prediction model. The controller has been compared with a classical PID controller and theoretical time-optimal trajectories in simulations. We identify the consequences of the OCP simplifications and propose a method to mitigate one of these effects. Finally, we have implemented the proposed controller onboard a consumer quadcopter and performed indoor flights to compare the controller's performance to a PID controller. Flight experiments have shown that the controller runs at 512hz onboard a Parrot Bebop quadcopter and is capable of fast, saturated flight, outperforming traditional PID controllers in waypoint-to-waypoint flight while requiring only minimal knowledge of the quadcopter's dynamics.

Contents

Acronyms

ADP	Adaptive/Approximate Dynamic Programming
ADR	Autonomous Drone Racing
AHRS	Attitude and Heading Reference System
AI	Artificial Intelligence
AIRR	Artifical Intelligence Robitic Racing
ANN	Artificial Neural Network
BVP	Boundary Value Problem
CNN	Convolutional Neural Network
DRL	Drone Racing League
EKF	Extended Kalman Filter
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDP	Heuristic Dynamic Programming
HJB	Hamilton-Jacobi-Bellman Equation
IEKF	Iterated Extended Kalman Filter
IMAV	International Micro Air Vehicles, Conferences and Competitions
IMU	Inertial Measurement Unit
INDI	Incremental Nonlinear Dynamic Inversion
IROS	International Robots and Systems
LQR	Linear Quadratic Regulator
MDP	Markov Decision Process
MPC	Model Predictive Control
NDI	Nonlinear Dynamic Inversion
NLP	Nonlinear Programming
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PID	Proportional-Integral-Derivative
PMP	Pontryagin's Minimum Principle
PnP	Perspective-n-point
QP	Quadratic Programming
RANSAC	Random Sample Consensus
RRT	Rapidly-exploring Random Tree

SE-SCP	Spherical Expansion and Sequential Convex Programming
SQP	Sequential Quadratic Programming
UAV	Unmanned Aerial Vehicle
V-SLAM	Visual Localization and Mapping
VIO	Visual Inertial Odometry
VO	Visual Odometry

List of Symbols

B	Input Matrix
C_d	Drag Coefficient
$\mathbf{e}_{\cdot}$	Error
$f(\cdot)$	System Dynamics Function
g	gravity (9.81 m/s^2)
$\mathcal{H}$	Hamiltonian
$J(\cdot)$	Cost Function
$K_{\cdot}$	Gain
m	Mass
$\mathbf{p}$	Co-state Vector
$\Phi(\cdot)$	Switching Function
ϕ	Roll Angle
p	Lagrangian Multiplier
x	Position
z	Altitude
Q	Cost Matrix
R	Cost Matrix
T	Thrust Force
t	Time
t_f	Final Time
θ	Pitch Angle
$\mathbf{u}$	Input Vector
W	Weight
$\mathbf{x}$	State Vector
$\mathbf{x}_r$	Reference Trajectory

List of Figures

List of Tables

1 Introduction

1.1. Motivation

Drone racing has been on the rise in the recent years. Every year the best pilots compete against each other and winners walk away with prices of hundred-thousands of dollars. During these events the pilots have to steer quadcopters through gates that are distributed along a complex racing track. The pilots wear first-person-view video-goggles that are connected to the on-board camera of the quadcopter so that they can quickly react whilst flying at speeds of over 140 km/h [1].
Simultaneously, new innovations are introduced in the autonomous sector of drones at an accelerating rate.
However, the recent Alphapilot 2019 challenge has shown that there is still a large gap in performance between human racing pilots and autonomous controllers. Human pilots still excel in being adaptive and improvising quickly to unsuspected events.
In this challenge Team MAVLab, from the TU Delft, won the competition against other automated drones by finishing a track in 11 seconds, flying fully autonomously. However, in a bonus round against a human pilot, Gab707, one of the world's best racing pilots finished the same track manually controlling the same drone in only 6 seconds [2].

Advancements in control theory show promise that the performance gap between human pilots and autonomous drones can be overcome in the near future. And despite that Bellman [3] and Pontryagin [4] have developed the mathematical foundation for optimal control theory in the 1950s, applying these methods to real-life scenarios are often so computationally expensive that, even today, the required hardware would be too heavy and power-hungry to be implemented onboard a small quadcopter. The complete approach has to address accurate state estimation, expensive numerical solvers and generation of feasible flight trajectories. Much of the relevant research had to rely on simulations, or off-board and off-line computations, as will be discussed in chapter 3. While for autonomous drone applications it is desirable to execute all computations on-board.

On the other hand, extensive research is performed on tiny, pocket-sized quadcopters and their on-board processing capabilities on the road to high-performance autonomous control. Currently, the objective in this field mostly comprises autonomous navigation, obstacle avoidance and swarming. Research in on-board time-optimal control solutions often result in machine learning approaches that require large datasets and significant training time. The ambition of this project is to combine fast time-optimal control with lightweight, (computational) power-limited quadcopters. Research has been performed in the fields of drone racing and optimal control theory and investigate how it can be applied in conjunction to quadcopters. The goal is to develop an on-board solution of a time-optimal controller which is able to run in real-time on a small quadcopter, demonstrating independence of external computers.

1.2. Objective & Research questions

Based on the arguments of section 1.1 the main objective of this project is:

to make a *small quadcopter* navigate through a course of positional waypoints in a *time-optimal fashion*, by developing a *controller that can run real-time and on-board*.

From which the following research questions are derived:

1. Is the complete control system computationally efficient enough to run onboard a power-limited microcontroller?
2. How well does the control system approach true time-optimality?

The first question is important to demonstrate that the solution can run in real-time onboard of the drone. However, in order to do so it is expected that some simplifications and assumptions will have to be chosen which may cause the solution to deviate from the theoretical time-optimal solution. Furthermore, other in-flight phenomena may affect the flight behavior to an unexpected level. These so-called 'reality gaps' must be identified and it must be investigated how they can be mitigated in order to answer the second research question.

1.3. Planning

The thesis project consists of two phases: the preliminary research phase and the main thesis phase.

In the preliminary research phase work is done to:

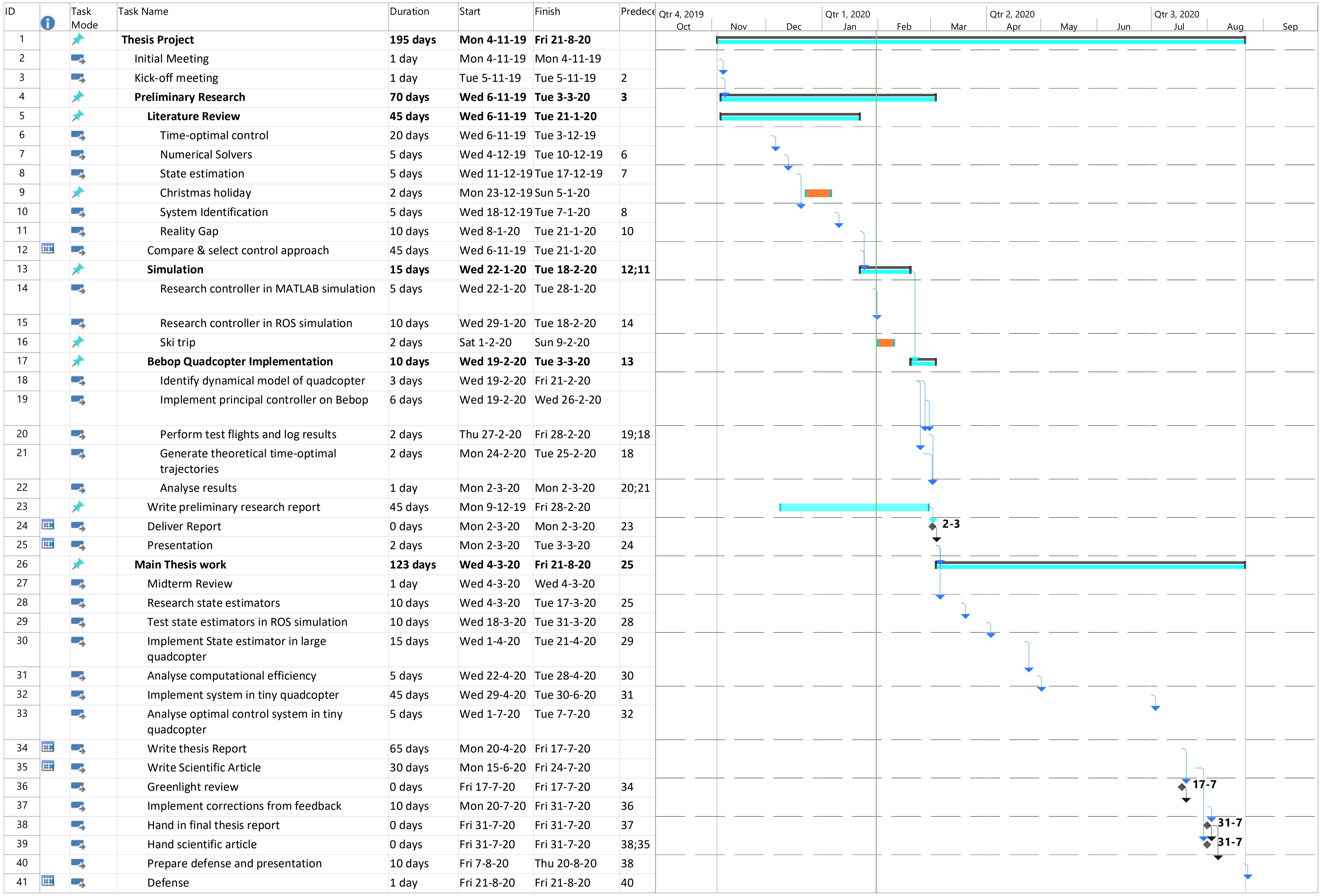
1. Perform a literature review to investigate more thoroughly research performed on optimal control formulations,solutions and practical implementations of optimal controllers.
2. Select which approach the thesis will take.
3. Demonstrate principal optimal controller in simulation.
4. Demonstrate principal optimal controller on a real quadcopter

In this context the principal optimal controller consists of only the system that is responsible for solving the optimal control problem and generating the control inputs. That is, subsystems that deal with state estimation and perception are not regarded yet. The preliminary phase is concluded with this document.

The main thesis phase will focus on working towards a complete practical implementation of the optimal controller on a tiny quadcopter. Which includes analyzing and minimizing the 'reality-gap' between simulation and real flight, completing the optimal controller with required systems such as perception and state estimation. Finally, a working solution shall be demonstrated and its performance will be compared to the theoretical time-optimal flight. The main thesis phase is concluded with a scientific paper and the thesis defense.

A detailed Gantt chart of the chronological planning of the thesis project is presented in subsection 1.3.1. It takes into account a 5 day work week and 3 weeks of holidays, giving a total active project time of 9 months.

1.3.1. Gantt Chart



ID	ⓘ	Task Mode	Task Name	Duration	Start	Finish	Predece
1		📌	**Thesis Project**	**195 days**	**Mon 4-11-19**	**Fri 21-8-20**	
2		➡	Initial Meeting	1 day	Mon 4-11-19	Mon 4-11-19	
3		➡	Kick-off meeting	1 day	Tue 5-11-19	Tue 5-11-19	2
4		📌	**Preliminary Research**	**70 days**	**Wed 6-11-19**	**Tue 3-3-20**	**3**
5		📌	**Literature Review**	**45 days**	**Wed 6-11-19**	**Tue 21-1-20**	
6		➡	Time-optimal control	20 days	Wed 6-11-19	Tue 3-12-19	
7		➡	Numerical Solvers	5 days	Wed 4-12-19	Tue 10-12-19	6
8		➡	State estimation	5 days	Wed 11-12-19	Tue 17-12-19	7
9		📌	Christmas holiday	2 days	Mon 23-12-19	Sun 5-1-20	
10		➡	System Identification	5 days	Wed 18-12-19	Tue 7-1-20	8
11		➡	Reality Gap	10 days	Wed 8-1-20	Tue 21-1-20	10
12	▦	➡	Compare & select control approach	45 days	Wed 6-11-19	Tue 21-1-20	
13		📌	**Simulation**	**15 days**	**Wed 22-1-20**	**Tue 18-2-20**	**12;11**
14		➡	Research controller in MATLAB simulation	5 days	Wed 22-1-20	Tue 28-1-20	
15		➡	Research controller in ROS simulation	10 days	Wed 29-1-20	Tue 18-2-20	14
16		📌	Ski trip	2 days	Sat 1-2-20	Sun 9-2-20	
17		📌	**Bebop Quadcopter Implementation**	**10 days**	**Wed 19-2-20**	**Tue 3-3-20**	**13**
18		➡	Identify dynamical model of quadcopter	3 days	Wed 19-2-20	Fri 21-2-20	
19		➡	Implement principal controller on Bebop	6 days	Wed 19-2-20	Wed 26-2-20	
20		➡	Perform test flights and log results	2 days	Thu 27-2-20	Fri 28-2-20	19;18
21		➡	Generate theoretical time-optimal trajectories	2 days	Mon 24-2-20	Tue 25-2-20	18
22		➡	Analyse results	1 day	Mon 2-3-20	Mon 2-3-20	20;21
23		📌	Write preliminary research report	45 days	Mon 9-12-19	Fri 28-2-20	
24	▦	➡	Deliver Report	0 days	Mon 2-3-20	Mon 2-3-20	23
25	▦	➡	Presentation	2 days	Mon 2-3-20	Tue 3-3-20	24
26		📌	**Main Thesis work**	**123 days**	**Wed 4-3-20**	**Fri 21-8-20**	**25**
27		➡	Midterm Review	1 day	Wed 4-3-20	Wed 4-3-20	
28		➡	Research state estimators	10 days	Wed 4-3-20	Tue 17-3-20	25
29		➡	Test state estimators in ROS simulation	10 days	Wed 18-3-20	Tue 31-3-20	28
30		➡	Implement State estimator in large quadcopter	15 days	Wed 1-4-20	Tue 21-4-20	29
31		➡	Analyse computational efficiency	5 days	Wed 22-4-20	Tue 28-4-20	30
32		➡	Implement system in tiny quadcopter	45 days	Wed 29-4-20	Tue 30-6-20	31
33		➡	Analyse optimal control system in tiny quadcopter	5 days	Wed 1-7-20	Tue 7-7-20	32
34	▦	➡	Write thesis Report	65 days	Mon 20-4-20	Fri 17-7-20	
35	▦	➡	Write Scientific Article	30 days	Mon 15-6-20	Fri 24-7-20	
36		➡	Greenlight review	0 days	Fri 17-7-20	Fri 17-7-20	34
37		➡	Implement corrections from feedback	10 days	Mon 20-7-20	Fri 31-7-20	36
38		➡	Hand in final thesis report	0 days	Fri 31-7-20	Fri 31-7-20	37
39		➡	Hand scientific article	0 days	Fri 31-7-20	Fri 31-7-20	38;35
40		➡	Prepare defense and presentation	10 days	Fri 7-8-20	Thu 20-8-20	38
41	▦	➡	Defense	1 day	Fri 21-8-20	Fri 21-8-20	40

Page 1

1.4. Document Structure

The document is divided in three main parts. A scientific paper in Part I, the literature review in Part II. Preliminary results of the thesis are presented in Part III.

1.4.1. Scientific Paper

The scientific paper presents all contributions of this thesis. The paper can be read as a standalone document. It gives a brief introduction to optimal control and related work before explaining the main thesis work and the setup and results of performed experiments. Additionally, some tests have performed that are not covered in the paper. These are briefly described in Appendix C.

1.4.2. Literature Review

The literature review is performed to investigate the relevant scientific work that can aid in solving the main objective of this thesis.
Consequently, the literature review starts with an introduction to the mathematical side of optimal control theory in chapter 2. It explains how the work of Bellman and Pontryagin have been crucial to defining and solving optimal control problems.
Secondly, in chapter 3 a review is given of relevant work on the most important quadcopter subsystems that make autonomous drone racing possible. It summarizes work on trajectory generation methods, quadcopter control, perception and state estimation.

1.4.3. Preliminary Results

The final part of this document consists of presenting the preliminary results of the thesis. Based on the literature review a solution to time-optimal control is suggested. It explains the design of a flight controller and which assumptions/simplifications were made for the benefit of computational efficiency. Furthermore, this controller has been demonstrated in simulations environments from which the results are analyzed. It is investigated to which extend this controller approaches time-optimal control. And what decisions have been made so-far to mitigate the disadvantageous effects of the simplifications.

1.4.4. Flight Experiment Results

The scientific paper describes the most relevant flight experiments and discusses their results. However, more flight experiments have been performed that have not been covered in the paper due to inconclusive results or incompleteness. For completeness, all flight experiments have been summarized in Appendix C.

Part I

Scientific Paper

Time-Optimal Control for Tiny Quadcopters

Jelle Westenberger, Guido C.H.E. de Croon, Christophe de Wagter

Abstract*—Time-optimal model-predictive control is essential in achieving fast and adaptive quadcopter flight. Due to the limited computational performance of onboard hardware, aggressive flight approaches have relied on off-line trajectory optimization processes or non time-optimal methods. In this work we propose a computational efficient model predictive controller (MPC) that approaches time-optimal flight and runs onboard a consumer quadcopter. The proposed controller is built on the principle that constrained optimal control problems (OCPs) have a so-called 'bang-bang' solution. Our solution plans a bang-bang maneuver in the critical direction while aiming for a 'minimum-effort' approach in non-critical direction. Control parameters are computed by means of a bisection scheme using an analytical path prediction model. The controller has been compared with a classical PID controller and theoretical time-optimal trajectories in simulations. We identify the consequences of the OCP simplifications and propose a method to mitigate one of these effects. Finally, we have implemented the proposed controller onboard a consumer quadcopter and performed indoor flights to compare the controller's performance to a PID controller. Flight experiments have shown that the controller runs at 512hz onboard a Parrot Bebop quadcopter and is capable of fast, saturated flight, outperforming traditional PID controllers in waypoint-to-waypoint flight while requiring only minimal knowledge of the quadcopter's dynamics.

***Index Terms*—time-optimal control, model-predictive control, MAVs**

I. Introduction

DRONE racing has been gaining popularity in recent years. In addition to human-piloted drone race competitions there are also events taking place that challenge the autonomous field of quadcopter control . Autonomous drone racing can be considered to be the most demanding test case of quadrotor control systems. The desired aggressive flights and adaptiveness puts all on-board systems to the limit. From state estimation and trajectory generation systems to trajectory tracking and inner-loop controllers [1]–[3].

Methods have been developed to automatically generate optimal trajectories. One of the most popular approaches is optimizing a trajectory for minimum-snap which improves the overall "smoothness" of the path [4], [5]. This is known to be beneficial to the measurement quality of the onboard sensors. Furthermore, the resulting trajectory formulation allows for convenient extraction of control inputs, using differential flatness, which greatly improves trajectory tracking performance when used as feedforward terms in a controller [4], [6]. Optimizing a trajectory polynomial for minimum-snap is an efficient process if the total flight time is predefined and dynamic feasibility is not included in the optimization problem. When the dynamic feasibility and total flight time constraints are considered, the computational cost increases significantly [5].

In optimal-control theory there are generally two approaches considered in finding the mathematical time-optimal solutions. The first deals with solving the Hamilton-Jacobi-Bellman equation (HJB) equation, which in essence searches the entire state-space of the system for the optimal sequence of states and control inputs. Such a problem can be solved recursively by using dynamic programming methods, or it can be solved by transforming the problem to a nonlinear programming (NLP) formulation and solving it by using numerical solver methods such as collocation or shooting [7].

The second approach, also known as the indirect method, attempts to find the time-optimal control solution by finding the state and control values for which the conditions for optimality are satisfied, as stated by Pontryagin's Minimum Principle (PMP). However, this approach does not guarantee that the optimal solution is at a global optimum [8].

Unfortunately, solving time-optimal principles has shown to be a computationally demanding process. And especially in a dynamic environment it is desired implement a path predictor to account for obstacles and other disturbances, which requires frequent re-calculations of the optimal trajectory. Therefore, most work consist of calculating the trajectory off-board on computers, and let the on-board systems only deal with the trajectory tracking [4], [9], [10].

To address this, work has been performed to approximate optimal-control with deep neural nets. [11]–[13]. These methods are less computationally demanding and can be executed onboard, but require a large dataset and considerable training time.

The work presented in this paper shows how time-optimal flight can be approached with a simplified OCP that is light enough to be solved onboard and can be implemented as an MPC. The suggested approach is based the knowledge that time-optimal control problems with constrained inputs have a so-called 'bang-bang' shape [8], [14]. This fact reduces the optimal control problem (OCP) to one in which only the time instances at which the control values switch from one saturated value to the other saturated value need to be optimized. We have developed a model predictive controller (MPC) that optimizes the switching time of a 'bang-bang' motion in one critical direction. The noncritical direction takes on a so-called 'minimum-effort' approach in which a constant attitude angle is optimized for minimum required acceleration. Thanks to a few simplifying assumptions this method only requires a minimal amount of function evaluations because the trajectories are described analytically. Making this method computational efficient enough to run onboard a small quadcopter.

Section II shows that for our simplified OCP the solution consists of a bang-bang input in attitude. In Section III we derive the prediction model of the proposed MPC and elaborate on how the control parameters are optimized. The MPC is tested in simulation and compared to trajectories from other approached in Section IV. Section V introduces a method to mitigate the effects of latency in attitude. The proposed has been implemented on a consumer quadcopter. The flight experiments are described in Section VI. Finally, the results are discussed in Section VII and a conclusion is given in Section VIII.

II. Simplified Optimal Control

In order to reduce the computational effort a more simplified approach to optimal control must be taken. It is known that for constrained OCPs the solution consist of a control input with a bang-bang or bang-singular-bang shape. We take advantage of this knowledge by assuming beforehand that the quadcopter's attitude has a bang-bang shape as well. [8] proves that the optimal control solution to a two dimensional quadcopter system is in fact bang-bang for thrust and bang-singular-bang for angular rate. However, as is shown in Appendix A, an analytical solution to the path prediction can only be found if the attitude angle and thrust is considered to be constant. This implies that angular rate cannot be taken into account in the analytical path prediction.
In Appendix B we show that for our model simplifications the attitude control input has a bang-bang shape. The altitude is assumed to be constant and therefore the constant thrust value will be governed by this condition.

III. Bang-Bang MPC

A model-predictive controller has been implemented in which the prediction assumes that the quadcopter's trajectory from its current position to its target consists of a bang-bang motion in either the longitudinal or lateral direction. We call this the 'critical' direction. That is, it will accelerate at an maximum attitude angle (pitch or roll), and subsequently decelerate at a maximum angle to reach the target with a specific desired velocity. Additionally, the remaining direction takes on a so-called 'minimum-effort' approach. In this direction the optimizer calculates the optimal constant angle such that the quadcopter reaches the noncritical target position at the same time as it reaches the critical target. In this way most available thrust will be utilized to minimize the critical position error as quickly as possible.

In practice, the critical and noncritical direction can be interchanged, as is also demonstrated in flight experiments described Section VI. However, for convenience, we assume for now that the critical dimension is always in the longitudinal direction w.r.t. the quadcopter's body. Therefore, the predicted maneuver will consist of a bang-bang motion in pitch angle and of a constant angle in roll. For the longitudinal motion the parameter to be optimized is the time instant at which the quadcopter switches from accelerating to decelerating, while for the lateral motion the parameter to be optimized will be the roll angle.

A. Path Prediction

A simple decoupled 2nd order model has been derived to predict the motion of the quadcopter. The derivation is shown in Appendix A. The longitudinal and lateral dynamics have been decoupled in this model. Consequently, the longitudinal and lateral components of the path are predicted individually. Because our optimal control approach is based on a 2-D model the quadcopter's heading is considered constant during the maneuvers. Furthermore, the vertical dynamics are omitted by assuming perfect altitude control. This implies that it is assumed that the collective thrust perfectly compensates for the quadcopter's weight. That is, the earth Z-component of the thrust is the opposite value of the quadcopters weight so that there is no acceleration in altitude. The collective thrust force is therefore always $W/(\cos\theta\cos\phi)$. Where W is the quadcopter's weight ($W = m\cdot g$), θ and ϕ are the pitch and roll angles, respectively. Moreover, the aerodynamic drag is assumed to be linearly proportional to, and acting against the quadcopters speed: $F_{\text{A}} = C_d \cdot V$. Where C_d is the drag coefficient.
So for the longitudinal direction the motion is described by:

$$x = c_1 e^{\frac{-C_d}{m}t} + c_2 + \frac{W\tan\theta}{C_d}t \tag{1a}$$

$$\dot{x} = \frac{-C_d}{m} c_1 e^{\frac{-C_d}{m}t} + \frac{W\tan\theta}{C_d} \tag{1b}$$

Where c_1 and c_2 are calculated by solving the equations for the initial position and velocity as can be seen in Appendix A.
The lateral direction is described by:

$$y = c_3 e^{\frac{-C_d}{m}t} + c_4 + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d}t \tag{2a}$$

$$\dot{y} = c_3 \frac{-C_d}{m} e^{\frac{-C_d}{m}t} + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d} \tag{2b}$$

Figure 1 shows a typical trajectory prediction when starting from rest and flying towards a target that is 10 meters ahead and 5 meters sideways. Note that the longitudinal trajectory consists of an acceleration and a braking segment while the lateral trajectory is a single segment with a constant roll angle. The longitudinal and lateral target are reached simultaneously.

Bang-Bang Prediction

As aforementioned, the longitudinal direction is considered to be a bang-bang motion. The trajectory is calculated by splitting it in two sections. First, the acceleration phase up to the switching time. Secondly, the braking phase from switching time up to the target. It is assumed that the attitude change is achieved instantly.

Lateral Prediction

The lateral direction does not consist of a bang-bang motion, but rather of a single, constant angle acceleration. So contrary to the bang-bang motion, this trajectory only consists of a single segment.

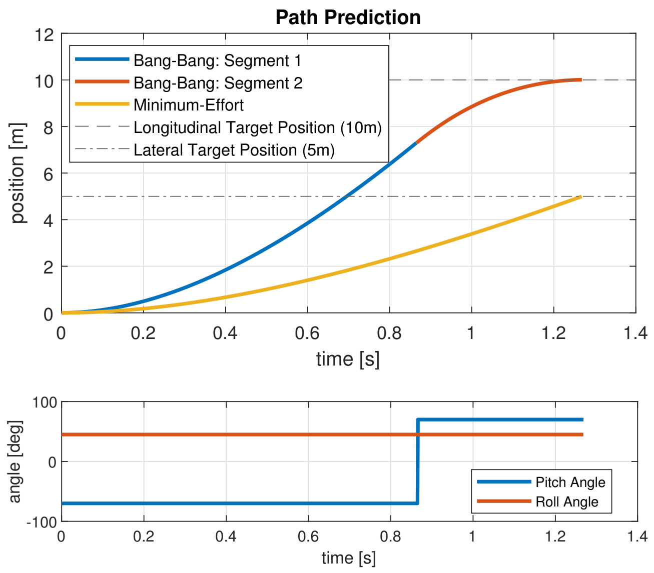


Fig. 1. Predicted trajectory and corresponding attitude angles.

B. *Solving Optimal Control*

The control optimization differs between the critical and noncritical directions. As aforementioned, the critical direction consists of a bang-bang maneuver, and therefore the switching time is the parameter-to-be-optimized. Since the noncritical direction consists of a constant angle maneuver, the parameter to be optimized for the noncritical motion is the roll angle.

a) Saturation Dimension Solver: For the critical dimension an optimal switching time must be found. The optimization procedure is based on reaching a specified desired speed at the target location. Because the prediction model is a discontinuous analytical equation,the maneuver can be described completely by only evaluating Equation 1 at the target position and at the switching instant to find the initial conditions for the second segment.

A bisection scheme has been implemented to iteratively adapt the switching time to minimize the velocity error at the target position. The scheme has been summarized in Algorithm 1. In Figure 2 the resulting trajectories of an optimization process can be seen. In this case the quadcopter is desired to have a desired velocity of 0 at a position of 30 meters ahead. The vertical dashed lines indicate the different switching times which are seen to converge to a value of 1.9 seconds.

b) Lateral Dimension Solver: For the lateral dimension the corresponding attitude angle is optimized. The scheme used for this is similar to Algorithm 1. However, instead of switching time the attitude angle is optimized. The predicted trajectory is no longer of the bang-bang type but only consists of a single segment corresponding with a constant attitude angle. The error value is determined by assessing the position of the lateral trajectory at the time of arrival that was found in the optimization of the longitudinal path.

The benefit of this approach is that the lateral error goes to zero at the same time as the longitudinal position error,

Algorithm 1

```
$t_0 \leftarrow 0$
$t_1 \leftarrow$ initial guess
$E_t \leftarrow$ error threshold
$y_d \leftarrow$ desired position
**while** $E > E_T$ **do**
    $t_s \leftarrow \frac{t_0+t_1}{2}$
    $t_t \leftarrow get\_time\_from\_desired\_speed(v_d)$
    $E \leftarrow get\_position(t_t) - y_d$
    **if** $E > 0$ **then**
        $t_1 \leftarrow t_s$
    **else**
        $t_0 \leftarrow t_s$
    **end if**
**end while**
```

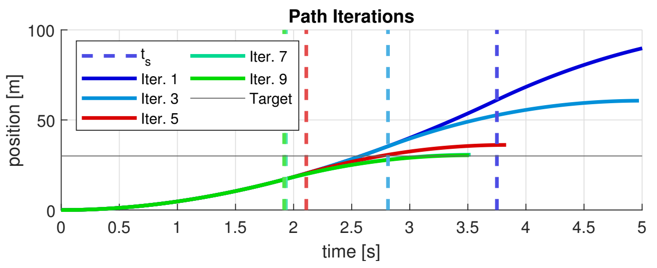


Fig. 2. Illustration of the switching time iterations and their corresponding trajectories

suggesting that no energy is wasted to by reaching the lateral target position sooner than necessary. The disadvantage is that no path constraints can be taken into account and that it is not possible to optimize for a desired lateral speed at the target.

IV. Simulations

Before applying the method to a real quadcopter, the method is tested in simulations first. Simulations have been performed in MATLAB [15], for testing the method's feasibility, and in Flightgoggles [16], for higher fidelity testing. Furthermore, to compare the results with 'true' time-optimal control the ICLOCS has been used to create time-optimal trajectories based on direct-collocation.

A. *Matlab Simulations*

A relatively simple simulator has been set up in Matlab which acted as a development platform for the suggested optimal controller. The dynamical model is almost identical to the model from which the path predictor is derived (see Appendix A). However, in contrast to the path predictor model the simulator model does include rotational dynamics by approximating the rotational acceleration as a first order time delay. The entire state-space system is described by:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} & \dot{\mathbf{r}} & \boldsymbol{\Theta} & \boldsymbol{\Omega} \end{bmatrix}^T$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_{\boldsymbol{\Omega}} & u_T \end{bmatrix}^T$$

$$f(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \\ \dot{\boldsymbol{\Theta}} \\ \dot{\boldsymbol{\Omega}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}} \\ \frac{1}{m}\left[\mathbf{R}_{E|B} \begin{bmatrix} 0 \\ 0 \\ -u_T \end{bmatrix} - C_D \dot{\mathbf{r}} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \right] \\ \mathbf{R}_{\Theta|\Omega} \boldsymbol{\Omega} \\ \frac{1}{\tau}\left(\mathbf{u}_{\boldsymbol{\Omega}} - \boldsymbol{\Omega}\right) \end{bmatrix} \tag{3}$$

Where $\mathbf{r}$ and $\dot{\mathbf{r}}$ are the quadcopter's position and velocity, respectively, expressed in world coordinates. $\boldsymbol{\Theta}$ contains the quadcopter's attitude, expressed in Euler angles. And $\boldsymbol{\Omega}$ is the quadcopter's body angular rate. The rotation matrices $\mathbf{R}_{E|B}$ and $\mathbf{R}_{\Theta|\Omega}$ transform body forces to the world reference frame and transform body angular rates to Euler angular rates, respectively.

a) Controller: The altitude is controlled by a PID controller that governs the collective thrust. A feedforward term is added that compensates for the quadcopters attitude ($T_{\text{feedforward}} = -W/(\cos\theta\cos\phi)$). The desired heading is controlled by a PD controller and the optimal control optimizer will output a desired pitch and roll angle. An inner PD control loop will map the desired attitude angles to angular rates. Absolute limits are predefined for the collective thrust and the roll and pitch angle. Because in this stage of development the model's fidelity was not a priority, generic values for mass, drag coefficients and the aforementioned limits were selected.

1) Longitudinal Flight: Figure 3 shows simulations for several straight flights. Here, the quadcopter starts from rest and should fly to a target 15 meters forward and pass that waypoint with a desired velocity. The plots show the desired and simulated pitch values and velocity for four separate flights in which the desired speed and attitude control gains are varied. This will show the influence of the instantaneous rotation assumption. As the attitude transition takes less time for the high-gain controller than it does for the lower-gain controller.

The plots show that the speed error at the target is larger for the low-gain controller because the transition to braking takes longer, increasing the mismatch with the instantaneous attitude change that prediction model assumes. Moreover, the at higher desired speeds the speed error is smaller because aerodynamic drag has a larger contribution to the deceleration at higher speeds.

2) Longitudinal + Lateral Flight: To test the effect of decoupling the dynamics in the predictor model a flight has been simulated in which the target lies 15 meters ahead and 5 meters to the right. For this flight the heading is fixed and the desired longitudinal speed has been set to 0. The resulting trajectory can be seen in Figure 4. It can be seen that during the transition at the start and at the switching time the commanded roll angle fluctuates instead of being constant

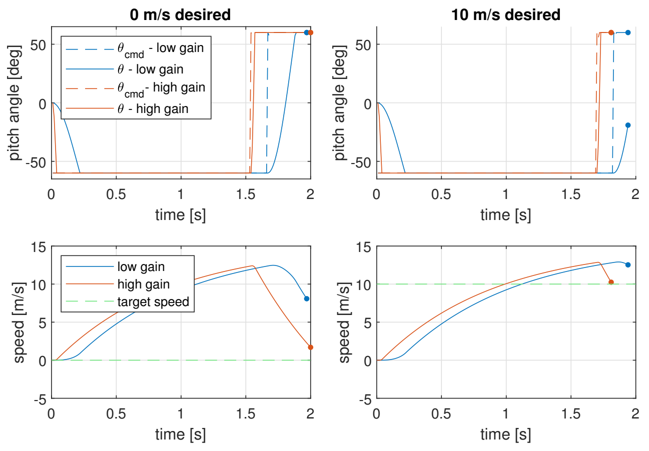


Fig. 3. Simulated straight trajectories for high and low gain attitude controller with 0 m/s and 10 m/s target speeds.

as in the prediction. This is caused by varying thrust from the altitude controller during the transition on which the calculated optimal roll angle is dependent. Nonetheless, the quadcopter reaches the target with a position error of 65.6 cm and a velocity error of 4.1 m/s.

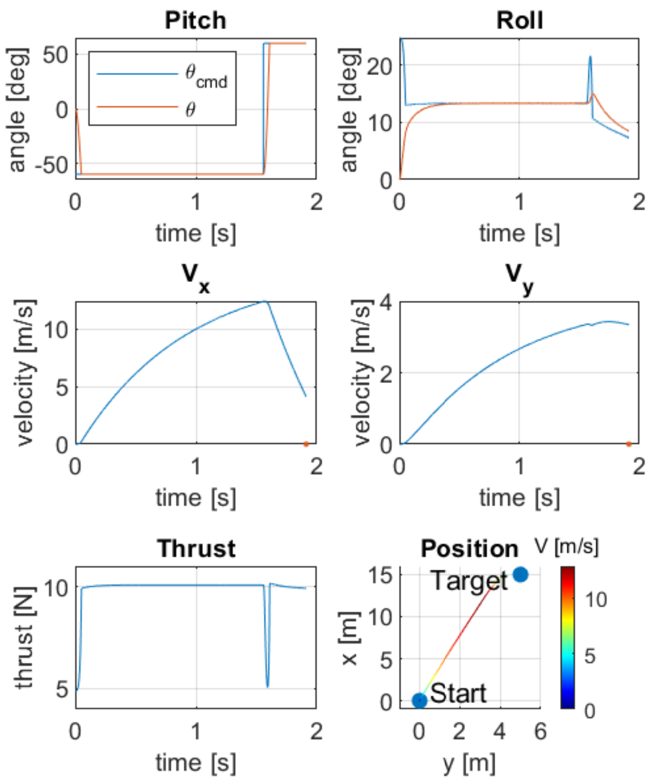


Fig. 4. Simulated Longitudinal + Lateral flight. The longitudinal desired speed at the target is 0 m/s.

a) Prediction Stability: The predicted time of arrival has shown to be a useful indicator of the prediction stability. I.e., this value would decrease proportionally with passed flight time if the flown trajectory matched the predicted trajectory perfectly. Figure 5 shows the predicted time of arrival for the simulated flight depicted in Figure 4. The values are corrected for the passed time, so the flight matches the prediction if the plot is horizontal. It can be seen that after transitions the plot is practically horizontal (the small fluctuations are caused by the numerical precision of the simulation and the bisection algorithm). From this becomes clear that the prediction and simulated flight are nearly identical during the acceleration and final braking phases, but do not comply during the attitude transitions.

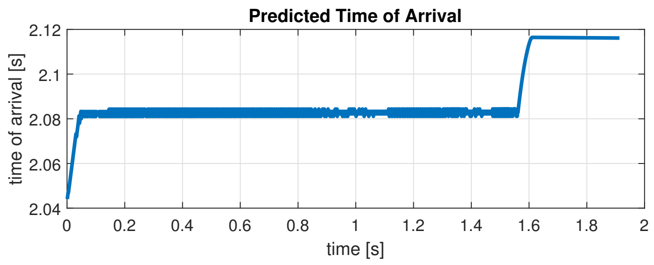


Fig. 5. Course of the estimated time of arrival during the longitudinal+lateral flight simulation. Corrected for passed simulation time.

B. Trajectory Comparisons

In this section the trajectories that our bang-bang method produces are compared to the theoretical time-optimal and minimum-snap solutions. Moreover, a classical PID position and velocity controller have been implemented to provide baseline trajectories. This controller uses a single set of gains. The minimum-time and minimum-snap trajectories are approximated with the help of the ICLOCS toolbox [17]. ICLOCS offers various methods of defining and solving optimal control problems numerically. We have used this tool to transcribe the OCP using the direct collocation method and IPOPT [18] as NLP solver.
Four different scenarios have been simulated: leveled longitudinal flight, longitudinal flight with altitude variation, leveled longitudinal + lateral flight, and finally, longitudinal + lateral flight with altitude variations. In all scenarios the quadcopter starts from rest and should end in rest. The relevant simulation parameters are listed in Table I.

Parameter	*Value*
Mass	0.452[kg]
Max T/W	3 [-]
Max roll and pitch rate	360 [°/s]
Max abs. roll and pitch angles	70°
Max yaw rate	180 [°/s]
C_D	0.57

TABLE I
SIMULATION SETTINGS

1) Longitudinal Flight: The first set of maneuvers to be compared are straight flights and can be can be seen in Figure 6. These flights can be accomplished with pitch and thrust control only.

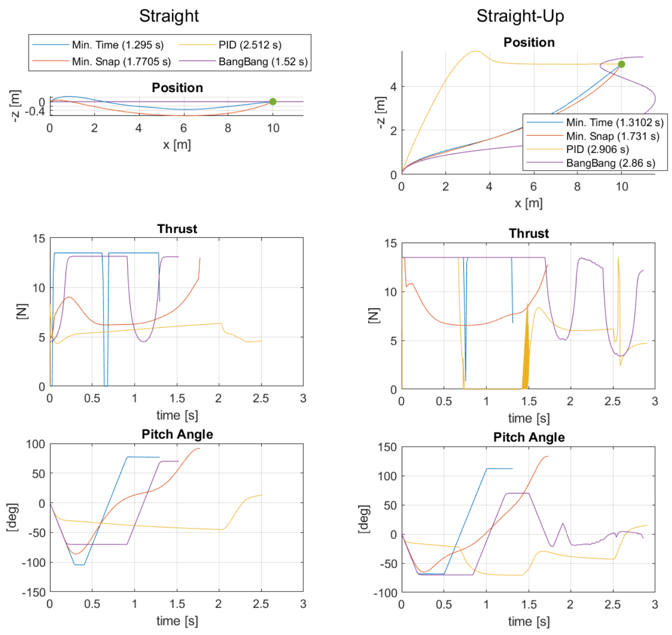


Fig. 6. Simulated trajectories for straight flight. The left column shows the results the target altitude is similar to the start's. In the right column the target altitude is 5m higher.

a) Constant Altitude: For the constant altitude case it can be seen that both the bang-bang and minimum-time approaches give bang-zero-bang inputs to the pitch rate. The thrust also has a bang-zero-bang shape in the minimum-time solution whereas the thrust in the PID and bang-bang controllers are governed by the PID altitude controller. Although the bang-bang controller is the second fastest to reach the target it should be noted that it overshoots its target. Which is the result of the relative slow transition at the switching instant.

b) Altitude Variation: In the second scenario the waypoint target altitude lies at 5m. Since the predictor of the bang-bang controller assumes constant thrust and altitude it is expected that the controller will perform poorly. In the resulting trajectories it can be seen that the bang-bang controller struggles to reach the target altitude because of the high pitch angles. When comparing to the minimum-time solution it can be seen that thrust is saturated much earlier and that the pitch angles increases earlier for more vertical acceleration. There is a optimal balance between horizontal and vertical acceleration. Whereas the PID controller on the other hand prioritizes on minimizing the altitude error first. Moreover, it is interesting to see that the minimum-time and minimum-snap positions look very similar, but that the thrust and pitch values are completely different.

2) Longitudinal + Lateral Flight: In this set of trajectories the flightplan consists of two waypoints to create a curved flight. For the variational altitude flight the first waypoint is 5 meters higher after which the final waypoint is back at the initial zero altitude. The resulting trajectories can be seen in Figure 7. Just as in the longitudinal test it is desired for the quadcopter to be at rest at the final waypoint. However, the bang-bang controller requires a desired velocity for the first waypoint as well in order to optimize for a switching time. This value has been chosen to be equal to the velocity at the first waypoint from the optimal-time solution. It is expected that this velocity will be too high because the optimal-time solution optimizes for the complete trajectory and will likely pass through the first waypoint at an angle that points closer to the final waypoint. The bang-bang controller on the other cannot predict beyond its next waypoint.

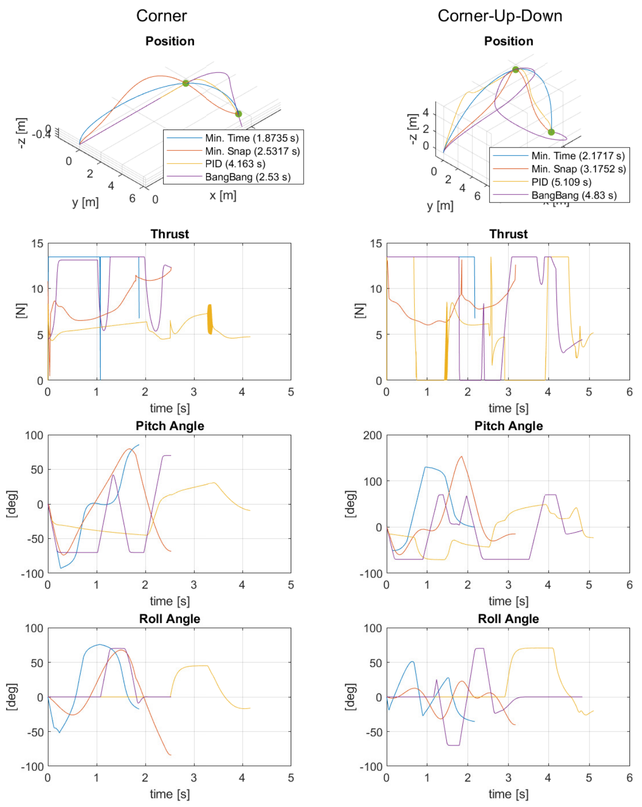


Fig. 7. Simulated trajectories for longitudinal + lateral flight. The left column shows the results all waypoint's altitudes are equal. In the right column the first waypoint's altitude is 5m higher.

In the constant altitude case, the minimum-time trajectory takes the form of a smooth continuous turn, first rolling to the left to better line up for the final waypoint. The minimum-snap trajectory even takes on a larger turn. As expected, the bang-bang controller overshoots the first waypoint before turning and heading to the final waypoint.

For the variational altitude trajectories shows that the PID and bang-bang controllers have similar behavior towards the first waypoint. However, while flying to the final waypoint the altitude controller will give less thrust to lower the altitude. The optimal-time solution on the other hand will keep the thrust saturated and leverage more extreme pitch and roll angles to control the altitude such that all power is available to keep accelerating or decelerating, which was also demonstrated in the work of Hehn, Ritz, and D'Andrea [8].

It can be seen that the time-optimal solution indeed keeps the thrust saturated throughout the entire flight. Its pitch angle going as high as $130°$. Since the PID and bang-bang only takes one waypoint in account at a time the first part of the trajectory looks similar. During the last segment the bang-bang controller still suffers from the overshoot of the first segment and cannot accelerate towards the final waypoint because the thrust is zero during the descent.

C. *ROS FlightGoggles Simulator*

Flightgoggles is a comprehensive quadcopter simulator developed by Guerra, Tal, Murali, *et al.* [16] to benefit research on autonomous control. It is built within the well-known ROS framework [19]. The quadcopter dynamics are simulated by a high fidelity model that includes rotor dynamics even approximates turbulence.
We have implemented the bang-bang controller to prove effective flight in a simulation with high fidelity dynamics.

1) Controller: The desired thrust, pitch angle and roll angle are calculated by the bang-bang controller. The desired heading is calculated from the position error such that the quadcopter always points toward the waypoint. When within 1m of the next waypoint the desired heading is frozen to avoid large fluctuations. A PID controller is used to map the desired attitude angles to angular rates which are sent to the inner loop controllers of Flightgoggles. Finally, the altitude controller is similar to the Matlab simulation in Subsection IV-A which consists of a PID controller with a feedforward term. For position and velocity control the groundtruth values of the simulator are used. The attitude is estimated with the provided on-board sensor simulations.
Although no detailed analyses were performed in this simulator. It was found that the suggested bang-bang controller is feasible for quadcopter models that have a higher fidelity compared to the path prediction model.

V. Transition Compensation

The assumptions introduced in the prediction model will lead to differences between the predicted trajectory and the actual quadcopter's flight path. Leading to sub-optimal flight results.

As explained in Section III-A it is assumed that switching maneuver from acceleration to deceleration is instantaneous. In practice it is not possible for a quadcopter to achieve infinitely high rotation rates and high prediction inaccuracies are expected during this transition. Because the quadcopter starts effectively braking later and at a position closer to the target it can be assumed that braking will always be initiated too late. This can also be seen in the results of Section IV-B.

In order to mitigate this issue a method has been implemented that approximates how much speed, position and time will be lost during the transition at the switching instant. Which are expressed as differences Δt, Δy and Δv relative to the values at the switching instant t_s, $y(t_s)$ and $v(t_s)$, respectively. Subsequently, the initial conditions of the second segment of the bang-bang predictions are adapted to these losses to improve the final segment of the predicted trajectory. Figure 8 shows the effect of applying compensation to a prediction. It can be seen that due to the approximated transition time the optimizer has calculated the switching time to be sooner for the compensated trajectory.

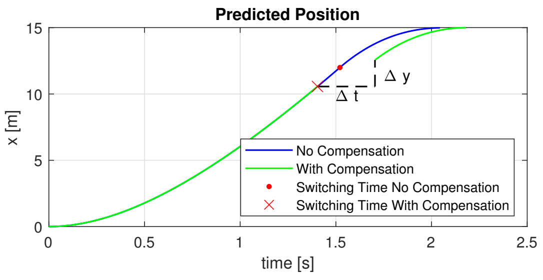


Fig. 8. Example of compensating for transition losses on the path prediction. It is approximated that the transition takes $\Delta t \ s$ seconds during which the quadcopter moves $\Delta y \ m$ and its speed changes with $\Delta v \ \frac{m}{s}$.

A risk of using this method is that during the transition the path prediction must be paused for the estimated transition period to avoid conflicting predictions and premature termination of the transition maneuver. Furthermore, the estimated transition losses cannot be incorporated in the prediction after the transition is completed. So during the braking, after the transition is completed, the optimizer will be under the impression that the current velocity is too low and thereby counteract the wins from the compensation to certain extent.

Figure 9 shows the effects of applying different degrees of compensation. In these simulations the target speed is zero. It can be seen that without compensation braking is initiated too late and therefore the quadcopter's speed at the target it still too high. However, one could also overcompensate and brake too early. As aforementioned, when applying compensation the quadcopter is forced to finish its transition and cannot be interrupted. The result with overcompensation is that after the transition the quadcopter's speed will be too low and a new bang-bang maneuver must be planned.

VI. Flight Experiments

To analyze the bang-bang controller's performance in real flight, several flight tests have been performed with a Parrot Bebop quadcopter. The controller's performance for different types of maneuvers is compared to the performance of a more traditional PID controller.

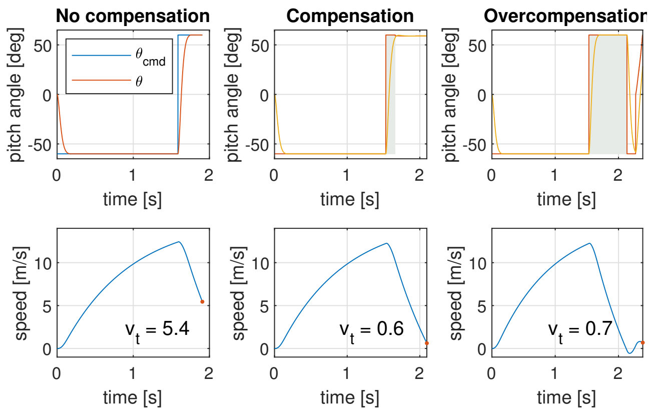


Fig. 9. Simulated straight flights with three different degrees of transition compensation. The desired speed at the target position is 0.

A. Experimental Setup

The bang-bang controller has been implemented in the autopilot framework PaparazziUAV[20]. And will be executed onboard a Parrot Bebop quadcopter. The flight experiments were performed in TU Delft's 'Cyberzoo', an enclosed indoor space purposed for UAV flights particularly. The space is outfitted with an Optitrack system that can accurately track the quadcopter's position and heading at high sampling frequencies.

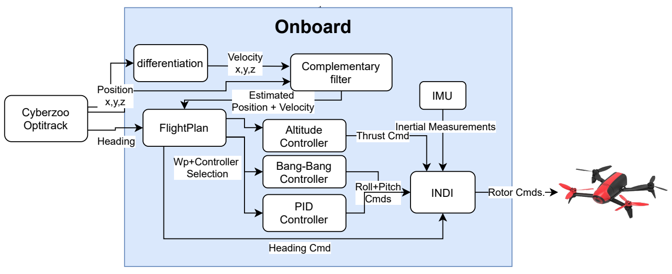


Fig. 10. Flight Experiments Control Pipeline

1) Control Pipeline: An overview of the control pipeline of the Bebop can be seen in Figure 10. Position measurements are taken by the Optitrack system and transmitted to the Bebop over WiFi. The quadcopter's velocity and acceleration are estimated from these measurements on-board by means of a complementary filter that assumed hover condition and uses the aerodynamic drag model as the path prediction model (Equation 1). Equation 4 shows how the velocity $\hat{\mathbf{v}}_E$ and position $\hat{\mathbf{r}}_E$ are estimated by combining the velocity and position measurements from the Optitrack system with the modeled acceleration $\mathbf{a}_E$.

$$\mathbf{a}_E = g\hat{\mathbf{z}}_B\mathbf{R}_{E|B} - \frac{C_d}{m}\hat{\mathbf{v}}_E \tag{4a}$$

$$\hat{\mathbf{v}}_E = \alpha\left(\hat{\mathbf{v}}_E + \mathbf{a}_E \cdot dt\right) + (1-\alpha)\,\mathbf{v}_{E,\text{GPS}} \tag{4b}$$

$$\hat{\mathbf{r}}_E = \alpha\left(\hat{\mathbf{r}}_E + \hat{\mathbf{v}}_E \cdot dt\right) + (1-\alpha)\,\mathbf{r}_{E,\text{GPS}} \tag{4c}$$

Where $\hat{\mathbf{z}}_B$ is the body z-axis unit vector, $\mathbf{R}_{E|B}$ is the transformation matrix from body to earth reference frame and α is the complementary coefficient.

Furthermore, Optitrack also provides heading measurements. The pitch and roll angles are estimated on-board by the Bebop's inertial measurement unit (IMU) and a complementary filter. The desired pitch and roll angles are provided by the bang-bang controller that is executed onboard in real-time at 512Hz. Which are passed on to inner loop controllers that are based on INDI [21]. Finally, the altitude is controlled by a PID controller with a feedforward term that compensates for the pitch and roll angles, similar to the simulations.
A high-gain PID position and velocity controller has been implemented as well for comparison. The feedback loops maps the position error to desired velocity which are mapped to desired roll and pitch rates. Since the Cyberzoo is limited by its available space, upper limits on the longitudinal and lateral velocity have been set to 10m/s to avoid colliding with the walls.
A special case is considered when a bang-bang maneuver must end in rest. Due to prediction inaccuracies the quadcopter will likely overshoot or undershoot the target position and as a consequence a new bang-bang maneuver will immediately be planned. This will result in endless aggressive bang-bang oscillations around the target position. To avoid this, the quadcopter's control approach will temporarily switch to PID after the first bang-bang maneuver has been completed.

B. *Transition Loss Estimators*

As explained in Section V the path predictor can be augmented to incorporate the losses in time, position and speed (Δt, Δy and Δv) that occur during the transition from acceleration to braking (or vice versa). For a real quadcopter these dynamics can be difficult to model accurately because complex aerodynamics and the changing aerodynamic shape affect the aerodynamic forces and moments acting on the quadcopter during the transition. However, for a proof of concept a simple linear regression model has been fitted against in-flight measurements to approximate the expected transition losses.
Three different models has been fitted by least-squares regression for forward, backward and lateral flight maneuvers. It is assumed that Δt, Δy and Δv are functions of the quadcopter's velocity at the switching instant and the total angle that the quadcopter needs to rotate. Table II lists the root-mean-square error (RMSE) for each individual estimator to indicate how well the models fit the measurements.

	Pitch Forward	Pitch Backward	Roll Sideways
Δt [s]	1.73e−2	1.59e−2	2.57e−2
Δy [m]	7.08e−2	7.95e−2	1.44e−1
Δv [$\frac{m}{s}$]	6.88e−2	1.37e−1	1.93e−1

TABLE II
THE ROOT-MEAN-SQUARE ERROR OF THE TRANSITION LOSS ESTIMATORS

In the control pipeline one of the three estimator models will be selected based on in which direction a bang-bang maneuver is planned. Subsequently, from the selected model the transition losses are calculated for each path prediction in the optimization process, as explained in Section V, which should lead to finding a better switching time.

C. *Motion Primitives Flights*

Test flights have been performed to analyze the Bang-Bang controller performance against a traditional PID controller with a fixed set of gains. Furthermore, the effect of the transition compensation has been investigated as well. Six different maneuvers have been established to test performance for different motion primitives. For each maneuver the quadcopter starts from rest and should end in rest. For example, Figure 11 shows the responses of three individual runs for a forward flight maneuver.

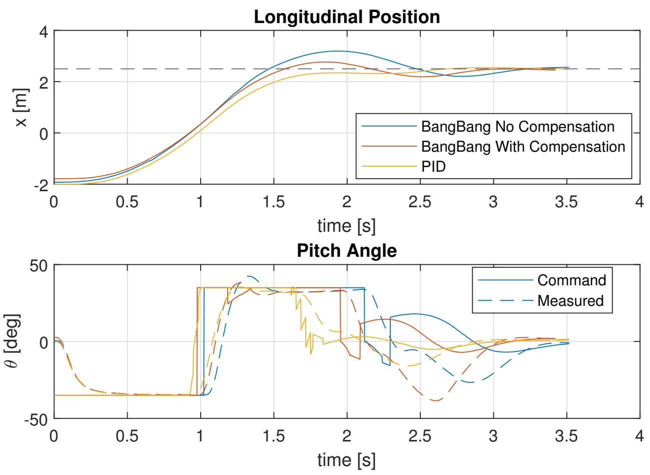


Fig. 11. Forward flight comparison between the PID, Bang-Bang controller and BangBang controller with transition compensation.

Here some distinctions between the three control approaches can be seen. Because the PID controller is a high-gain velocity controller the commanded pitch angles strongly resemble a bang-bang shape. However, as the position approaches the target the pitch command converges towards zero. The PID controller is in fact a little bit overdamped as it does not cross the target. Also the beneficial effect of the transition compensation can be seen. The compensated bang-bang controller brakes slightly earlier which decreases the velocity error at the target and the overshoot. Overall, each controller performance can best be described by time of arrival and the velocity error at the target position. To test different combinations of longitudinal and lateral waypoints as well as altitude variation the following maneuvers have been performed: forward, backward, sideways, forward-sideways, forward-up and forward-down. The results of all test flights are summarized in Table III. It should be noted that the gains of the PID controller has been optimized for the forward-sideways maneuver and remain fixed for all other maneuvers.

From the results can be seen that the transition compensation system reduces the overshoot and velocity error for all maneuvers. However, as the PID controller is actually close to overdamped and practically never has overshoot it is impossible for the bang-bang controller to surpass the PID for this performance parameter. Nonetheless, it can also be seen that in terms of speed there is a trade-off between reaching

Controller	**Maneuver**					
	Forward	Backward	Sideways	Forward-Sideways	Forward-Up	Forward-Down
	Mean Time of Arrival [s]					
Bang-Bang	**1.38** (n=4)	**1.41** (n=4)	**1.29** (n=5)	**1.47** (n=4)	**1.25** (n=3)	**1.50** (n=3)
Bang-Bang Comp.	1.42 (n=15)	1.47 (n=12)	1.37 (n=11)	1.53 (n=15)	1.32 (n=7)	1.52 (n=7)
PID	1.48 (n=10)	1.54 (n=8)	1.43 (n=8)	1.51 (n=8)	1.40 (n=5)	1.54 (n=5)
	Mean Overshoot [m]					
Bang-Bang	0.62 (n=4)	0.81 (n=4)	0.53 (n=5)	0.27 (n=4)	1.20 (n=3)	0.77 (n=3)
Bang-Bang Comp.	0.18 (n=15)	0.22 (n=12)	0.06 (n=11)	0.11 (n=15)	0.51 (n=7)	0.20 (n=7)
PID	**0.05** (n=10)	**0.04** (n=8)	**0.04** (n=8)	**0.04** (n=8)	**0.03** (n=5)	**0.14** (n=5)
	Mean Velocity Error [$\frac{m}{s}$]					
Bang-Bang	3.05 (n=4)	3.51 (n=4)	3.06 (n=5)	1.85 (n=4)	3.82 (n=3)	3.38 (n=3).
Bang-Bang Comp.	1.60 (n=15)	1.88 (n=12)	0.69 (n=11)	0.38 (n=15)	2.63 (n=7)	1.63 (n=7)
PID	**0.08** (n=10)	**0.08** (n=8)	**0.14** (n=8)	**0.06** (n=8)	**1.01** (n=5)	**0.20** (n=5)

TABLE III
PERFORMANCE VALUES THE DIFFERENT CONTROLLERS IN 6 DIFFERENT MANEUVERS.

the target fast with overshoot, or reaching the target more slowly, but more accurately. The bang-bang controller with compensation is almost always positioned comfortably in the middle.

D. Consecutive waypoints flight

To approach flight maneuvers that approach autonomous drone races more closely a 3m by 4m rectangular flightplan has been set up in the Cyberzoo. Again, separate flight with the bang-bang controller and the PID controller have been performed. A key difference with the motion primitives of Section VI-C is that quadcopter is not instructed to come to rest at each waypoint. In fact, for the bang-bang controller it was found that a desired speed of 2 m/s yielded the most stable results. Since the PID controller bases the desired speed on the position error it would still come to rest at each waypoint. To still allow a smooth flight the flightplan has been adapted for both controller to switch to the next waypoint when the quadcopter is within 70 cm of the target. A limitation of the bang-bang controller is that the predictor cannot incorporate a variable heading. Therefore, the heading is kept fixed for both the bang-bang controller and the PID controller. Moreover, the bang-bang controller switches the saturation direction based on which component of the position error is largest. That is, if the next waypoint lies mostly ahead or behind there will be bang-bang motion in pitch. If the waypoint lies more to the left or right there will be a bang-bang motion in roll.

Figure 12 shows a top-view of the multiple waypoint flights as well as an averaged trajectory. Compositions of single runs from both controller are found in Figure 14. Already a clear difference in shape between the bang-bang and PID controller can be seen. The bang-bang controller trajectories are more consistent and smoother than the trajectories from the PID controller. Furthermore, as can be seen in Figure 13 the bang-bang controller does not only pass the waypoints more closely, it also has always completed a circle more quickly than the PID controller.

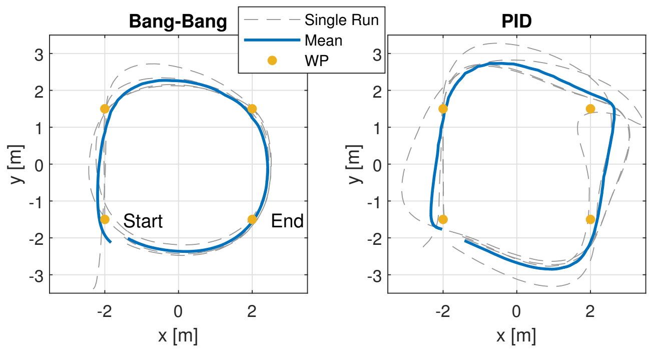


Fig. 12. Top-view of multiple-waypoints flight

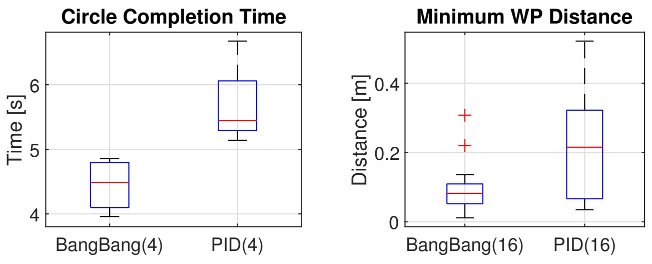
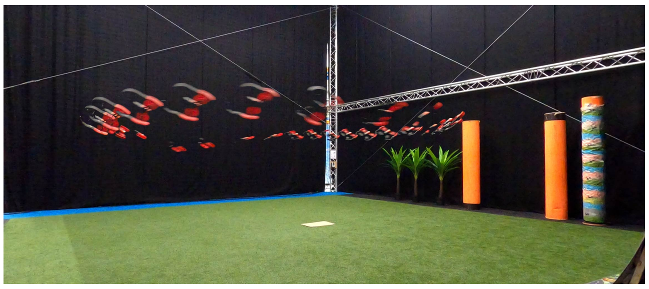
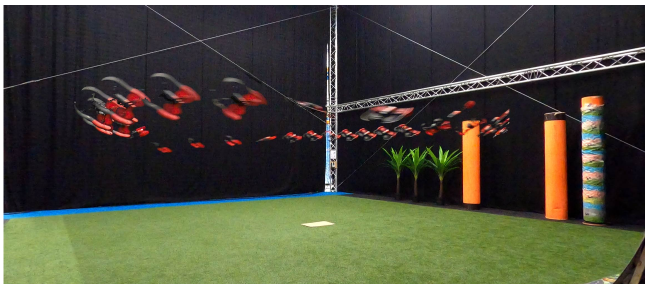


Fig. 13. A boxplot comparison between the Bang-Bang and PID controller, looking at the time to complete a circle and how close the quadcopter approaches the waypoints.

The difference in performance is mainly caused by the fact that a PID controller uses the same high gains for longitudinal and lateral control and can therefore not prioritize between directions. This implies that controller will attempt to minimize the position error in both directions as quickly as possible, causing oscillations in the direction that is reached first and not optimally leveraging the available thrust. One could mitigate this problem to some extend by incorporating separate sets of gains in the flightplan, but this can be a tedious process.
On the other hand, the bang-bang controller will automatically only opt for maximum accelerations in one direction whilst going for the minimum required acceleration in the other.


(a) BangBang


(b) PID

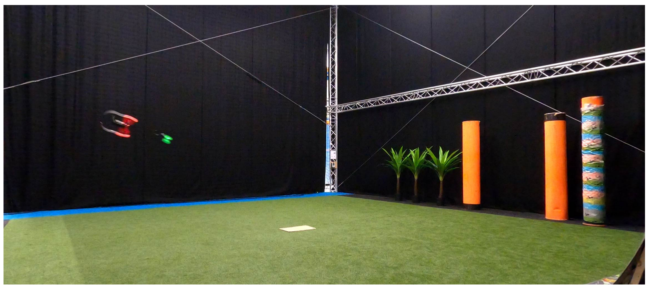

(c) Position difference after one circle completion (green is the bang-bang)

Fig. 14. Compositions of BangBang and PID circular trajectories

Moreover, as the required thrust to maintain altitude scales with $(\cos\theta\cos\phi)^{-1}$ we have to put saturation limits on the roll and pitch angles. For the PID approach it must be taken into account that the roll and pitch angle can often both be saturated at the same time. For the bang-bang controller this is much less likely and therefore a higher limit can be put on the angle responsible for the saturation direction while maintaining stability. It should be noted that in all flight experiments the same saturation limits were given to the bang-bang and PID controller.

E. Prediction Stability

As aforementioned in Section IV-A2, an effective measure for prediction performance is the development of the estimated time of arrival. Figure 15 shows this and the attitude angles a single run of the multiple waypoints flight. The predicted time of arrival is corrected for the elapsed time such that a perfect prediction would yield a horizontal line. Initially it can be seen that the predicted time rises, indicating that it will take longer that initially expected to reach the waypoint. Which is caused by the transition from rest in pitch. However, after this the predicted time lowers because it is flying faster than initially expected. This can be caused by an inaccurate aerodynamic

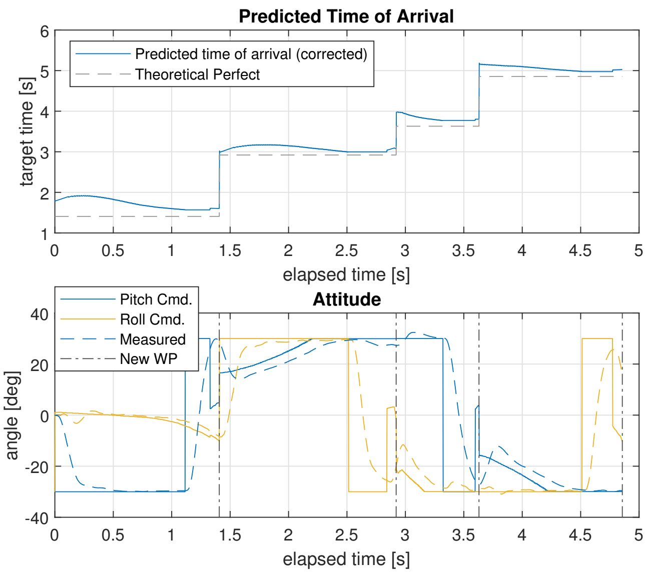


Fig. 15. Predicted time of arrival throughout one circle

drag model and by the effect the roll angle has on the forward acceleration.
Furthermore, between 1.4 and 2.9 seconds the lateral direction becomes the saturation dimension and a bang-bang motion for roll is planned. Here the predictor assumes that the pitch must have a constant angle, but it can be seen that it actually slowly increases during its course to the next waypoint. This is not only caused by an inaccurate prediction of non-saturation direction. Remember that the predictor only uses the prediction in the saturation dimension to estimate the time of arrival. Since the calculated constant angle is based on the predicted time of arrival, the fact that this value decreases means that an increasingly higher pitch angle is required to still reach the target at the predicted time of arrival. This implies that the prediction quality of the saturation direction influences the prediction stability of the remaining direction.

VII. Discussion

In this work we have presented a computationally lightweight approach to time optimal control. It is based on the assumption that the optimal trajectory consists of bang-bang inputs in pitch or roll. This has simplified the OCP such that only the optimal switching time and a constant attitude angle need to be found. Simplifications to the dynamic model allow for analytical evaluation of the position and speed which drastically reduces the required computational effort compared to solvers that need to propagate the states. As a result the complete optimizer pipeline has no trouble predicting 512 optimized trajectories per second on a Parrot Bebop drone. However, these simplification and assumptions do have an effect on how well the theoretical time-optimal is approached. First of all, we have to manually specify the desired speeds at the waypoints. For which we do not know beforehand what values are time-optimal. Furthermore, we are limited to the constraints of the relative simple dynamics model that predictor uses. Which includes constant altitude and heading. Therefore, the bang-bang controller cannot look past its next waypoint which is required if we want to achieve global time-optimality for the entire course. Also no complex aerodynamics such as wakes and propeller dynamics are taken into account.

But as the simulations and flight tests have shown the assumption that attitude changes are instantaneous has the biggest effect on prediction accuracy. The fact that the model does not take into account that it takes some time to rotate from one extreme attitude angle to the other results in a switching time that will always be too late, making the quadcopter overshoot its target. To mitigate this issue we have suggested a system to compensate for this transition by simply using a fitted linear model to estimate how much time the transition will take and how much the velocity and position change during this period. The predicted paths are then adapted with this values which leads to a better switching time. The flight tests have shown great reductions in overshoot when using this system, as shown in Table III, despite that the estimator is linear and derived from noisy measurements.

Flight tests with consecutive waypoints have been performed with our suggested bang-bang controller and a more traditional PID controller. Here it became clear that the bang-bang controller outperforms the PID both in course completion time as in terms of how closely the waypoints were approached, under the condition that the heading remains fixed. The main reason for its superior performance is that the controller effectively puts most effort in minimizing the largest positional error component whilst putting the minimum required effort in the remaining direction. A PID controller attempts to reach both the longitudinal as the lateral target with identical priority. Which results in large simultaneous roll and pitch angles and positional oscillations.

Despite the limited conditions under which the suggested controller is optimal, its ease of implementation is a major benefit. Where a PID controller must carefully be tuned for different maneuvers or a neural network-based controller must be trained beforehand on a huge dataset, our bang-bang controller does not require gain tuning and the only knowledge required from the quadcopter is its mass, equivalent drag coefficient and estimators for the transition compensation system. All of which could also easily be learned on-line. This could for instance be implemented in future work by means of recursive least-squares estimators [20] or a more sophisticated method. Additional future work suggestions includes making the saturation angles adaptive. As currently the maximum roll and pitch angles must be defined beforehand and take into account a margin to have an amount of thrust available to correct for deviations and disturbances. An adaptive system could be developed that estimates how much thrust must be reserved for these corrections and adapt the roll and pitch angle limits accordingly.

VIII. Conclusion

In this paper we have presented a computational efficient controller that is fast enough to act as an MPC on computationally-limited quadcopters while still approaching time-optimality under certain conditions. The controller is based on the assumption that the optimal maneuver to be planned has a bang-bang shape in either pitch or roll angle. The corresponding optimal control problem thereby reduces to a problem for which we only need to solve for a switching time and a constant angle. Which can be solved extremely quickly using a simple bisection scheme thanks simplifications to the dynamical model that allows for analytical path evaluations. A transition estimation model has been implemented to compensate for rotational dynamics that the predictor cannot take into account. With this system implemented the bang-bang controller has shown to outperform a traditional high-gain PID controller for indoor flights with consecutive waypoints. Furthermore, the suggested controller requires no gain tuning and can therefore be implemented easily in any quadcopter. Despite that future work on adaptive saturation angles, better transition estimation and better aerodynamics modeling are suggested, we have shown that this relatively simple approach to time-optimality results in fast flights with minimal computational power and minimal knowledge of the quadcopter's dynamics.

References

[1] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, "The IROS 2016 Competitions [Competitions]," *IEEE Robotics and Automation Magazine*, vol. 24, no. 1, pp. 20–29, 2017, ISSN: 10709932. DOI: 10.1109/MRA.2016.2646090.

[2] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T. C. Au, and S. J. Kim, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, no. 2, 2019, ISSN: 18612784. DOI: 10.1007/s11370-018-00271-6.

[3] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "AlphaPilot: Autonomous Drone Racing," 2020. arXiv: 2005.12813. [Online]. Available: http://arxiv.org/abs/2005.12813.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2520–2525, ISBN: 9781612843865. DOI: 10.1109/ICRA.2011.5980409.

[5] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," *Springer Tracts in Advanced Robotics*, vol. 114, no. Isrr, pp. 649–666, 2016, ISSN: 1610742X. DOI: 10.1007/978-3-319-28872-7_37.

[6] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018, ISSN: 23773766. DOI: 10.1109/LRA.2017.2776353. arXiv: 1712.02402.

[7] F. Biral, E. Bertolazzi, and P. Bosetti, "Notes on numerical methods for solving optimal control problems," *IEEJ Journal of Industry Applications*, vol. 5, no. 2, pp. 154–166, 2016, ISSN: 21871108. DOI: 10.1541/ieejjia.5.154.

[8] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, no. 1-2, pp. 69–88, 2012, ISSN: 09295593. DOI: 10.1007/s10514-012-9282-3.

[9] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Journal of Robotics Research*, 2012. DOI: 10.1177/0278364911434236.

[10] E. Tal and S. Karaman, "Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2018-Decem, pp. 4282–4288, 2019, ISSN: 07431546. DOI: 10.1109/CDC.2018.8619621. arXiv: 1809.04048.

[11] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep Drone Acrobatics," *arXiv*, 2020, ISSN: 23318422. DOI: 10.15607/rss.2020.xvi.040. arXiv: 2006.05768. [Online]. Available: http://arxiv.org/abs/2006.05768.

[12] S. Li, E. Ozturk, C. De Wagter, G. C. H. E. de Croon, and D. Izzo, "Aggressive Online Control of a Quadrotor via Deep Network Representations of Optimality Principles," 2019. arXiv: 1912.07067. [Online]. Available: http://arxiv.org/abs/1912.07067.

[13] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep Drone Racing: From Simulation to Reality With Domain Randomization," *IEEE Transactions on Robotics*, pp. 1–14, 2019, ISSN: 1552-3098. DOI: 10.1109/tro.2019.2942989. arXiv: 1905.09727. [Online]. Available: http://arxiv.org/abs/1905.09727.

[14] R. Bellman, I. Glicksberg, and O. Gross, "On the "bang-bang" control problem," *Quarterly of Applied Mathematics*, vol. 14, no. 1, pp. 11–18, 1956, ISSN: 0033-569X. DOI: 10.1090/qam/78516.

[15] MATLAB, *version 9.7.0 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[16] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality," 2019. arXiv: 1905.11377. [Online]. Available: http://arxiv.org/abs/1905.11377.

[17] Y. Nie, O. Faqir, and E. C. Kerrigan, "ICLOCS2: Try this Optimal Control Problem Solver before you Try the Rest," *2018 UKACC 12th International Conference on Control, CONTROL 2018*, vol. 2, no. 2017, p. 336, 2018. DOI: 10.1109/CONTROL.2018.8516795.

[18] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, 2006, ISSN: 00255610. DOI: 10.1007/s10107-004-0559-y.

[19] Stanford Artificial Intelligence Laboratory et al., *Robotic operating system*, version ROS Melodic Morenia, May 23, 2018. [Online]. Available: https://www.ros.org.

[20] P. Brisset, A. Drouin, M. Gorraz, P.-s. Huard, and J. Tyler, "The Paparazzi Solution," *Mav2006*, 2006.

[21] E. J. Smeur, Q. Chu, and G. C. De Croon, "Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 450–461, 2016, ISSN: 07315090. DOI: 10.2514/1.G001490. [Online]. Available: http://resolver.tudelft.nl/uuid:31536cfa-89e1-4d44-873e-f2f398bd69ca.

Appendix A
Prediction Model Derivation

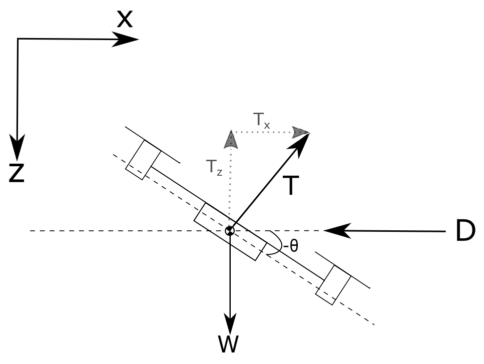


Fig. 16.

$T =$ Thrust
$D =$ Drag $= C_d \cdot V$
$W = mg =$weight
$m =$ mass
$g =$ gravity
$\theta =$ attitude angle
$x =$ position

Assume perfect constant altitude control:

$$\begin{aligned} T_z =& W \\ T =& \frac{W}{\cos\theta} \\ T_x =& T\sin\theta = W\frac{\sin\theta}{\cos\theta} = W\tan\theta \end{aligned} \tag{5}$$

Sum of forces:

$$\begin{aligned} m\cdot\ddot{x} =& T_x - D = W\tan\theta - C_d\cdot\dot{x} \\ \ddot{x} =& g\tan\theta - \frac{C_d}{m}\dot{x} \end{aligned} \tag{6}$$

Homogeneous equation:

$$\ddot{x} + \frac{Cd}{m}\dot{x} = 0 \tag{7}$$

Characteristic equation:

$$\begin{aligned} r^2 + \frac{C_d}{m}r =& 0 \\ r\left(r + \frac{C_d}{m}\right) =& 0 \\ r_1 = 0, r_2 =& \frac{-C_d}{m} \\ x_h =& c_1 + c_2 e^{\frac{-C_d}{m}t} \end{aligned} \tag{8}$$

Particular equation:

Assume $x_p = A\cdot t$, where A is a constant:

$$\begin{aligned} x_p =& At \\ \dot{x_p} =& A \\ \ddot{x_p} =& 0 \\ 0 =& g\tan\theta - \frac{C_d}{m}\cdot A \\ A =& \frac{W\tan\theta}{C_d} \\ x_p =& \frac{W\tan\theta}{C_d}t \end{aligned} \tag{9}$$

$$x = x_h + x_p = c_1 e^{\frac{-C_d}{m}t} + c_2 + \frac{W\tan\theta}{C_d}t \tag{10a}$$

$$\dot{x} = c_1\frac{-C_d}{m}e^{\frac{-C_d}{m}t} + \frac{W\tan\theta}{C_d} \tag{10b}$$

Solve for initial conditions:

$$\begin{aligned} \dot{x}_0 =& c_1\frac{-C_d}{m}e^{\frac{-C_d}{m}t_0} + \frac{W\tan\theta}{C_d} \\ c_1 =& \left(\dot{x}_0 - \frac{W\tan\theta}{C_d}\right)\frac{m}{-C_d e^{\frac{-C_d}{m}t_0}} \\ x_0 =& c_1 e^{\frac{-C_d}{m}t_0} + c_2 + \frac{W\tan\theta}{C_d}t_0 \\ c_2 =& x_0 - c_1 e^{\frac{-C_d}{m}t_0} - \frac{W\tan\theta}{C_d}t_0 \\ =& x_0 - \left(\dot{x}_0 - \frac{W\tan\theta}{C_d}\right)\frac{m}{-C_d e^{\frac{-C_d}{m}t_0}}e^{\frac{-C_d}{m}t_0} - \frac{W\tan\theta}{C_d}t_0 \end{aligned} \tag{11}$$

In the MPC we always assume that $t_0 = 0$ which makes the constants:

$$\begin{aligned} c_1 =& \left(\dot{x}_0 - \frac{W\tan\theta}{C_d}\right)\frac{m}{-C_d} \\ c_2 =& x_0 - \left(\dot{x}_0 - \frac{W\tan\theta}{C_d}\right)\frac{m}{-C_d} = x_0 - c_1 \end{aligned} \tag{12}$$

Because of the specific rotation order of euler angle transformations, the lateral component of the thrust force is also affected by θ. Therefore, the lateral position and velocity are described by:

$$y = c_3 e^{\frac{-C_d}{m}t} + c_4 + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d}t \tag{13a}$$

$$\dot{y} = c_3\frac{-C_d}{m}e^{\frac{-C_d}{m}t} + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d} \tag{13b}$$

Where constants c_3 and c_4 are again found with the initial conditions y_0 and $\dot{y}_0$.

Appendix B
Bang-Bang Proof

This section demonstrates how PMP can be used to prove that a bang-bang control input shape is the solution to a time-optimal control problem. This derivation is similar to the derivation given by Hehn, Ritz, and D'Andrea [8] but is adapted to work with attitude angle as control input rather than attitude rate.

Consider the system depicted in Fig. 16. The motion of this quadcopter can be described by:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} x & \dot{x} & z & \dot{x} & \theta \end{bmatrix}^T \\ \mathbf{u} &= \begin{bmatrix} u_{\dot{\theta}} & u_T \end{bmatrix}^T \\ \begin{bmatrix} \underline{u_{\dot{\theta}}} & \underline{u_T} \end{bmatrix}^T &\leq \mathbf{u} \leq \begin{bmatrix} \overline{u_{\dot{\theta}}} & \overline{u_T} \end{bmatrix}^T \\ f(\mathbf{x},\mathbf{u}) &= \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{m}\left(u_T \sin\theta - C_d \cdot \dot{x}\right) \\ \dot{z} \\ \frac{1}{m}\left(W - u_T \cos\theta - C_d \cdot \dot{z}\right) \\ u_{\dot{\theta}} \end{bmatrix} \end{aligned} \tag{14}$$

However, we assume constant altitude and instantaneous rotation. Therefore we can discard the $\dot{z}$, $\ddot{z}$ and $\dot{\theta}$ dynamics. Furthermore, state θ becomes input u_θ.

In optimal control theory, for a system of the form $\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t)$ and a cost function of the form $J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt$, the Hamiltonian can be given by:

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[f(\mathbf{x}(t), \mathbf{u}(t), t)] \tag{15}$$

For an OCP in which we desire to minimize time the cost function $h(\mathbf{x}(t_f), t_f) = 0$ and $g(\mathbf{x}(t), \mathbf{u}(t), t) = 1$. The Hamiltonian then becomes:

$$\mathcal{H} = 1 + \mathbf{p}^T f(\mathbf{x}, \mathbf{u}) = 1 + p_1 \dot{x} + p_2 \frac{1}{m}\left(u_T \sin u_\theta - C_d \dot{x}\right) \tag{16}$$

Where p_n are the Lagrange multipliers (or costates). PMP states that the optimal control input minimizes the Hamiltonian. From this it can be found that, depending on the sign of p_2, u_θ^* becomes either $\underline{u_\theta}$ or $\overline{u_{\dot{\theta}}}$. Similarly,$u_T$ is either $\underline{u_T}$ or $\overline{u_T}$. The conditions do not hold when p_2, u_T or u_θ is zero. In this case the nonzero control input takes on a so called singular value. That can often be solved by explicitly expressing the costates from the second PMP that states that

$$\dot{\mathbf{p}} = -\nabla_x \mathcal{H}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{p}) \tag{17}$$

From this we find that:

$$\begin{aligned} \dot{p_1} &= -\frac{\partial \mathcal{H}}{\partial x} = 0 \rightarrow p_1 = c_1 \\ \dot{p_2} &= -\frac{\partial \mathcal{H}}{\partial \dot{x}} = -p_1 \rightarrow p_2 = c_2 - c_1 t \end{aligned} \tag{18}$$

This means that p_2 is only zero at a single instant when $t = \frac{c_2}{c_1}$. Which we can neglect. And because u_T is governed by the hover condition it is never zero, we therefore can also discard the singular arc of u_θ and state that u_θ has a bang-bang shape.

Part II

Literature Review

2 Optimal Control Theory

The largest theoretical basis that the research will be based upon is the optimal control theory. This field in mathematics is concerned with finding the control laws for which a system will move from one state to a target state while minimizing a certain cost function. This chapter will give a brief introduction to Optimal Control Theory and explain several common methodologies within the framework.

2.1. Introduction to Optimal Control

Several classes have been distinguished within the framework of optimal control theory with each their own methodologies of formulating and solving optimal control problems (OCPs). Generally, an OCP takes on the following form:
A system of differential equations, representing the dynamics:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.1}$$

Where $x(t)$ is the state variable and $u(t)$ is the input variable. This system should propagate from a initial state $x(t_0)$ to a target state $x(t_f)$ while minimizing a cost function of the form:

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{2.2}$$

The cost functions consists of two main terms. $h(\mathbf{x}(t_f), t_f)$ is a function that determines how much the total cost depends on the final state values and the final time. Whereas $g(\mathbf{x}(t), \mathbf{u}(t), t)$ determines how much the total cost depends on the values the states and inputs take on along the path.

A control sequence that minimizes the cost function over the time-horizon $[t_0 \rightarrow t_f]$, while respecting all constraints, is considered to be the solution to the optimal control problem. The corresponding minimum cost value J^{*1} is also called the 'optimal cost-to-go' and is defined as:

$$J^*(\mathbf{x}, t) = \min_{\mathbf{u}(t \rightarrow t_f)} J(\mathbf{x}, t, \mathbf{u}(t \rightarrow t_f)) \tag{2.3}$$

2.2. Discrete case: Bellman Equation

The Bellman equation is an equation that describes the optimal cost-to-go along the optimal control path. It is derived using the principle of optimality:

> *Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.* (Bellman [3])

The following example from Kirk [5] is used to demonstrate this principle in a more intuitive way:

[1]The * superscript notation indicates that the corresponding variable or function is optimal

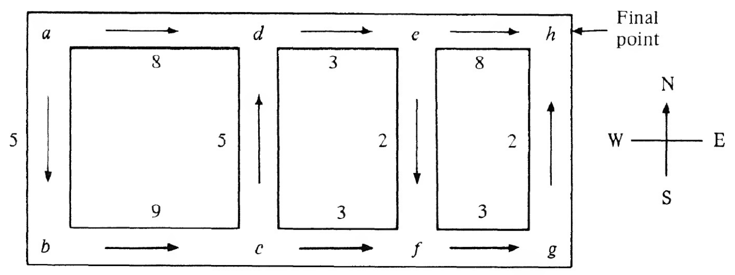


Figure 2.1: Available routes between states with corresponding costs (from [5])

Consider Figure 2.1. Say that one would like to travel from state a to state h in a minimum-cost fashion. The costs to travel from one state to an adjacent state are shown along the edges (e.g. to travel from a to d adds 8 to the total cost). A procedure to find the minimum-cost path could be to start at the target state and work backwards from there, as shown in Table 2.1.

Current state α	Input	Next State x_i	minimum cost from α to h via x_i	minimum cost from α to h	Optimal Input
g	N	h	2	2	N
f	E	g	$3+2=5$	5	E
e	E	h	8		
	S	f	$2+5=7$	7	S
d	E	e	$3+7=10$	10	E
c	N	d	$5+10=15$		
	E	f	$3+5=8$	8	E
b	E	c	$9+8=17$	17	E
a	E	d	$8+10=18$	18	E
	S	b	$5+17=22$		

Table 2.1: Example adapted from Kirk [5]

This table demonstrates the principle of optimality. I.e. the optimal cost to reach the target state from the current state is equal to the optimal cost to reach an adjacent state plus the optimal cost to reach the target state from the adjacent state. The global optimum is then found by selecting the input value that will lead to the optimal adjacent state.

Using Equation 2.2, Equation 2.3 and the principle of optimality the Bellman equation can be derived:

$$\begin{aligned} J^*(x,t) &= \min_{\mathbf{u}(t\to t_f)} \left(h\left(\mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_1} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt + \int_{t_1}^{t_f} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \right) \\ &= \min_{\mathbf{u}(t\to t_f)} \left(\int_{t_1}^{t_f} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt + \min_{\mathbf{u}(t\to t_f)} \left(h\left(\mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_1} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \right) \right) \\ &= \min_{\mathbf{u}(t\to t_f)} \left(\int_{t_1}^{t_f} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt + J^*(\mathbf{x}(t_1), t_1) \right) \end{aligned} \tag{2.4}$$

This equation allows to split a optimal control problem into smaller parts. But, in this form it can only be applied to discrete-time problems. The Hamilton-Jacobi-Bellman equation is a format of the Bellman equation which is intended for continuous-time problems.

2.3. Continuous-time case: Hamilton-Jacobi-Bellman Equation

Equation 2.4 in its current form can only be applied in discrete-time optimal control problems. However, in order to be able to apply it in continuous-time control problems the equation has to be derived in a different way:

Again, combining Equation 2.2 and Equation 2.3:

$$\begin{aligned} J^*(\mathbf{x}(t),t) &= \min_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le tf}} \left(\int_t^{t_f} g(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau + h\left(\mathbf{x}\left(t_f\right),t_f\right)\right) \\ &= \min_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le tf}} \left(\int_t^{t+\Delta t} g(\mathbf{x}(\tau)d\tau + \int_{t+\Delta t}^{t} g(\mathbf{x}(\tau)d\tau + h\left(\mathbf{x}\left(t_f\right),t_f\right)\right) \end{aligned} \tag{2.5}$$

Implementing the principle of optimality:

$$J^*(\mathbf{x}(t),t) = \min_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le t+\Delta t}} \left(\int_t^{t+\Delta t} g(\mathbf{x}(\tau)d\tau + \mathbf{J}^*(\mathbf{x}(t+\Delta t),t+\Delta t)\right) \tag{2.6}$$

Subsequently, the Taylor series is used to expand Equation 2.6 around $(\mathbf{x}(t),t)$.

$$\begin{aligned} J^*(\mathbf{x}(t),t) = \min_{\mathbf{u}(t)} &\left(\int_t^{t+\Delta t} g(\mathbf{x}(t))d\tau + J^*(\mathbf{x}(t),t) + \left[\frac{\partial J^*}{\partial t}(\mathbf{x}(t),t)\right]\Delta t \right. \\ &\left. + \left[\frac{\partial J^*}{\partial \mathbf{x}}(\mathbf{x}(t),t)\right]^T [\mathbf{x}(t+\Delta t) - \mathbf{x}(t)] + o(\Delta t)\right) \end{aligned} \tag{2.7}$$

Where $o(\Delta t)$ are the higher order terms. For small Δt:

$$\begin{aligned} J^*(\mathbf{x}(t),t) = \min_{\mathbf{u}(t)} &\{g(\mathbf{x}(t),\mathbf{u}(t),t)\Delta t + \mathbf{J}^*(\mathbf{x}(t),t) \\ &+ J_t^*(\mathbf{x}(t),t)\Delta t + J_{\mathbf{x}}^{*T}(\mathbf{x}(t),t) f(\mathbf{x}(t),\mathbf{u}(t),t)\Delta t + o(\Delta t)\} \end{aligned} \tag{2.8}$$

Where J_t and $J_{\mathbf{x}}$ are the partial derivatives of J w.r.t. t and $\mathbf{x}$, respectively. Next remove the terms that do not depend on $\mathbf{u}(t)$ from the minimization.

$$\begin{aligned} 0 = J_t^*(\mathbf{x}(t),t)\Delta t + \min_{\mathbf{u}(t)} &\{g(\mathbf{x}(t),\mathbf{u}(t),t)\Delta t \\ &+ J_{\mathbf{x}}^{*T}(\mathbf{x}(t),t) f(\mathbf{x}(t),\mathbf{u}(t),t)\Delta t + o(\Delta t)\} \end{aligned} \tag{2.9}$$

Then by taking the limit $\Delta t \to 0$ and dividing by Δt:

$$0 = J_t^*(\mathbf{x}(t),t) + \min_{\mathbf{u}(t)} \left\{g(\mathbf{x}(t),\mathbf{u}(t),t) + J_{\mathbf{x}}^{*T}(\mathbf{x}(t),t) f(\mathbf{x}(t),\mathbf{u}(t),t)\right\} \tag{2.10}$$

Which is known as the HJB because of the similarity to the Hamilton-Jacobi equation [6]. This differential equation has the following boundary condition:

$$\mathbf{J}^*(\mathbf{x}(t_f),t_f) = h(\mathbf{x}(t_f),t_f) \tag{2.11}$$

For convenience, the Hamiltonian is defined as:

$$\begin{aligned} \mathcal{H}(\mathbf{x}(t),u(t),J_x^*,t) &= g(\mathbf{x}(t).\mathbf{u}(t),t) + J_{\mathbf{x}}^{*^T}(\mathbf{x}(t),t)\,[f(\mathbf{x}(t),\mathbf{u}(t),t)] \\ \mathcal{H}\,(\mathbf{x}(t),\mathbf{u}^*\,(\mathbf{x}(t),J_{\mathbf{x}}^*,t)\,,J_{\mathbf{x}}^*,t) &= \min_{\mathbf{u}(t)} \mathcal{H}\,(\mathbf{x}(t),\mathbf{u}(t),J_{\mathbf{x}}^*,t) \end{aligned} \tag{2.12}$$

Which means that the HJB-equation can be written as:

$$0 = J_t^*(\mathbf{x}(t),t) + \mathcal{H}\,(\mathbf{x}(t),\mathbf{u}^*(\mathbf{x}(t),J_x^*,t),J_x^*,t) \tag{2.13}$$

2.4. Pontryagin's Minimum Principle

The calculus of variations is a branch within mathematics that has been used in solving particular optimization problems. The fundamental idea behind this theorem is that in variational problems the variations of a functional [2] must be zero on extremal curves. This is analogous to the theorem that within

[2]Functionals can be interpreted as 'Functions of Functions' as they refer to a class of functions that take functions as input. An indefinite integral is well known example of a functional [5]

functions the derivatives become zero on extreme points. The Russian mathematician Lev Pontryagin has developed a theory, called PMP, which provided a set of necessary, but not sufficient conditions for optimality. Being non-sufficient implies that these conditions must hold for the optimal solution of a problem, but may also hold for a non-optimal solution. Nonetheless, finding solutions for which the minimum principle is satisfied can be convenient for identifying candidates. [5]
In optimal control theory PMP can be helpful in solving for which inputs to a system a minimum cost will be found. In all practical cases of optimal control the inputs are always constrained. In these cases the cost function (Equation 2.2) can be augmented to have the constraints included inside the function:

$$J(\mathbf{u}) = h\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[f(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}(t)]dt \tag{2.14}$$

Where $\mathbf{p}$ is a vector, called the co-state, containing Lagrange multipliers[3]. From this equation conditions for optimality are derived again.

The fundamental difference in this derivation, as compared to the derivation of the HJB-equation, is that the calculus of variations is used to find the conditions for which a cost function becomes stationary to small perturbations of the control inputs. Whereas the HJB-equations starts from the principle of optimality as described in section 2.2 and expands this recurrence relation to a continuous-time form.
In this system the Hamiltonian is defined as:

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[f(\mathbf{x}(t), \mathbf{u}(t), t)] \tag{2.15}$$

Which is formatted differently than Equation 2.12, but has the same fundamental meaning. And the PMP conditions for optimality are [5]:

$$\left.\begin{aligned} \dot{\mathbf{x}}^*(t) &= \frac{\partial \mathcal{H}}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t) \\ \dot{\mathbf{p}}^*(t) &= -\frac{\partial \mathcal{H}}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t) \\ \mathbf{0} &= \frac{\partial \mathcal{H}}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t) \end{aligned}\right\} \begin{array}{l}\text{for all} \\ t \in \left[t_0, t_f\right]\end{array} \tag{2.16}$$

However, if these conditions are satisfied it only implies that an extremal has been found. In other words, it is not yet clear whether or not a maximum or minimum cost has been reached. Therefore the minimum principle states:

$$\mathcal{H}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right) \leq \mathcal{H}\left(\mathbf{x}^*(t), \mathbf{u}(t), \mathbf{p}^*(t), t\right) \tag{2.17}$$

Which ensures that the extremal is indeed at a (local) minimum cost. Equations 2.16 and 2.17 are necessary and sufficient conditions for optimality w.r.t. a cost function. In Appendix A a derivation of these conditions can be found.

In some specific optimal control problems these conditions are sufficient to find an analytical solution to the two-point boundary value problem. However in most cases there are too many unknowns and therefore numerical methods are required to find a solution.

2.5. Solving Optimal Control Problems

Simple optimal control problems may be able to be solved analytically, but most practical optimal control problems are too complex to be solved in this way. Properties that increase the complexity of a problem are for instance: dimensionality of the system (number of states and inputs), planning horizon, format of the cost function and constraints. In practice, the solution of such problems rely on numerical methods, which in turn rely on available computational resources.

Despite some exceptions, the solving approaches can generally be divided in three classes: Dynamic programming, direct methods and indirect methods [7–10]. Figure 2.2 shows a compact overview.

[3]Lagrange multipliers are convenient variables that are selected to define the augmented function in such a way that its derivative is zero for extremal points. Note that Equation 2.14 is independent of $\mathbf{p}$ because the equality constraint term, $f(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}(t)$, is zero along the entire trajectory. [5]

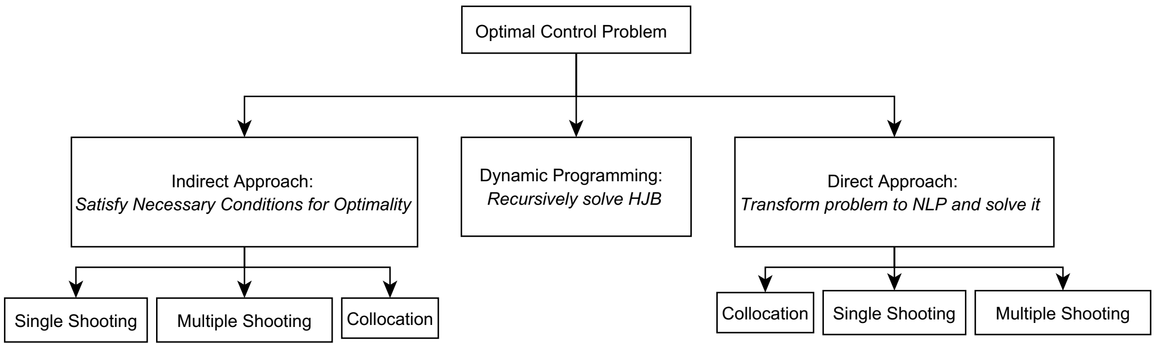


Figure 2.2: Overview of solving OCPs approaches

2.5.1. Dynamic Programming

In the dynamic programming approach attempts are made to find the optimal solution by solving the Bellman equation, or in the continuous-time case the HJB-equation (Equation 2.13). Unfortunately, there are only limited cases of optimal control problems that can be solved analytically. The vast majority must be solved in a numerical, recursive way. In that case the problem leverages the principle of optimality and is split into smaller sub-problems which are solved individually using a tabular approach. This process starts from the desired target state and steps back to the initial state, similar to the example given in section 2.2. Methods that use this approach are forms of *exact* Dynamic Programming.

A big drawback from this approach is that the cost function will need to be evaluated in the entire state-space to find the global minimum. The complexity, and therefore the required computational resources to solve the problem, scales exponentially with the dimensions of the problem.[3, 11].

Fortunately, there are methods that can make dynamic programming a feasible approach for certain control problems. ADP, for instance, is a class of dynamic programming that uses different methods to *approximate* the cost function (the HJB equation in the optimal control case) instead of calculating it exactly as in the aforementioned tabular approach. The ADP methods distinguish themselves from each other by the approximation method they use. Examples are: Neurodynamic Programming, HDP, Dual HDP, Action-Dependent HDP and Globalized Dual HDP. However, almost all methods use a reinforcement learning approach based on MDPs and/or ANNs. [11–13]. MDPs are processes that specify which decisions (control inputs) to make in semi-stochastic systems where the outputs are probabilistic.

2.5.2. Direct Methods

The class of direct methods approach the optimal control problem by transforming it into an NLP problem and subsequently solve it with one of the various well-known solver methods. Direct methods can deal more easily with different types of inequality constraints when compared to indirect methods, which are required to calculate co-state equations. Co-state equations are the equations that should solve the second necessary condition in Equation 2.16 and thus deal with the Lagrange multipliers.

The majority of off-the-shelf optimization tools used in finding solutions to optimal control problems make use of direct methods. The most common implementation make use of either (multiple) shooting or collocation methods.[8]

Direct Shooting

Direct shooting may in essence be one of the simplest forms of the solving methods. It is often compared to optimizing the aim of a cannon to hit a target. Which is an initial value problem. The process of optimizing this problem starts with guessing the initial state and control inputs (i.e. cannon angle and impulse magnitude). These values are then iteratively changed based on the results of the simulated trajectory.
This approach can be implemented in more complex optimal control problems by discretizing and sub-

sequently parameterizing the control space by piecewise functions. Finally, the resulting ODEs are solved by means of numerical integration and gradient descent methods [8].
A difference is made between single shooting and multiple shooting approaches. In multiple shooting approaches the time interval is divided into smaller sections. Each section is then solved by a single shooting method as described above. State continuity between the sections are enforced by adding the by adding the corresponding boundary conditions to the NLP formulation.

Unfortunately, this methods has shown to be rather sensitive to the initial conditions. To reduce this sensitivity the shooting method has been extended to multiple shooting. This technique splits the problem into smaller segments. The total problem will now increase in size because each new segments comes with new variables, constraints and boundary conditions. A direct result is that more computation time is required to solve the problem[14].

Direct Collocation
Collocation (sometimes also called Transcription) is a method very similar to shooting. In addition to the control space this method also parameterizes the state space. This implies that the controls and states are approximated by polynomial splines so that the differential equations can be represented by algebraic constraints. Each constraint must be satisfied at a specific collocation points. Off-the-shelf NLP-solvers exist that are able to solve these systems of equations.

Within direct collocation several classes exist such as trapezoidal-, orthogonal-, (pseudo)spectral-, and Hermite-Simpson collocation. The classes distinguish them self from one another mostly by the way the collocation points are chosen, the way the system is discretized and what kind of parameterization is used[10].

2.5.3. Indirect Methods
Approaches that attempt to satisfy the conditions for optimality stated by Ponyryagin's minimum principle (Equations 2.16 and 2.17) fall under the indirect methods. Contrary to direct methods, which convert the problem in a non-linear programming problem and attempt to solve the HJB equation explicitly. Fundamentally speaking, direct methods try to find the minimum of the objective function whereas indirect methods try to find the root of the necessary conditions.[14]

Generally, indirect methods start by creating the necessary conditions and then discretize them. A two-point BVP is obtained that must satisfy the boundary conditions for $\dot{\mathbf{x}}$ and $\dot{\mathbf{p}}$. The boundary conditions and the optimal control inputs are not solved simultaneously. Instead, the control inputs are found by locally minimizing the Hamiltonian (Equation 2.15) for each time step, which are subsequently substituted in the boundary condition equations and iteratively updated in the case the conditions are violated.

Similarly to the direct methods, indirect methods can be solved by shooting and collocation, but the main difference is that besides the states, also the adjoint variable (co-states) must be integrated. And more difficult, an initial guess for the Lagrange multipliers must be made. Which can be challenging since the Lagrange multipliers have no intuitive meaning. Therefore, it can be more troublesome to implement an indirect method as opposed to a direct method.[8, 10]

2.6. LQR Control
LQR control is a form of optimal control that can be applied to certain linear systems with quadratic cost functions. The solution to the problem is a feedback controller that minimizes the cost function.

The cost function typically looks like:

$$J(\mathbf{x},\mathbf{u}) = \int_0^\infty \tilde{\mathbf{x}}^T Q \tilde{\mathbf{x}} + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}} dt \tag{2.18}$$

Where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ are the errors to the reference state and input, respectively. Q and R are cost matrices related to the states and inputs.

It is assumed that the optimal cost-to-go is $J^*(\mathbf{x}) = \mathbf{x}^T P \mathbf{x}$. The solution for P can be found by solving the continuous algebraic Riccati equation.
It is found that the optimal feedback control law is:

$$\begin{aligned} \mathbf{u} &= u_0 + K\,(\mathbf{x} - \mathbf{x}_0) \\ K &= -\,R^{-1}B^T P \end{aligned} \tag{2.19}$$

Where B is the input matrix to the state-space system,[15, 16].

Foehn and Scaramuzza [16] have designed an LQR controller to find the optimal thrust and body rate inputs for trajectory tracking and hover conditions. In their work they needed to use the HDP approach of Al-Tamimi, Lewis, and Abu-Khalaf [13] to solve the problem of finding P.

To find the controller's parameters the solution for optimal control must be derived analytically from the differential equations that describe the system.
Therefore, for true mathematical optimal control, it is required that the system's equations are able to be solved analytically. Consequently, LQR control is typically used in relatively simple linear systems. Such as fast inner loop controllers for rotor control.
However, LQR can be effective in nonlinear systems as well. In these cases the system can be linearized around specific trim points of the states. [16, 17]

2.7. Chapter Summary

This chapter has explained two main approaches of finding optimal control with respect to a cost function. Namely, one approach that is derived from the classic "traveling salesman" problem, i.e. the HJB equation. And the second approach is PMP which uses calculus of variations to derive the necessary conditions for optimality.

These are just methods of defining optimal control problems. Despite that some particular problems can be solved analytically, the majority of practical problems will need a numerical approach to find a solution. These approaches can be divided in three classes: Dynamic programming, direct methods and indirect methods. Each of which comes with its own benefits and limitations.

Dynamic programming methods attempt to recursively solve the HJB equation and has as benefit that it searches the entire state-space for the optimal sequence of control inputs. Guaranteeing that the optimal solution is the global optimal solution. Despite that it is suitable for certain mixed integer and continuous problems, it suffers from the 'curse of dimensionality'. That is, the complexity (and thus in numerical approaches, the required computational power) increases exponentially as the problem's dimensions increase [3, 8]. Powell [18] even speaks of the "three curses of dimensionality" because this problem does not only apply to the dimension of the problem's state vector, but also to the dimensions of the possible inputs and outputs.

Direct methods transform the OCP to an NLP problem and have as advantage that the inequality constraints are more intuitive to handle than in the case of indirect methods. Therefore, there are numerous off-the-shelf software toolboxes available that can efficiently solve OCPs with a wide range of different structures. Software toolboxes such as IPOPT, MUSCOD, ICLOCS and ACADO are well known to have a good applicability.

Indirect methods attempt to find a control input sequence that satisfies the necessary conditions for optimality as stated by PMP. Similar to the direct methods, there is a range of software solutions available that are robust and computational efficient to solve a variety of optimal control problems. According to Biral, Bertolazzi, and Bosetti [8] there is no clear difference between direct and indirect methods in terms of accuracy and computational efficiency. This was based on experiments that show that differences in computation time for specific problem mainly depend on how the different solvers adopt the optimal control problem. It was shown that the discretized NLP from the direct method is almost identical to the BVP of the indirect method. However, defining a OCP in an indirect methods is more difficult because of the non-intuitive nature of the co-states.

Finally, LQR control as an optimal control approach was explained. LQR is always implemented as a feedback controller. This controller is designed in such a way that it minimizes a quadratic cost function and are most often used as inner loop controllers to linear systems.

3 Drone Racing

In this chapter an introduction to ADR is given. Even though drone racing with human pilots is still a rather recent phenomenon. Steps are being made to have editions where the human element is removed from the equation. Despite being primarily intended for entertainment purposes ADR seems to be the perfect environment to test the physical limits of a quadcopter and demonstrate the latest advances in autonomous systems.
Thanks to the competitive element of these events it seems to be the perfect platform to apply time-optimal control to. Moreover, the required aggressive and high-acceleration maneuvers put the on-board hardware and systems to the test. Particularly systems such as perception and state-estimation suffer in performance in high velocity and low-light conditions.
Naturally, the relevant developments will not only prove to be useful in ADR, but are likely to be beneficial in other fields such as military, search & rescue or any other aviation/robotics oriented fields.

In human-piloted drone races the pilots typically have to guide their drone through a course with gates and obstacles. Naturally, humans have the ability to quickly recognize obstacles and steer their drone according to a trajectory that they have in mind. Even in dynamic environments humans can quickly adapt their strategy accordingly.
In ADR on the other hand, it is not that simple unfortunately. There are certain disciplines required to successfully follow the same steps as a human pilot.

3.1. Chapter Outline

This chapter starts in section 3.2 with a brief overview of the current esteemed ADR events during which the newest relevant developments in robotic systems are demonstrated.
In the subsequent sections the different subsystems that enable ADR are being addressed starting with automatic trajectory generation in section 3.3.

Where a human can recognize a gate and its current position relative to the drone almost instantly, an autonomous drone depends, among others, on subsystems that are responsible for perception and state estimation. Which current state of affairs will be expanded upon in sections 3.5 and 3.6. Additionally, automatic trajectory generation methods are required to efficiently plan a route through the course, which ideally should be the fastest path possible. This will be elaborated on in section 3.3. Finally, once the trajectory is known the quadcopter's rotors should be actively controlled in such a manner that the quadcopter robustly tracks the desired trajectory. section 3.4 explains the different methods that enable this.

3.2. Competitions

In this section a summary is given of the most prestigious event during which the latest developments in autonomous drone racing are being showcased.

3.2.1. Alphapilot/AIRR

In 2019 Lockheed Martin and the DRL came together to organize the first edition of AIRR, also named Alphapilot. In this competitions teams have to develop the control pipeline for a 3 kg quadcopter and compete in multiple rounds at different race tracks. Each team had to work with identical hardware, but were given complete freedom to develop the flight control software.
The quadcopter was outfitted with 4 cameras and an Nvidia Xavier processing unit alongside a IMU and laser range finder.

The teams were not able to work with the quadcopter's hardware directly, but were given a development kit which consisted of a desktop pc and the NVIDIA Jetson AGX Xavier GPU to develop the software on and test the code in a simulation environment. They were responsible for creating a system that could perceive gates, which were distributed along the course, and the flight control system that would steer the quadcopter to fly through these gates.[2]

3.2.2. IROS

Since 2016, an ADR competition has been a part of the yearly recurring IROS conference. The ADR is seen by multiple research groups as a prestige opportunity to demonstrate the latest developments in (small) UAVs. Generally, the competitions consists of a obstacle course that the UAV must pass through in the shortest time. Both static and moving obstacles are placed in the course to test the autonomous systems for robustness in dynamic, cluttered environments.[19, 20]

3.2.3. IMAV

The IMAV is a yearly event during which the newest development on small UAVs are displayed. Each year, several challenges, each with a different focus are given to teams to participate in. For instance, one challenge focuses on the small and efficient drones, where the other focuses on stability in a turbulent environment.[21]

3.3. Trajectory Generation

In this section various methods of generating trajectories will be discussed. A trajectory is, in the context of quadcopter control, in most cases not simply a series of positional waypoints, but actually a description of the path that some of the states and/or their derivatives will follow. When generating a trajectory it is important to incorporate the quadcopter's dynamics such that the quadcopter is actually able to reach the sequence of state values.

Especially in dynamic environments it is crucial that a quadcopter can adapt to environmental changes and obstacles. One way of addressing this is by re-planning the trajectory during flight which is the principle of MPC.
One particular example is the work of Mueller, Hehn, and D'Andrea [22]. In their paper an efficient approach to an MPC implementation is introduced that allows a quadcopter to intercept a thrown ball and bounce it to a target position. In order to do so the ballistic trajectory of the ball must be predicted continuously. Based on this, an optimal flight path will be calculated to intercept the ball and simultaneously bounce it to the target. Due to the low computational effort of this method experiments have shown that thousands of potential ways to hit the ball can be calculated and evaluated at a rate of 50Hz. Despite that the calculations are performed off-board by a consumer-spec laptop, it shows great potential nonetheless for on-board implementation of MPC cases that do not require such a shear amount of path evaluations as in this example.

3.3.1. Minimum-Snap

There are numerous approaches in trajectory generation, but one of the most cited papers in this field is the work from Mellinger and Kumar [23]. Who have developed a method to generate minimum-snap[1]trajectories subjected to pre-defined constraints in real-time.
One appeal of minimum-snap trajectories is that its smoothness benefits the measure quality of the on-board sensors and that no aggressive control inputs are needed [24]. However, a more important benefit is that the required body moments can be derived analytically from the trajectory if the positional snap values are known as well as the second derivative of the yaw. Which will be used as feedforward

terms in the tracking controller that Mellinger and Kumar have created, as will be explained in subsection 3.4.1.

The trajectories are generated by formulating the minimum-snap optimization as a QP problem, constraints are added that force the trajectory to stay within a small corridor around the waypoints. In the formulation of the trajectories the times of arrival at each waypoint need to be specified beforehand. In order to further optimize the trajectory for minimum snap, an additional optimization problem has been added to find the optimal segment durations (the paths between the waypoints are called segments) while keeping the final time of arrival constant. Which is solved using a constrained gradient descent method.
Finally, when the final trajectories are known, the required control inputs can be derived by leveraging the differential flatness[2]property of the system.

In their experiments they have shown that trajectories can be generated in real-time and fast enough to adapt to dynamic obstacles. Flying through a falling hoop for instance. However, the experimental setup consisted of an external motion capture system to estimate position, velocity and attitude. Additionally, it should be noted that the trajectory generation was performed off-board and that this method does not yield time-optimal trajectories.

In the work of Richter, Bry, and Roy [26] the method of minimum-snap trajectory generation from [23] is revised. An issue of Mellinger's and Kumar's method is being addressed in which the QP becomes ill-conditioned in the case too many segments, high-order polynomials, or widely varying segments times are concerned. This is done by reformulating the constrained QP from [23] as an unconstrained QP which solves for the endpoint derivatives directly instead for the polynomial coefficients. This yields a more numerically stable optimization approach than the original. Moreover, an ”aggressiveness” parameter is included which allows to put a cost on the total trajectory time. By increasing this parameter's value the generated trajectory becomes less snap-optimal and more time-optimal.

3.3.2. Cluttered Environments

The aforementioned trajectory generations methods allows the inclusion of corridor constraints to guarantee that the resulting trajectory remains within certain positional bounds and does not collide with a fictional wall. However, these corridors will need to be defined beforehand which requires manual effort. Especially in unknown or dynamic environments, which is very relevant in ADR, it is of importance that trajectory generation methods can automatically adapt to obstacles.

Landolfi et al. [27] have addressed this by combining and extending the work of Mellinger and Kumar [23] with a SE-SCP algorithm, that originates from the research of Baldini et al. [28]. This algorithm is used to find the shortest path and create an initial set of consecutive waypoints in a cluttered environment. The SE-SCP algorithm can be seen a spherical version of the well-known RRT methods, which were originally used for path planning purposes, but have proven to be beneficial for searching high-dimensional spaces as well [29, 30]. An example of the working principle of SE-SCP can be seen in Figure 3.1.
Once the initial waypoints are known, a minimum-snap trajectory is generated in a similar way to the aforementioned approach of unconstrained QP that Richter, Bry, and Roy [26] have used as well. Furthermore, wind has been included the dynamical model and an $\mathcal{L}_1$ adaptive controller has been implemented that introduces robustness to wind and mass changes. Although no practical experiments have been performed. This method has been verified in a sophisticated simulation environment, using a dynamical model based on the CrazyFlie 2.0 quadcopter [31].

Penin et al. [32] created have enhanced time-optimal trajectory generation to keep visual cues inside view during a maneuver. This particularly useful to quadcopters that use a monocular camera for VIO.

[1]Snap is the fourth derivative of position.

[2]Differential flatness is the property of a system in which states can be expressed explicitly in terms of inputs and states or derivatives of the states. An advantage of such systems is that the required sequence of input values can be found algebraically from the state trajectory. [25]

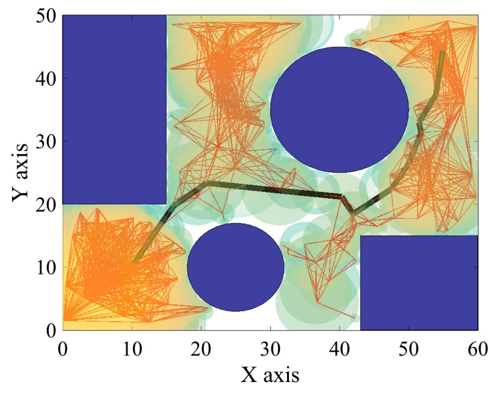


Figure 3.1: Here SE-SCP is used to find the shortest path from the lower-left corner to the upper-right corner. (from Baldini et al. [28])

In their approach a standard NLP, time-optimal control formulation is extended to include the feature points to always lie within the field of view of the camera. Subsequently, it is solved using a SQP method as used in NLOPT [33]. A downside is that the resulting trajectories are expressed in B-spline polynomials, limiting the type of motion that quadcopter can perform. Moreover, the visibility constraint is a hard constraint which puts a cap on the maneuverability.
Falanga et al. [34] addresses these issue in their work. They have created a framework in which the cost function couples perception objectives with action objectives. Allowing to dynamically optimize the priority put on perception and on the flight objective. The problem is solved using an SQP and is fast enough to run in real-time and on-board, enabling it to function as an MPC.

3.3.3. Machine Learning Approaches

Using machine learning principles to generate trajectories is an approach that has recently become more popular. At the core, the idea is to generate training and validation data off-line by means of one of the more computational expensive numerical solvers and use it to train an ANN in simulation and/or in real flight. The goal is that these ANNs are able to produce optimal trajectories and/or control inputs independently.

For example, Tang, Sun, and Hauser [35] have come up with an approach in which they first generate a library of optimal trajectories off-line and use them to train a neural network that is able to output trajectories on-line, using the quadcopter's current state as input. Additionally, since these outputs may not fully satisfy all constraints due to approximation errors, the trajectories are refined by a single iteration of a quadratic-programming solver. Although a network needs to be trained extensively this system is eventually able to map trajectories in a matter of milliseconds. Which is fast enough to run in real-time and can therefore be used as an MPC.
The solutions have been tested on a CrazyFlie 2.0 quadcopter. That is, the neural network is executed externally and control commands are transmitted to the quadcopter. Moreover, their solution is compared to different approaches, such as minimum-snap and three different NLP methods. In Table 3.1 an overview can be found of the required computation time and resulting cost (lower is better) for each of these methods.
It was found that the solutions with NLP solvers generally yield a lower cost value. However, they also require significant more calculation time, whereas the paper's ANN method only takes 1.8ms for only a slightly higher cost.
The minimum-snap method comes close with a computation time of 10.21ms. In experiments it was found that the Crazyflie had more trouble to track these trajectories than the trajectories from the paper's method because the paper also adjusted the cost function to penalize rapid changes in the rotors' speed. Which is beneficial to quadcopters with a lower thrust-to-weight ratio such as the Crazyflie. However, for all tracking experiments a PID positional controller was used to track the trajectory, which tends not to lead to optimal tracking performance.

Li et al. [36] solve this issue by developing a neural network that solves for a more simple scenario.

	NN+OSQP	Minimum-Snap	SL+NLP	NNOC	NN+NLP
Time [ms]	1.80	10.21	382.9	194.7	131.1
Cost	8.73	8.88	8.64	8.64	8.64

Table 3.1: Approaches Comparison (from Tang, Sun, and Hauser [35])

Instead of producing time-optimal trajectories the network is trained to output optimal thrust and optimal pitch rate commands. With practical experiments, this controller is compared to a method that uses differential flatness to find the control values as in [23].
Results have shown that the neural network approach can make the quadcopter reach its target almost 60% faster than the differential flatness approach. However, since only thrust and pitch rate commands are calculated, this method can only approach optimal control in 2D scenarios.

More complex maneuvers were performed in the work from Kaufmann et al. [37]. They have demonstrated a deep learning approach to extreme autonomous acrobatic maneuvers. In this method a neural network is trained not to generate trajectories, but to accurately track reference maneuvers consisting of extreme accelerations (up to 3g). In addition to the reference trajectory the ANN architecture also processes on-board sensor measurements such that the quadcopter does not only rely on traditional state estimation methods, which typically suffer from reduced accuracy during high acceleration maneuvers.

Similar research has been performed by Camci and Kayacan [38], who have developed a method in which reinforcement learning is used to learn motion primitives that can be used in real-time motion planning. Using this approach they are able to quickly search and plan specific swift maneuvers in an existing trajectory.

Loquercio et al. [39] have developed a pipeline in which a convolutional neural network is trained on raw monocular images from a simulation environment to perceive the gates it should fly through and to output a positional waypoint and a desired velocity. Subsequently, a short minimum-jerk trajectory is generated from the current position to the waypoint. Although the training can only be performed off-line. The neural network can run in real-time and on-board a real quadcopter. Experiments have shown good robustness to changing lighting conditions and moving the gates.

3.3.4. Solving Time-Optimal Principles

Hehn and D'Andrea [40] generate a time-optimal trajectory by solving the conditions for optimality as stated by PMP (see section 2.4). The $x-, y-$ and z-coordinates are decoupled and optimal state-to-state trajectories are solved for each dimension separately, following the *direct adjoining approach* for optimal control problems with state constraints, as described by Hartl, Sethi, and Vickson [41]. They have tested this method in practice, using a Ascending Technologies 'Hummingbird' quadcopter. The trajectories and control inputs were calculated by an external computer in real-time and transmitted to the quadcopter during flight. This method has shown to be computational efficient enough to be able to re-plan trajectories at a rate of 50Hz in this experimental setup. Figure 3.2 shows what a typical time-optimal input control sequence looks like for one of the dimensions. This characteristic shape is called bang-zero-bang control and is often the solution to linear time-optimal control problems [42]. One downside of Hehn and D'Andrea's approach is that the maximum allowed acceleration must be specified beforehand and is fixed. Furthermore, some aerodynamic effects are neglected which is likely the cause for trajectory deviations during decelerations.

Knowing beforehand whether a solution has a bang-type solution can simplify the solving process drastically. The OCP reduces in such a case from a continuous control input problem to one that is discrete. One only need to solve for the optimal switching time instances. Between those instances the control input is constant. Lucas and Kaya [43] have been working on calculating switching times in optimal control problems where bang-bang controllers are used to control nonlinear systems. Their work has resulted in a numerical algorithm, based on Newton's method, that increases computational efficiency in solving the time-optimal bang-bang problems with one or two control inputs.

Similarly, Shen and Andersson [44] have been working on minimum-time control of bang-bang controllers. In their work they start with a second order system and derive the necessary conditions for optimal control using Pontryagin's Minimum Principle. From which follows that there is no analytical solution to calculate the optimal switch time. However, from the system dynamics they have derived an affine mapping that relates the switch time to the final time. Using this mapping they can represent the final time and switch time curves in a phase plane plot. An algorithm has been created that finds the crosspoint of these curves numerically which yields the solution to both the optimal switching time and the time of arrival. Despite the fact that this research has been performed with the control of servos and actuators in mind, the methodology could be applied to quadcopter control as well.

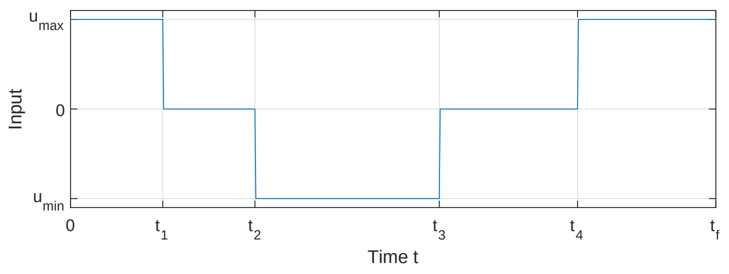


Figure 3.2: Characteristic time-optimal control input sequence (adapted from Hehn and D'Andrea [40])

As a followup, Hehn, Ritz, and D'Andrea [45] continued on extending this work to a benchmarking tool. In the algorithm there is no priority set on computational efficiency, but the goal is to create a robust comparison method to identify the lower bound quadrotor performance that can be used to compare other time-optimal controllers or see the influence of certain physical parameters of a quadcopter. In the paper a two-dimensional, first-principles quadcopter model is used. Which means that only two-dimensional time-optimal trajectories can be calculated with this tool and that lesser dominant dynamics, such as rotor dynamics and aerodynamic drag, are neglected. Nonetheless, the methodology has been validated by performing tests with real quadcopters which were able to perform maneuvers that the tool had calculated.

3.4. Control

In traditional control solutions, such as PID controllers, quadcopters are steered from waypoint to waypoint by having an outer control loop map a positional error to a desired heading and velocity and inner control loops take care of the angular rates and propeller power setpoints.
In optimal control theory on the other hand, solutions often result in trajectories that can be seen as time-continuous or discrete descriptions of the quadcopter's position, attitude and velocity as a function of time. Making quadcopters fly in a time-optimal manner therefore deals with the problem of finding the time-optimal trajectories, as discussed in section 3.3, but also letting the quadcopter track the trajectories accurately.

3.4.1. Trajectory Tracking

In drone racing human pilots are able to fly aggressive trajectories that traditional inner PID control loops may not be able to track accurately. Faessler, Falanga, and Scaramuzza [46] have created an angular rate controller that is based on LQR methods. LQR controllers for this purpose have been researched before by Kawai and Uchiyama [47] and Wang, Ghamry, and Zhang [48], but only dealt with single-axis angular rate control. This paper on the other hand, describes an method in which an LQR solution for the coupled angular rates in all three dimensions. Moreover, an iterative thrust mixing scheme has been developed that, in case of propeller power saturation, prioritizes between generating torques or generating total thrust in such a way that tracking errors are minimized. Experiments with a custom-built quadcopter have shown that, compared to a state-of-the-art PID controller [49], angular rate tracking errors have reduced with almost 50% and positional tracking errors with more that 25%.

As explained in section 3.3, Mellinger and Kumar [23] have created a method to generate minimum-snap trajectories while incorporating the dynamics of the vehicle. In addition to generating a minimum-snap trajectory, they also make use of the model's differential flatness property to analytically derive the control inputs to track the reference trajectory.
It is shown that from the positional acceleration and the yaw reference values the thrust vector can be derived. Moreover, from the jerk and yaw rate references the attitude rate can be found. And finally, from the reference snap and yaw acceleration the attitude angular accelerations are found. These values are fed to the inner-loop controllers that then take care of the individual propeller speeds.

Despite that the optimal control inputs are known now, a feedback controller is still required to account for disturbances and initial state inaccuracies. Therefore, the control inputs are used as feedforward terms in a PD controller. In which the tracking error governs the desired thrust force vector as shown in Equation 3.1.

$$\mathbf{F}_{\text{des}} = -K_p\mathbf{e}_p - K_v\mathbf{e}_v + mg\mathbf{z}_w + m\ddot{\mathbf{x}}_r \tag{3.1}$$

Where $\mathbf{e}_p$ is the tracking error in position, $\mathbf{e}_v$ is the tracking error in velocity and $mg\mathbf{z}_w$ accounts for gravity. The feedforward term, $\ddot{\mathbf{x}}_r$, is the acceleration from the reference trajectory. Subsequently, the desired attitude is found from the body axes which can be derived from the reference trajectory's yaw angle and assuming that the body's z-axis aligns with the desired thrust vector. Inner PD control loops then relate the attitude error to propeller speeds.
This method has been demonstrated by flying a quadcopter through a set of circularly positioned hoops, reaching speeds up to 2.6 m/s.

One shortcoming on the method presented in this paper is that the dynamic model does not include any aerodynamic forces acting on the body which will lead to trajectory tracking inaccuracies at higher flying speeds. Luckily, this has been addressed in the work of Faessler, Franchi, and Scaramuzza [50] in which they extend the dynamic model to include linear aerodynamic rotor drag terms. They prove the differential flatness of this extended model such that the control inputs can again be derived analytically from a reference trajectory. They created a tracking controller that is very similar to the controller used by Mellinger and Kumar [23] as explained above. It only differs by accounting for aerodynamic effects in calculating the desired thrust and also in using the reference angular rate and accelerations as feedforward terms (Mellinger and Kumar only use the reference position, speed and acceleration values).
Experiments have shown that tracking performance already visibly improves from a speed of 0.5 m/s onward, compared to the model without drag. The RMS tracking error reduces with 50% overall with speeds up to 5 m/s. Flight tests were performed for a circular trajectory and for a 'Gerono lemniscate' trajectory (a characteristic 8-figure).

It was found that tracking performance increased when the drag parameters were identified for the particular trajectory to be flown. This is an issue since it is not possible to find one set of drag parameters that optimizes tracking performance for all types of trajectories. The cause for this is the assumption that rotor drag can be modeled as a linear system which is not sufficient to accurately model all complex aerodynamics.

Tal and Karaman [51] have solved this in their control pipeline which combines feedforward terms from differential flatness with INDI. INDI is a model-based control method that inverts the dynamic model to calculate the required increments for states and control inputs to achieve the desired states [52]. This method is known to be very robust against model inaccuracies and external disturbances. In fact, the dynamic model used in the paper only incorporates one external force to account for all collective disturbances such as aerodynamic forces, shunting the need for identifying the aerodynamics while still achieving high tracking performance. An overview of the system can be seen in Figure 3.3.

Similar to Equation 3.1 a PD controller is used to find a desired acceleration vector. From which the desired thrust, roll and pitch angles are found using INDI, while differential flatness is used to find the desired Euler rates and accelerations. Then a sequence of NDI and INDI is used again to find the desired body angular accelerations and body moments, respectively. By simple inversion of the quad-

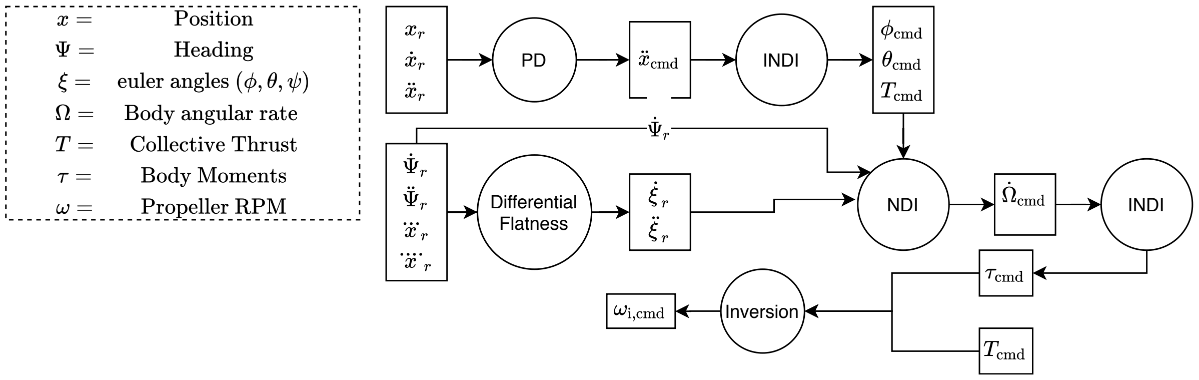

$x =$ Position
$\Psi =$ Heading
$\xi =$ euler angles (ϕ, θ, ψ)
$\Omega =$ Body angular rate
$T =$ Collective Thrust
$\tau =$ Body Moments
$\omega =$ Propeller RPM
x_r $\dot{x}_r$ $\ddot{x}_r$
PD
$\ddot{x}_{\text{cmd}}$
INDI
ϕ_{cmd} θ_{cmd} T_{cmd}
$\dot{\Psi}_r$
$\dot{\Psi}_r$ $\ddot{\Psi}_r$ $\dddot{x}_r$ $\ddddot{x}_r$
Differential Flatness
$\dot{\xi}_r$ $\ddot{\xi}_r$
NDI
$\dot{\Omega}_{\text{cmd}}$
INDI
τ_{cmd}
T_{cmd}
Inversion
$\omega_{\text{i,cmd}}$

Figure 3.3: Tal and Karaman's control pipeline (derived from Tal and Karaman [51])

copter's inertia the propeller speeds can be derived from the desired body moments and thrust.
This controller has been put to the test by letting a 1kg quadcopter follow aggressive trajectories indoors. For instance, a flight in which an Lemniscate trajectory was flown can be seen in Figure 3.4. The proposed controller only The robustness of this controller was demonstrated by adding a drag plate to the quadcopter, effectively changing its aerodynamic properties by an unknown amount. Speeds of up to 8m/s and accelerations of 2g were achieved, but the maximum RMS tracking error was only 3.9 cm for the proposed controller and 4.5 cm with the drag plate attached.

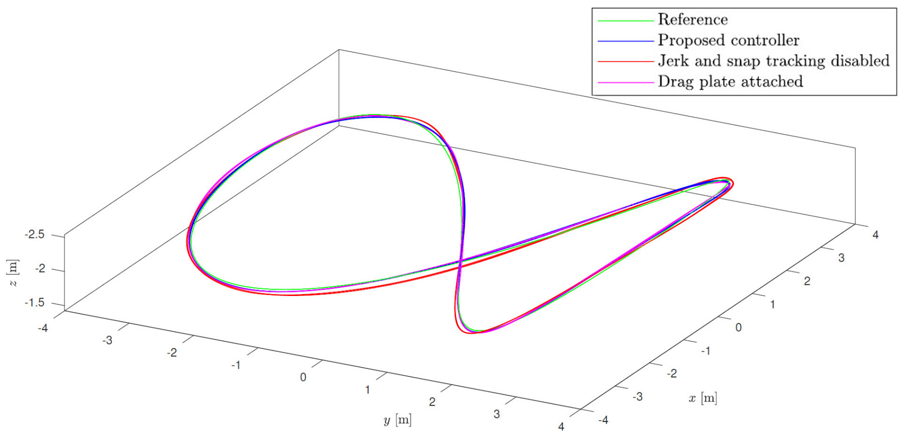


Figure 3.4: Lemniscate trajectory tracking (from Tal and Karaman [51])

3.5. Perception

The perception subsystem is responsible for making a quadcopter perceive elements from its environment. Although there is some overlap between perception and state estimation this section will focus on the different vision-based techniques that are currently applied to quadcopters.
In the context of ADR perception systems have the following two important functions:

- VO
- Obstacles & Gates recognition

VO is crucial in fast and accurate estimation of the quadcopter's motion and/or attitude. In drone racing GPS is often not accurate and fast enough due to the races often being held indoors and requiring both high-speed and high-precision maneuvers. Furthermore, the race often contains competitive elements that require to navigate through gates and avoid (moving) obstacles. Therefore, the quadcopter also relies on perception systems to quickly and accurately recognize these objects such that a control policy can be designed to react accordingly.

3.5.1. Visual Odometry

VO is the process in which the quadcopter derives its position and attitude using only one or a sequence of image frames. One speaks of VIO when VO is combined with IMU measurements. The typical VO pipeline consists can be seen in Figure 3.5. [53]

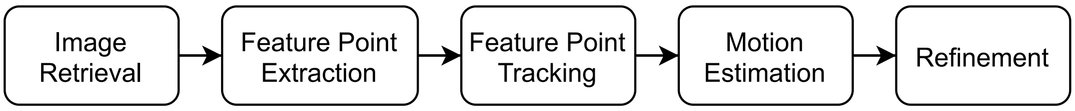


Figure 3.5: VO pipeline

For image retrieval one can choose to use a monocular camera, a stereo camera, or even more. Stereo odometry has as benefit that depth can be estimated from two frames at a single instant in time. Depth can still be estimated in monocular cases, but must be determined from relative motion of the image's elements between frames. Only relative depth between elements in the frame can be estimated. To find the absolute scale can only be determined if the true dimension of an element in the frame is known, or from fusing with measurements from other sensors such as the IMU or laser rangers. [54].

The second step consists of extracting feature points from the frames. This process searches the image frames for patches with characteristic patterns, often based on gradients in pixel brightness. Corners are for instance typical elements that can be unique distinguished within a frame thank to their high contrast in brightness gradient.
The goal is to detect the same feature points in different images independently, such that its motion between frames can be estimated. The area around each feature point is defined by a descriptor. For good matching accuracy the descriptor must allow each individual feature point to be described reliably and distinctively.
One of the most popular descriptors are SIFT [55] and SURF [56] which are both invariant for scale and rotation. Other descriptors, such as BRISK and ORB [57, 58], have been developed to improve computational efficiency, accuracy or robustness. Mukherjee, Jonathan Wu, and Wang [59] have carried out an extensive comparison study on various well-known feature point detectors and descriptors, evaluating each method for robustness, accuracy and efficiency.

The next step is to match feature points between image frames. The simplest approach is brute-force matching, in which each feature point descriptor from one frame is compared to each descriptor in the subsequent frame. Feature points that are most similar are considered to be matches. However, the number of comparisons scales quadratically with the total amount of feature points, making computation time a concern. Therefore it may be beneficial to use an indexing system to distribute all the descriptors across, based on the description properties, such that for each feature point the search range is limited. Even faster, one can make use of secondary motion estimations (using the IMU for instance) to predict in which area of the next frame each feature point is expected to be, narrowing down the search-area.
Unfortunately, there is always a risk of having false-positive. These mismatches can for instance be caused by blur, image noise and occlusions. There are methods to remove such outliers. RANSAC is a well-known method that is well suited this case [60]. In each iteration RANSAC selects a random set of samples, fits a model to these points and finally labels all samples that deviate too much from the model as outliers. [61].

The fourth step is to calculate the camera's motion between image frames from the matched features. Which method is used depends on whether the feature points are specified in 2-D or 3-D.

2-D to 2-D
In this case the feature points in two subsequent images are all specified in 2-D. The transformation between the two frames can be found by finding the "essential" matrix. This matrix contains the geometric relations of the two frames and can be decomposed in a translation matrix and a rotation matrix. In the 2D-2D case the essential matrix is found by using the epipolar constraint. Which determines the line on which the matched feature point lies on in the subsequent image, as illustrated in Figure 3.6.

Defining the constraint for at least five matching pairs gives a solvable system of homogeneous equations to which the essential matrix is the solution. Subsequently, the translation and rotation matrix can be decomposed from the essential matrix, describing the relative motion of one image to the other. [54]

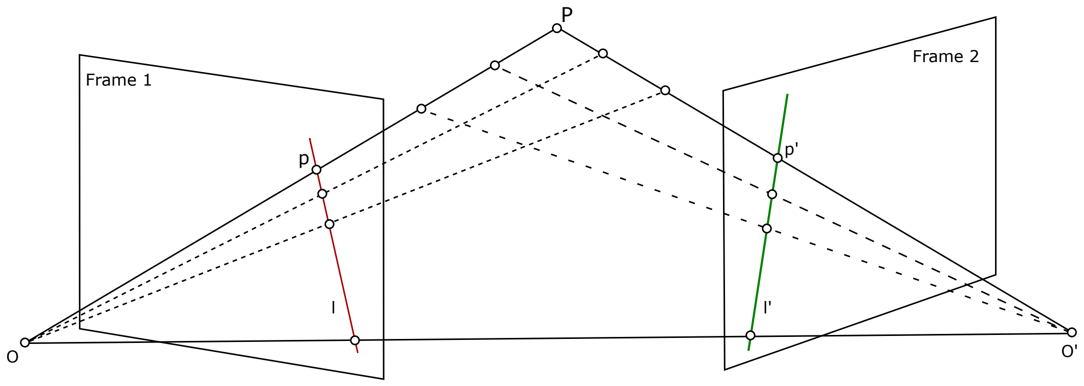


Figure 3.6: Feature point P is projected as p on frame 1 and as p' on frame 2. The epipolar constraint states that potential matches for p must lie on l' and potential matches for p' must lie on l. (adapted from Zhang et al. [62])

3-D to 3-D
3-D to 3-D motion estimation is possible when two stereo image pairs are available. The all feature points can now be specified in 3-D coordinates. In this case motion can be extracted by first triangulating the matched features. Subsequently, a matrix is calculated that minimizes the sum of L_2-distances between the matched feature pairs, which is theoretically possible with having only three matched pairs. The translation and rotation can then be extracted from this characteristic matrix. [54]

3-D to 2-D
On the 3-D to 2-D case the feature points in the older frame are specified in 3D (from a previous motion estimation for instance), but are specified in 2-D in the newer image frame. This particular problem of estimation the camera's pose is called PnP and can be solved if there are 3 matching pairs at a minimum (P3P). However, having only 3 matches yields 4 possible solutions, additional points help in removing perspective ambiguities in this case. An efficient way of solving P3P is described by Kneip, Scaramuzza, and Siegwart [63].

3.5.2. Object Recognition

In drone racing navigating through gates is one of the aspects that make ADR difficult. The system must recognize gates quickly and reliably such that the autopilot can respond accordingly. The high-velocity travel and changing lighting conditions makes this especially challenging. Object recognition is clearly not only of essence to ADR but of even more importance in autonomous driving. In which object recognition has more complex goals, from recognizing roads, cars and people to detecting and interpreting traffic signs. On top of that, the consequences of failure are more severe and therefore the reliability of the object recognition systems must be significantly higher than in ADR.

Classical Computer Vision Approaches
In recent years there have been many advances in AI approaches to object recognition and classification. But before AI became feasible quadcopters had to rely on more traditional computer vision methods and are still utilized in small quadcopters with limited computational power [64].
Such methods generally relied more explicitly detecting certain obvious characteristics of an object, such as color or shape. That is, the user has to manually specify beforehand which characteristics to look for.

For instance, gates are often rectangular. In this case a method such as the Hough transform may be useful to find all rectangular shapes in an image. A Hough transform can be used to find many different kinds of shapes under the condition that the shape in question can be parameterized, such as circles, straight lines and sinusoidal shapes [65]. The result is a mask per that indicates per frame which pixels belong to the shape. Using edge and corner detectors such as the Canny edge [66] detector and Harris corner detector [67] can be useful to give more meaning to such mask by indicating where the detected lines start or end, and by providing the gate's corner locations, from which information about relative pose and perspective can be derived.

For example, Jung et al. [68] have presented a gate detection method for the 2016 IROS ADR Challenge. Their approach starts with detecting the edges of the gates, based on the bright orange color of a gate and using a canny edge detector. The next step is to find the contours of the gate by dilating the edges. Once the width and height of the box were known the center point can be derived to which the quadcopter was steered.

One year later, during IROS 2017, Li et al. [69] presented the 'Snake Gate' method to detect gates. Similarly to Jung et al. [68], their approach was also based on the fact that the gate was bright orange of color. But contrary to their edge detection method, they randomly search for points that match the gate's color and subsequently search the adjacent pixels for continuity of the gate, in an up-down, left-right fashion. Similar to the popular 'Snake' video game. Once the gate's corners are known the camera's pose with respect to gate is calculated, using measurements from the onboard AHRS.

Convolutional Neural Networks
The more AI approaches that have gained popularity over the recent years most often deal with training neural network to distinguish gates in an image frame. CNNs may be the most often utilized in object recognition approaches. CNNs are ANNs which contain at least one convolutional layer. A convolutional layer is specially designed to process data from multiple arrays as input and is characterized by some neurons sharing weights in form of a so-called convolutional kernel, or filter. At each neuron the input data is convolved with local kernels before being passed on to the next layer. In most architectures a convolutional layer is often followed by a pooling layer that combines the output of a few neurons into one, decreasing the dimensions of the network and consequently making the network less sensitive to small variations of image features. A typical CNN architecture can be seen in Figure 3.7.

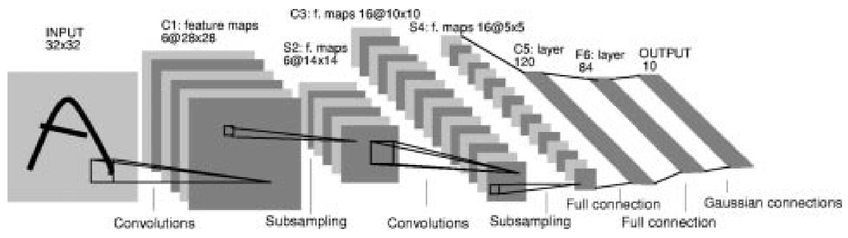


Figure 3.7: Example of a classic CNN architecture. This particular one is developed by LeCun et al. [70] to recognize handwritten characters. (from [70])

A popular training method applied to CNNs is error-backpropagation. In this supervisory approach, the partial derivatives of the error of the objective function with respect to the trainable parameters is calculated. Subsequently a gradient-descent scheme is used to iteratively update each parameter. A large dataset of labeled example images is required to evaluate the network and to quantify the partial derivatives, at each iteration.[71]

CNNs architectures often consist of more than one convolutional layer. Each of which applies convolution to the output of the previous layer. Interestingly, converged kernels give insight on how detailed patterns are built-up from more simple features. It is found that the kernels of the first convolutional layer gives strong responses to basic shapes such as edges and gradients, whereas the responses in deeper layers are increasingly more complex and specific to the object of interest. This cascade of increasing complexity allows for instance to robustly distinguish the face of a dog from the face of a cat [72, 73].

In the context of ADR, Jung et al. [74] have demonstrated the use of a CNN that runs on-board and reliably detects the center point of a single gate. The CNN was use the Single Shot MultiBox Detector approach from Liu et al. [75], but was modified by using a different CNN architecture based on the design of Krizhevsky, Sutskever, and Hinton [73]. However, in order to increase execution speed it was necessary to remove higher-level convolutional layers. Lowering the average precision, but allowing close to 30 fps processing speed.

Kaufmann et al. [76] have created an even more advanced pipeline. In their approach they use a CNN from [77] to extract features from images, subsequently a multilayer perceptron network estimates the relative post of the gate. The pose information is used by a MPC to plan a trajectory through the gate. Members of this team also participated in the 2019 Alphapilot Challenge. Their entire pipeline is presented in [78]. For gate detection they moved away from using a CNN to estimate gate pose directly, but instead used a segmentation approach to identify the four corners of the gates. Subsequently they have trained a network that can find part affinity fields based on the work of Cao et al. [79], which are used to find the edges that connects the corners of the gates. Finally, a VIO approach is used to determine the quadcopters pose relative to the gate.

The main benefit of using deep learning approaches is that there is less handcrafting involved compared to more traditional computer vision methods. The training process will determine the values of the kernels, which essentially specifies the characteristic features of an object that must be searched for. The downside however, which generally holds for any neural network, is that CNN architectures are often so complex that it acts as a black box, making debugging a process of trial and error. Moreover, deep learning approaches are known to be computationally expensive, especially in training the network. But executing a CNN in real-time can be challenging for small quadcopters.

3.6. On-board State Estimation

In order to effectively control a quadcopter it is essential to have knowledge of its states. Therefore, state estimation is a fundamental subsystem in quadcopter control. Which states need to be estimated is generally determined by the states on which the controllers are dependent. Most often these include height, attitude and angular rate for the inner loop controllers. The outer loop controllers rely on position and velocity estimates.[80]

A number of the practical experiments performed in the work described in section 3.4 used an external motion capture system to optically track the position and attitude of quadcopters. The quadcopters are equipped with visual markers which the system is able to track in 3D space. Although this system often yields close-to-groundtruth results it is an expensive solution and unpractical in cluttered and outdoor environments. In practice it is desirable to have on-board state estimation, which can be challenge due to limited computational power and sensor accuracy of the on-board hardware.

3.6.1. Visual Approaches to State Estimation

As explained in subsection 3.5.1 perception methods such as PnP or deep learning approaches, can be used to calculate the pose of a camera with respect to the world. From the pose one can estimate the quadcopter's attitude and position. However, real-world information about the image frames must be known beforehand because the estimated pose is always relative to either an earlier frame or to a detected object such as a gate. In the former case the method will be sensitive to drift when no feedback can be applied. Which is why VIO methods use inertial measurements to account for drift.[53]

V-SLAM is an approach that seems very similar to VO but differs from it by being concerned with global consistency of the trajectory. That is, historical odometry estimates are stored to detect so-called loop-closure, which will be used to refine the traveled trajectory. VO on the other hand focuses on optimizing the estimated odometry based only on the last couple of frames. As the complete history of the trajectory is not relevant to control performance. [54]

3.6.2. Estimating Attitude

In determining the attitude of a quadcopter the IMU's gyroscope, accelerometer and magnetometer are used in most cases. The gyroscope is used to estimate the angular rate whereas the accelerometer can be used to find the initial orientation with respect to the direction of gravity. The latter method can only be used when the quadcopter is steady, or when a method is available to subtract the body accelerations from the accelerometer measurements. For instance, thrust may always be aligned with the body z-axis and so will the body accelerations in undisturbed flight. This leaves 2 accelerometer axes available to estimate the attitude w.r.t. the gravity vector.

However, in reality blade-flapping may occur when the quadcopter is moving. This is a phenomenon caused by the fact that the propeller blades experience different airspeeds when comparing the advancing blade with a retreating blade w.r.t the airspeed. This will result in a difference of the produced lift along the rotor path. The collective lift force will tilt the rotor slightly as if it were acting 90 degrees ahead from the location of the maximum lift due to gyroscopic precession. This will cause a slight tilt in each rotor as well as for the collective thrust as illustrated in Figure 3.8. Which implies that the collective thrust no longer aligns with the body z-axis, making it less trivial to estimate the correct attitude using accelerometers only. [80, 81]

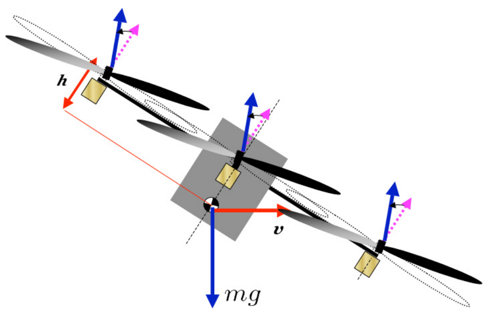


Figure 3.8: The effect of blade flapping on thrust (adapted from Huang et al. [81]).

Mahony, Kumar, and Corke [80] explain that the low-frequency component of a 3-axis accelerometer can be used to estimate the attitude in slow, steady flights. However, in order to improve the estimation for faster dynamics multiple the accelerometer measurements can be fused with the gyroscope and magnetometer by means of complementary filter or a Kalman filter. A complementary filter approximates the attitude by making a weighted combination of the fast and slow dynamics. E.g., in the simplest form it can look like :

$$\hat{R}_{i+1} = R_{i,\text{acc}} \cdot \alpha + (1-\alpha)(\Omega_{i,\text{gyro}} \cdot dt + \hat{R}_i) \tag{3.2}$$

Where $\tilde{R}$ is the approximation of attitude R, R_{acc} is the angle as derived from the accelerometer measurements and Ω_{gyro} is the angular rate as measured by the gyroscope. α determines to what proportion the next estimate should rely on the accelerometers or on the integrated gyroscope measurements. This term needs to be carefully tuned in order to give reliable results under different circumstances.
A more sophisticated complementary filter has been developed by Mahony et al. [82] that combines the gyroscope, magnetometer and accelerometer and other attitude estimation sources. Each separate estimation comes with its own corresponding gain that give the related cut-off frequency so that the filter can be carefully tuned to only select the appropriate frequency signals of each source.

3.6.3. Filtering

As aforementioned, attitude estimations can be calculated by combining fast dynamics from accelerometers with the slower dynamics of magnetometers. This is a basic example of sensor fusing or filtering and is not only applied in attitude estimation but often also necessary for reliable position and velocity estimation. The idea is to combine measurements from different sources to make use of their individual advantages (being fast or accurate) to mitigate the individual disadvantages (being slow or inaccurate) to improve the overall estimation of the state.

Typically, complementary filters do not include statistical properties describing the noise in a signal and are designed by frequency analysis mostly [83]. Kalman filters on the other hand are filters that estimate a state by taking a weighted average of measurements and model-based predictions. The weight (Kalman gain), depends on the statistical certainty of the measurements, which is measured by the variance of the prediction errors [84]. The basic Kalman filter is intended for linear systems and can be made optimal with respect to different cost functions, but extensions have been developed to work with nonlinear systems as well, such as the extended and unscented Kalman filters.

The EKF linearizes the nonlinear prediction model about nominal values of the states and inputs, but this method may give unsatisfactory results when a system is highly nonlinear. This is mainly caused because the variance values are propagated based on the linearized model that may not adequately represent the local linearity of the system. One attempt to mitigate this issue is by improving the linearization. This is the principle on which the IEKF is based. This approach adds an iterative loop in the EKF algorithm to repeatedly re-linearize the model about new nominal states, which were estimated based on the linearization of the previous loop [85].
Julier and Uhlmann [86] have developed the unscented Kalman filter to work with highly nonlinear systems. This filter makes a nonlinear approximation of the covariance values at several sample points around the mean. This allows for a more robust tuning of the Kalman gain. Crassidis and Markley [87] use this method in a attitude estimation pipeline for spacecraft which was shown in simulation to have a better performance than an EKF.

3.6.4. Estimating Position

Thanks to advances in inertial sensors it is possible to obtain relatively cheap and small systems that are able to estimate position good accuracy and stability. Most position estimation methods rely on a (slow) absolute position estimation method such as GPS and/or VIO, and use (fast) dead-reckoning integration of estimated velocity values and accelerometer measurements to estimate the position in between the GPS/VIO updates.
However, the quality of measurements are highly dependable on what maneuvers are performed. For instance, accelerometers in IMUs do not only measure linear accelerations a body experiences, but also includes the influence from gravity and centrifugal accelerations (Coriolis effect) that need to be corrected for. [88]

As for height, altitude can often be estimated well enough in outdoor flights using a barometric altimeter and GPS. Indoors however, there GPS does not work to provide absolute height determinations. Barometric measurements can still be used for relative altitude estimations, but may be sensitive to changing atmospheric conditions. Therefore, one can complement this system with downward oriented distance sensors such as radar, lidar or sonar. Which are known to have good accuracy and relatively high sampling rates. Naturally, distance measurements will be influenced by objects on the floor and when the floor is non-planar.[89]

3.6.5. Estimating Velocity

To estimate velocity one would expect it be done by differentiating the position estimates. However, for the desired maneuvers of ADR the velocity estimations must be fast and accurate. Meaning that for positional differentiating to work well enough the position estimates must be even more accurate and fast. As explained in subsection 3.6.4, absolute position measurements are slow and estimations rely on velocity values as well. In other words, there is a mutual dependency. Mahony, Kumar, and Corke [80] demonstrate a simple way to calculate horizontal velocity from accelerometer measurements. For this approach they assume that the quadcopter is steady in altitude and use a linear drag model that includes blade flapping. Moreover, a recursive low-pass filter is applied to the estimation to account for noise in the acceleration measurements.

3.7. Chapter Summary

ADR is considered to be a exciting platform to demonstrate the latest developments on quadcopter control, automated trajectory generation, perception and state estimation. IROS and IMAV are prestigious events where research groups are given the opportunity to present their work in a form of competitions that test their approaches on control, object detection and planning. In 2019 the first edition of the Alphapilot challenge has been held. This event was brought to live to decrease the gap between human piloted drone racing and ADR by letting the participating teams challenge one of the best human pilots. This chapter presented the most crucial subsystem and methodologies to enable a quadcopter to operate fully autonomously.

section 3.3 explained the different approaches to automatic trajectory generation. Which started with methods that optimized trajectories for smooth, minimum-snap trajectories, and others that aimed at efficiently planning around obstacles or finding time-optimal solutions. In addition to the methods that focused on mathematically solving the OCPs work has also been performed on machine learning approaches. These methods train ANNs with either real flight data or simulation data to give optimal control inputs.

section 3.4 focused on controlling the quadcopters to track these trajectories. It was shown that most control pipelines relied on feedforward terms, which could be derived form reference trajectories by leveraging differential flatness, as well as on feedback terms to minimize tracking errors. Improved tracking methods were developed by taking account more complex aerodynamic effects in calculating feedforward terms as well as implementing more robust inner loop controllers such as INDI.

In section 3.5 the application of perception systems were elaborated on. Two main goals of perception in ADR are VO and object detection. VO is used to derive the quadcopter's motion and pose from images captured by the onboard camera.
The most clear purpose of object detection is to visually detect gates so that the flight planner can calculate a trajectory that steers the quadcopter through the gate. It was shown that classical computer vision methods could be used to detect gates, but that it requires some manual handcrafting and the gates to be very distinguishable from their surroundings, both in color and shape.
The use of deep learning methods has shown to be very robust. CNNs have been shown to be very effective in classifying objects and have over the years become very popular methods for gate detection.

Finally, in section 3.6 the process of state estimation was described. It was shown how VO and PnP are robust, but expensive solutions to estimate position and attitude. Moreover, an explanation was given on how different on-board sensors could be combined to improve state estimation by using filtering techniques such as the complementary filter or EKF.

A

Pontryagin's Conditions for Optimality

In this appendix the derivation Pontryagin's Minimum Principle will be used to derive conditions for optimality. This derivation is adapted from Kirk [5]. Starting from the cost function:

$$J(\mathbf{u}) = h\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt \tag{A.1}$$

It is assumed that there are no bounds on the admissible states and control inputs. For the derivation where the control regions are actually bounded the reader is encouraged to follow the derivation in Kirk [5]. Where it is found that only the third condition for optimality is different from the unbounded case in this appendix.

Assuming that $h(\mathbf{x}(t_f), t_f)$ is differentiable it can be written as:

$$h\left(\mathbf{x}\left(t_f\right), t_f\right) = \int_{t_0}^{t_f} \frac{d}{dt}[h(\mathbf{x}(t), t)]dt + h\left(\mathbf{x}\left(t_0\right), t_0\right) \tag{A.2}$$

Since $\mathbf{x}(t_0)$ and t_0 are constant $h(\mathbf{x}(t_0), t_0)$ can be omitted from the minimization and the cost function can be written as:

$$\begin{gathered} J(\mathbf{u}) = \int_{t_0}^{t_f} \left\{ g(\mathbf{x}(t), \mathbf{u}(t), t) + \frac{d}{dt}[h(\mathbf{x}(t), t)] \right\} dt = \\ \int_{t_0}^{t_1} \left\{ g(\mathbf{x}(t), \mathbf{u}(t), t) + \left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t), t)\right]^T \dot{\mathbf{x}}(t) + \frac{\partial h}{\partial t}(\mathbf{x}(t), t) \right\} dt \end{gathered} \tag{A.3}$$

Next the differential equation constraints are added by introducing Lagrange multipliers $\mathbf{p}(t)$ and changing the cost function into the augmented form:

$$J_a(\mathbf{u}) = \int_{t_0}^{t_f} \left\{ g(\mathbf{x}(t), \mathbf{u}(t), t) + \left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t), t)\right]^T \dot{\mathbf{x}}(t) + \frac{\partial h}{\partial t}(\mathbf{x}(t), t) + \mathbf{p}^T(t)[\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}(t)] \right\} dt \tag{A.4}$$

For convenience the term within the integral is shortened to $g_a(\mathbf{x}(t), \dot{\mathbf{x}}, \mathbf{u}(t), \mathbf{p}(t), t)$. So that:

$$J_a(\mathbf{u}) = \int_{t_0}^{t_f} g_a(\mathbf{x}(t), \dot{\mathbf{x}}, \mathbf{u}(t), \mathbf{p}(t), t)dt \tag{A.5}$$

Using the principles of the calculus of variations and assuming that the end points at t_f can both be specified or free, the variation of $J_a(\mathbf{u}^*)$ can be found:

$$
\begin{aligned}
\delta J_a(\mathbf{u}^*) = 0 = & \left[\frac{\partial g_a}{\partial \dot{\mathbf{x}}}\left(\mathbf{x}^*(t_f), \dot{\mathbf{x}}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f\right)\right]^T \delta \mathbf{x}_f \\
& + \left[g_a\left(\mathbf{x}^*(t_f), \dot{\mathbf{x}}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f\right) - \left[\frac{\partial g_a}{\partial \dot{\mathbf{x}}}\left(\mathbf{x}^*(t_f), \dot{\mathbf{x}}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f\right)\right]^T \dot{\mathbf{x}}^*(t_f)\right] \delta t_f \\
& + \int_{t_0}^{t_f} \left\{ \left[\left[\frac{\partial g_a}{\partial \mathbf{x}}\left(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right)\right]^T \right.\right. \\
& \left. - \frac{d}{dt}\left[\frac{\partial g_a}{\partial \dot{\mathbf{x}}}\left(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right)\right]^T \right] \delta \mathbf{x}(t) \\
& + \left[\frac{\partial g_a}{\partial \mathbf{u}}\left(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right)\right]^T \delta \mathbf{u}(t) \\
& \left. + \left[\frac{\partial g_a}{\partial \mathbf{p}}\left(\mathbf{x}^*(t), \dot{\mathbf{x}}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right)\right]^T \delta \mathbf{p}(t) \right\} dt
\end{aligned}
\tag{A.6}
$$

Next, write out the terms within the integral that contains $h(\mathbf{x}(t), t)$ and write out the partial derivatives:

$$
\begin{aligned}
& \frac{\partial}{\partial \mathbf{x}}\left[\left[\frac{\partial h}{\partial \mathbf{x}}(\mathrm{x}^*(t), t)\right]^T \dot{\mathbf{x}}^*(t) + \frac{\partial h}{\partial t}(\mathbf{x}^*(t), t)\right] - \frac{d}{dt}\left\{\frac{\partial}{\partial \dot{\mathbf{x}}}\left[\left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t), t)\right]^T \dot{\mathbf{x}}^*(t)\right]\right\} = \\
& \left[\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}^*(t), t)\right] \dot{\mathbf{x}}^*(t) + \left[\frac{\partial^2 h}{\partial t \partial \mathbf{x}}(\mathbf{x}^*(t), t)\right] - \frac{d}{dt}\left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t), t)\right] = \\
& \left[\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}^*(t), t)\right] \dot{\mathbf{x}}^*(t) + \left[\frac{\partial^2 h}{\partial t \partial \mathbf{x}}(\mathbf{x}^*(t), t)\right] - \left[\frac{\partial^2 h}{\partial \mathbf{x}^2}(\mathbf{x}*(t), t)\right] \dot{\mathbf{x}}^*(t) - \left[\frac{\partial^2 h}{\partial \mathbf{x} \partial t}(\mathbf{x}^*(t), t)\right] = 0
\end{aligned}
\tag{A.7}
$$

So assuming that the order differentiation can be interchanged, the terms that involve $h(\mathbf{x}(t), t)$ vanish from the integral. The integral then becomes:

$$
\begin{aligned}
& \int_{t_0}^{t_f} \left\{ \left[\left[\tfrac{\partial g}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right]^T + \mathbf{p}^{*T}(t)\left[\tfrac{\partial \mathbf{a}}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right] \right.\right. \\
& \quad \left. - \tfrac{d}{dt}\left[-\mathbf{p}^{*T}(t)\right] \right] \delta \mathbf{x}(t) + \left[\left[\tfrac{\partial g}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right]^T \right. \\
& \quad \left.\left. + \mathbf{p}^{*T}(t)\left[\tfrac{\partial \mathbf{a}}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right]\right] \delta \mathbf{u}(t) + \left[\left[\mathbf{a}(\mathbf{x}^*(t), \mathbf{u}^*(t), t) - \dot{\mathbf{x}}^*(t)\right]^T\right] \delta \mathbf{p}(t) \right\} dt
\end{aligned}
\tag{A.8}
$$

This term must become zero on extremals. Note that the coefficient of $\delta \mathbf{p}(t)$ is always zero since it represents the constraint $\dot{\mathbf{x}}^*(t) = \mathbf{a}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)$ and therefore can take any arbitrary values. So they can be selected to make the coefficient of $\delta \mathbf{x}(t)$ zero:

$$
\dot{\mathbf{p}}^*(t) = -\left[\frac{\partial \mathbf{a}}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right]^T \mathbf{p}^*(t) - \frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)
\tag{A.9}
$$

Since $\delta \mathbf{u}(t)$ is independent its coefficient must be zero:

$$
\mathbf{0} = \frac{\partial g}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t) + \left[\frac{\partial \mathbf{a}}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), t)\right]^T \mathbf{p}^*(t)
\tag{A.10}
$$

In order to satisfy Equation A.6 the remaining terms outside of the integral must be zero:

$$
\begin{aligned}
& \left[\frac{\partial h}{\partial \mathbf{x}}\left(\mathbf{x}^*(t_f), t_f\right) - \mathbf{p}^*(t_f)\right]^T \delta \mathbf{x}_f + \left[g\left(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), t_f\right) + \frac{\partial h}{\partial t}\left(\mathbf{x}^*(t_f), t_f\right)\right. \\
& \qquad \left. + \mathbf{p}^{*T}(t_f)\left[\mathbf{a}\left(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), t_f\right)\right]\right] \delta t_f = 0
\end{aligned}
\tag{A.11}
$$

Equations eqs. (A.9) and (A.10) together with the constraint $\dot{\mathbf{x}}^*(t) = \mathbf{a}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), t\right)$ form the necessary conditions for optimality. It is convenient to use the Hamiltonian notation $\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)]$ so the necessary conditions can be written as:

$$\left.\begin{aligned} \dot{\mathbf{x}}^*(t) &= \frac{\partial \mathcal{H}}{\partial \mathbf{p}}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right) \\ \dot{\mathbf{p}}^*(t) &= -\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right) \\ \mathbf{0} &= \frac{\partial \mathcal{H}}{\partial \mathbf{u}}\left(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t\right) \end{aligned}\right\} \begin{array}{l}\text{for all} \\ t \in \left[t_0, t_f\right]\end{array}$$

$$\begin{aligned} &\left[\frac{\partial h}{\partial \mathbf{x}}\left(\mathbf{x}^*\left(t_f\right), t_f\right) - \mathbf{p}^*\left(t_f\right)\right]^T \delta \mathbf{x}_f + \left[\mathcal{H}\left(\mathbf{x}^*\left(t_f\right), \mathbf{u}^*\left(t_f\right), \mathbf{p}^*\left(t_f\right), t_f\right)\right. \\ &\qquad \left. + \frac{\partial h}{\partial t}\left(\mathbf{x}^*\left(t_f\right), t_f\right)\right] \delta t_f = 0 \end{aligned} \tag{A.12}$$

Part III

Preliminary Work

4
Lightweight Optimal Control

This chapter will elaborate on the work that has been performed in finding control approaches that are computationally efficient and simultaneously approach optimal-time solutions.

4.1. Bang-Bang Optimal Control

The approach that has been investigated is a lightweight, model predictive bang-bang controller. As explained in subsection 3.3.4, bang-bang type control is a control approach in which the control switches between saturated input values. This approach was considered promising because the OCP is reduced to one where the switching times are the only variables to be solved. Which is in benefit to the desired low computational effort.

Take for example a very simple temperature regulator of a room. This controller would instruct the central heating system to either be idle, heat or cool the room at maximum power. It is obvious that this is the fastest method of reaching the target temperature from an offset. However, one can imagine that overshoot is likely to happen if one does not switch between the input values in time. And that can be difficult to have the temperature settle on the target exactly since the control input does not scale proportionally with the temperature offset.

It is known that when the control input is constrained, the time-optimal control sequence consists of bang-bang control in thrust and bang-singular-bang in rotational control, as explained in subsection 3.3.4. Singular control is a specific type of control in sections of the trajectory where Pontryagin's Minimum Principle cannot be easily solved. Other means have to be used to find the solution for that part of the trajectory. Hehn, Ritz, and D'Andrea [45] do this by adding an equality constraint on one specific co-state which allows the control input to be solved explicitly.

In the following it will be demonstrated using PMP that for a simple 2D dynamic model with constrained inputs the time-optimal solution is of the bang-singular-bang type.

Take the quadcopter illustrated in Figure 4.1. The system is described as:

$$\begin{aligned}
\mathbf{x} &= \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta \end{bmatrix}^T \\
\mathbf{u} &= \begin{bmatrix} u_{\dot{\theta}} & u_T \end{bmatrix}^T \qquad\qquad \begin{bmatrix} \underline{u_{\dot{\theta}}} & \underline{u_T} \end{bmatrix}^T \leq \mathbf{u} \leq \begin{bmatrix} \overline{u_{\dot{\theta}}} & \overline{u_T} \end{bmatrix}^T \\
f(\mathbf{x},\mathbf{u}) = \dot{\mathbf{x}} &= \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{m}\left(u_T \sin\theta - C_d \cdot \dot{x}\right) \\ \dot{z} \\ \frac{1}{m}\left(W - u_T \cos\theta - C_d \cdot \dot{z}\right) \\ u_{\dot{\theta}} \end{bmatrix}
\end{aligned} \tag{4.1}$$

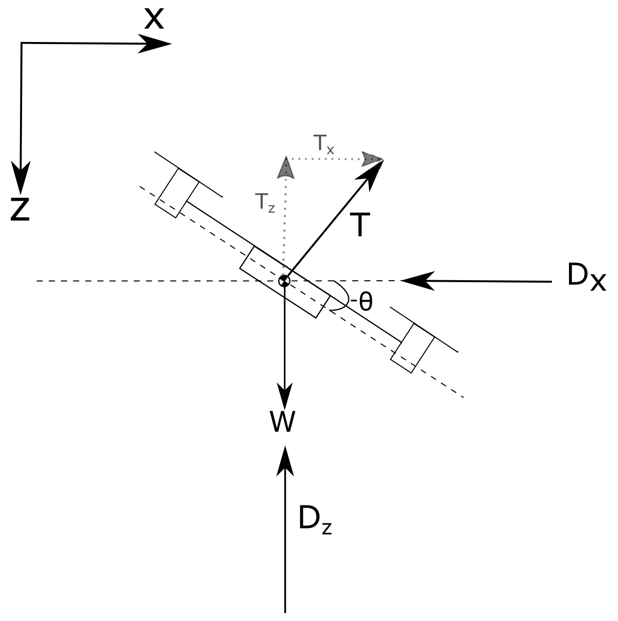


Figure 4.1: 2-D Quadcopter model
T =Thrust
D_x =Drag = $C_d \cdot \dot{x}$
$W = mg$ =Weight
m =Mass
g =Gravity
θ =Attitude angle
x =Position

Since a time-optimal trajectory is desired the cost function is simply:

$$J = \int_{t_0}^{t_f} 1 dt \tag{4.2}$$

Compare this to Equations 2.14 and 2.15 and it becomes clear that the Hamiltonian of this system is:

$$\mathcal{H} = 1 + \mathbf{p}^T f(\mathbf{x}, \mathbf{u}) = 1 + p_1 \dot{x} + p_2 \frac{1}{m} (u_T \sin\theta - C_d \dot{x}) + p_3 \dot{z} + \frac{p_4}{m} (u_T \cos\theta - C_d \dot{z} + W) + p_5 u_{\dot{\theta}} \tag{4.3}$$

PMP states that the optimal control input minimizes the Hamiltonian. So for Equation 4.3 it can be seen that $\mathcal{H}$ can be minimized by $u_{\dot{\theta}}$ and u_T separately.
Starting with the optimal pitch rate, the optimal control input $u^*_{\dot{\theta}}$ can be found minimizing $p_5 u_{\dot{\theta}}$.

$$u^*_{\dot{\theta}} = \underset{u_{\dot{\theta}} \in [\underline{u_{\dot{\theta}}}, \overline{u_{\dot{\theta}}}]}{\operatorname{argmin}} \{p_5 u_{\dot{\theta}}\} \tag{4.4}$$

From this becomes clear that $u^*_{\dot{\theta}}$ either becomes $\underline{u_{\dot{\theta}}}$ or $\overline{u_{\dot{\theta}}}$ if p_5 is larger or smaller than zero, respectively. p_5 becomes the regulating switching function for $u_{\dot{\theta}}$. If p_5 is zero for a nontrivial time this condition cannot hold and $u^*_{\dot{\theta}}$ becomes a singular arc. The singular solution for $u_{\dot{\theta}}$ can be derived from the condition that the switching function p_5 must remain zero by stating that $\dot{p}_5 = 0$.
From the second PMP condition in Equation 2.16 $\dot{p}_5$ can be calculated as:

$$\dot{p}_5 = -\frac{\partial \mathcal{H}}{\partial \theta} = -(p_2 u_T \cos\theta + p_4 u_T \sin\theta) \frac{1}{m} \tag{4.5}$$

θ^*_{sing} can be found by assuming that p_5 is zero during this period. And therefore $\dot{p}_5$ is equal to zero. By subsequently differentiating the resulting expression with respect to time optimal control input $u^*_{\dot{\theta},\text{sing}}$

can be found for the singular arc:

$$\begin{aligned} \theta^*_{\text{sing}} &= \arctan \frac{-p_2}{p_4} \\ u^*_{\theta,\text{sing}} &= \frac{\partial \theta^*_{\text{sing}}}{\partial t} = \frac{\frac{\partial}{\partial t}\left(\frac{-p_2}{p_4}\right)}{1+\left(\frac{-p_2}{p_4}\right)^2} \end{aligned} \tag{4.6}$$

Hehn, Ritz, and D'Andrea [45] demonstrate how to derive expressions for $\mathbf{p}$ so that they can be expressed in terms of constants and time. However, they have left out aerodynamic drag in their model which made it easier to find solutions for p_2 and p_4.

Similarly, it can be shown that the optimal control input for thrust, u_T^* has a bang-bang shape:

$$u_T^* = \underset{u_T \in \left[\underline{u_T}, \overline{u_T}\right]}{\operatorname{argmin}} \frac{p_2}{m} u_T \sin\theta^* + \frac{p_4}{m} u_T \cos\theta^* \tag{4.7}$$

From which the switching function $\Phi_T = \frac{p_2}{m}\sin\theta^* + \frac{p_4}{m}\cos\theta^*$ can be defined. Again, it can be seen that u_T^* is either $\underline{u_T}$ or $\overline{u_T}$ when $\Phi_T > 0$ or $\Phi_T < 0$, respectively.

4.2. Bang-Bang Controller

An optimal bang-bang control problem is in essence different from other OCPs because the constrained control values are known beforehand, one only needs to calculate when to switch between the saturated values.
Therefore, the suggested controller consists of a model predictive controller that leverages the simplicity of the bang-bang optimization principle.

The controller is decoupled for two dimensions: Bang-Bang control input is applied to the 'saturation dimension'. That is, pitch angle when moving in the body X-axis and roll angle when moving in the body y-axis. The controller will optimize the switching time for this direction. The reason that the roll and pitch angles have been selected as control inputs instead of rotational rate or torque is that this model allows to solve for position analytically, as will be shown in subsection 4.2.1. It can be shown that bang-bang control will still be the optimal solution for this by modifying Equation 4.3 by replacing θ by input u_θ and omitting the terms related to the rotational dynamics and z. Equation 4.7 then becomes:

$$\left[u_T^*, u_\theta^*\right] = \underset{\substack{u_T \in \left[\underline{u_T}, \overline{u_T}\right] \\ u_\theta \in \left[\underline{u_\theta}, \overline{u_\theta}\right]}}{\operatorname{argmin}} \frac{p_2}{m} u_T \sin u_\theta^* \tag{4.8}$$

From which it becomes clear that u_θ^* must either be $\underline{u_\theta}$ or $\overline{u_\theta}$ depending on the values of p_2 and u_T. The solution would be singular if $p_2 u_T = 0$.

The second dimension is the direction which is lateral to the saturation dimension and in this direction the angle is optimized instead of the switching time. This choice was made because in ADR it is often not necessary to minimize the lateral position error as quickly as possible. So more power can be directed towards the longitudinal control.

4.2.1. Prediction Model

As it is intended to be able to run the MPC on a tiny quadcopter emphasis is put on keeping the required computational power to a minimum to maximize the rate at which predictions can be made. For a start, this means that the required number of evaluations of the dynamic model will need to be limited.
Typical MPCs predicts ahead by numerically propagating the dynamics from its current state to a so-called planning-horizon. The number of evaluations then depend on the frequency of prediction, the

time-horizon and the discretization scheme.

On the other hand, the suggested method uses a 1D dynamic model that evaluates the position and velocity analytically in one direction. The model assumes perfect altitude control. That is, the thrust component that aligns with the Earth Z-axis is always equal to the weight: $T = \frac{W}{\cos\theta\cos\phi}$
Furthermore, it is assumed that the attitude angle is constant between switching instances and that the aerodynamic drag is linearly proportional to the speed. In Appendix B the derivation of the position and speed prediction is given. With as result:

$$x = c_1 e^{\frac{-C_d}{m}t} + c_2 + \frac{W\tan\phi}{C_d}t \tag{4.9a}$$

$$\dot{x} = \frac{-C_d}{m} c_1 e^{\frac{-C_d}{m}t} + \frac{W\tan\phi}{C_d} \tag{4.9b}$$

Where x is the position at time t, Cd is the drag coefficient, m is the mass, W is the weight $(m \cdot g)$ and ϕ is the attitude angle corresponding to the direction (pitch for forward prediction, roll for lateral prediction). Furthermore, constants c_1 and c_2 are calculated from the initial position and velocity:

$$c_1 = \left(\dot{x}_0 - \frac{W\tan\phi}{C_d}\right)\frac{m}{-C_d} \tag{4.10a}$$

$$c_2 = x_0 - c_1 \tag{4.10b}$$

See Figure 4.2. The optimal bang-bang motion from one position to a target position, starting and ending at rest, consists of an acceleration phase, a switching phase and a deceleration phase. When it is assumed that the switching phase is instantaneous a complete prediction can be made by splitting the trajectory in two parts. The first part is valid for the acceleration phase and the second part for the deceleration phase. The constants for the second phase are calculated from the states at the switching time from the first stage.

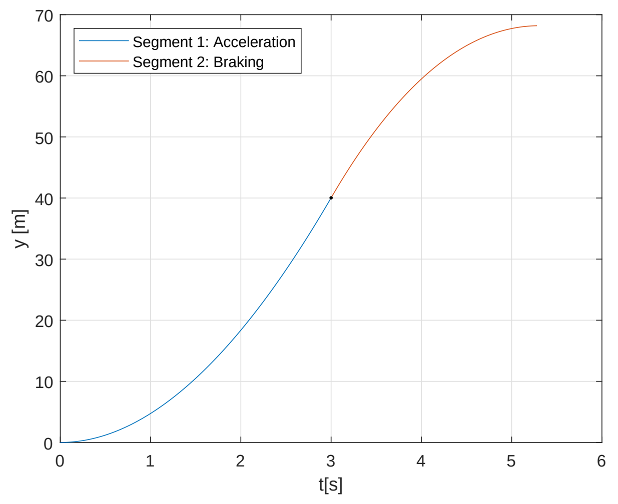


Figure 4.2: One-dimensional bang-bang maneuver trajectory.

Moreover, the values of θ and ϕ depend on which direction is being predicted. Since the x and y directions are being treated as having uncoupled dynamics in the prediction model. In the case x is the saturation dimension θ becomes the saturated value (even if the quadcopter has not reached this state yet in the simulation) and ϕ becomes the value of the roll angle in the simulated quadcopter's state. The opposite is true in case y is the saturation dimension.
This is done to correctly take the available thrust into account in case the quadcopter is excited in both roll and pitch.

4.2.2. Switch Time Solver

The optimal switching time is calculated iteratively adjusting the switch time from an initial guess. Due to the analytical nature of the dynamic model it is not necessary to discretize and propagate the prediction trajectory.
It can be seen in Equation 4.9 that the time cannot be solved for position x analytically. However, it can be solved for velocity $\dot{x}$. Therefore, the switching time is optimized as follows:
In each iteration the time at which the desired speed is reached during the deceleration phase is calculated explicitly from Equation 4.9b. That time is substituted in the position equation to find the position at which the desired speed is reached. Subsequently, the prediction time is adjusted in a bi-section algorithm based on whether this position is above or below the target position. In other words, the switching time is adjusted based on whether the prediction trajectory started braking too early or too late. The algorithm is stopped when the position error comes below a threshold value. Algorithm 1 summarizes this process.

Algorithm 1

```
$t_0 \leftarrow 0$
$t_1 \leftarrow$ initial guess
$E_t \leftarrow$ error threshold
$y_d \leftarrow$ desired position
while $E > E_T$ do
  $t_s \leftarrow \frac{t_0+t_1}{2}$
  $t_t \leftarrow$ get_time_from_desired_speed$(v_d)$
  $E \leftarrow$ get_position$(t_t) - y_d$
  if $E > 0$ then
    $t_1 \leftarrow t_s$
  else
    $t_0 \leftarrow t_s$
  end if
end while
```

Figure 4.3 illustrates typical results of this algorithm for a generic quadcopter. In this case the quadcopter starts from rest and is desired to come to rest 30 meters ahead. The predicted trajectory of each iteration is shown. It can be seen that the iterations converge to a switching time of 1.9 seconds, as indicated by the vertical dotted lines.

Per iteration only evaluations of the speed and position at the switching instant and at the target position are required.

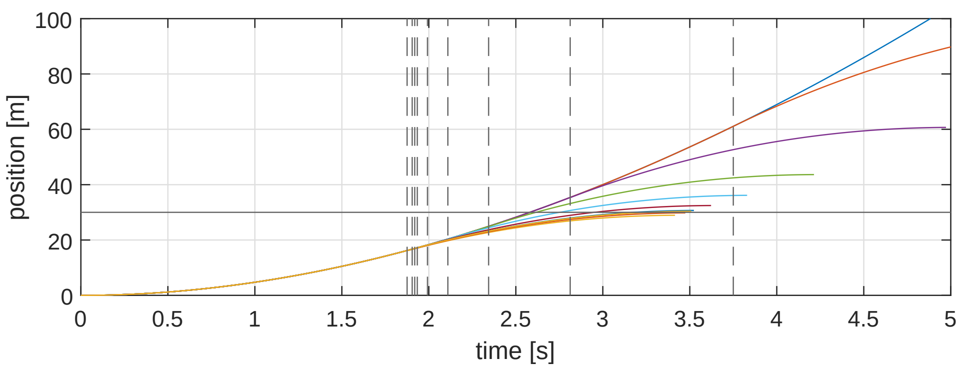


Figure 4.3: Bisection switching time optimization

Note that by using this method the switching time can only converge to a value that is larger than zero. It has been decided that the controller should switch between acceleration and braking when the switching time has converged close to zero. In section 4.5 attention is given on making this decision adaptive to compensate for the loss of time and deceleration caused by neglecting rotational dynamics in the prediction model.

4.2.3. Lateral Control

The lateral motion is controlled in a different way than the saturation direction. That is, the same prediction model is used, but no bang-bang motion is predicted. Instead, using the same bisection approach as for the switch time optimizer, the attitude angle is iteratively adjusted instead of the switch time. The goal is to reach the target position exactly at the time of arrival that is obtained from the prediction of the saturation direction. In this way only a bare minimum of the available is thrust is used to correct the lateral position error (which is smaller than the saturation direction position error) reserving more power to correct the position error in the saturation direction.

4.2.4. Braking

Because the prediction model will not match the real-life trajectory perfectly there is a chance that the algorithm will instruct the quadcopter to brake too early, and make it unable to reach the target. When no measures are taken the MPC will respond with fast switching bang-bang maneuvers to correct this. This will render the quadcopter highly unsteady, especially when close to the target where small inaccuracies will result in saturated responses nonetheless.

In order to avoid this an optimizer has been implemented that, similarly to the later controller in subsection 4.2.3, will optimize the attitude angle during braking. The result will be that in the case the quadcopter started braking too early it will continue to brake at a smaller angle. On the other hand, if the quadcopter started braking too late, it is impossible to brake harder because it is assumed that bang-bang accelerates and decelerates with a maximum force. The quadcopter will likely overshoot its target.

4.3. Matlab Simulations

To test the suggested controller different simulations have been set up. First, a relative simple simulator has been developed in Matlab. The dynamical model is a simple Newtonian model in which the aerodynamic drag is calculated similarly as in the prediction model. The altitude is controlled by a PID controller with a feedforward term and a PID controller controls the angular rate and the angular acceleration is approximated with a time delay: $\ddot{\theta} = \frac{1}{\tau}\left(\dot{\theta}_{\text{desired}} - \dot{\theta}\right)$.
The bang-bang controller has been implemented in the simulator and allows to simulate flight from waypoint to waypoint. Subsection subsection 4.3.1 contains the initial observations that were discovered while testing the principles of the controller.

4.3.1. Simulation Observations

To test the principals of the model-predictive bang-bang controller a number of simple flight plans have been implemented in the simulator. The plans consists of straight flights and skewed flights at different desired velocities and saturation angles. The simulations demonstrated how well the controller functions work for control in the saturated direction, lateral direction or a combination, while varying control parameters such as desired speeds. Moreover, it will become obvious to which extend the prediction model matches the 'true' model of the quadcopter.

Straight Flight:
A comparison for straight flight can be seen in Figure 4.4. In this flight the quadcopter starts from rest and should fly straight 15 meters forward and pass that waypoint with a certain desired velocity. The plots show the desired and actual pitch values and velocity for four separate flights. The differences between the flight are that the desired velocity at the target is either 0 or 10 m/s. Moreover, to demonstrate the influence of neglecting the rotational dynamics in the prediction model, flights have been simulated with a high and low gain attitude controller.

The simulation stops when the target position has been reached. So the final velocity in the plot is desired to match the desired velocities as close as possible.

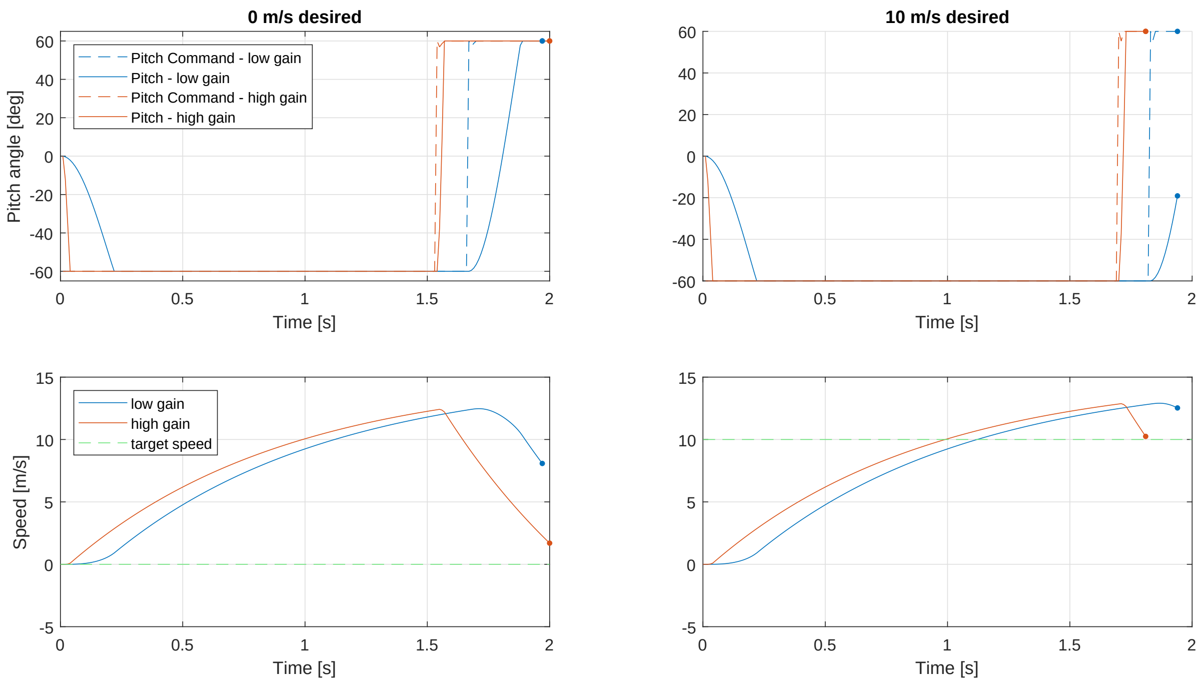


Figure 4.4: Straight Trajectory - Comparison between low- and high-gain inner loop

It can be seen that there is a significant difference between the low gain and high gain results. Although not perfect, the high gain trajectory matches its desired velocity at the target much closer than the low gain variant. This is caused by the fact that the actual deceleration of the quadcopter does not match the predicted deceleration because the pitch angle does not transfer to the desired pitch angle fast enough when starting to brake. In the low gain + 10 m/s case the pitch angle has barely changed 30% before the quadcopter has already passed the waypoint.

Moreover, it can be seen for both the low and high gain cases that when the desired velocity is high the velocity error is smaller than when it is low. This is because at higher velocity the predicted deceleration will match the actual deceleration more closely due to the fact that the aerodynamic drag has a larger contribution to the total deceleration and the simulation model and prediction model use the same aerodynamic model. The aerodynamic model only depends on velocity and not on attitude. So there is no mismatch caused by neglecting the rotational dynamics in the prediction.
This will likely not be the case in higher fidelity simulation models because then the aerodynamic drag is also dependent on the surface area that is projected normal to the velocity direction. Which means that in that case the drag is dependent on the attitude.

Skewed Flight
In Figure 4.5 the resulting trajectory can be found if the quadcopter is ordered to fly to a waypoint which lies 10 meters ahead and 5 meters to the right from its starting position. The desired heading remains fixed to the positive X direction.

The desired velocity has been set to 0 in the X-direction. The quadcopter passes the target waypoint with a distance of 42 cm, but the velocity does not reach the desired target velocity of 0. From the roll commands it can be seen that the lateral control is stable during the acceleration phase, but fluctuates when the braking phase starts.

From this figure it can also be seen that assuming decoupled translational dynamics for the path predictor has only a small effect on the prediction accuracy. This can be concluded from the fact that the predicted roll angle is more or less constant during the acceleration phase. Which implies that the lateral dynamics of the quadcopter matches the predicted lateral trajectory.

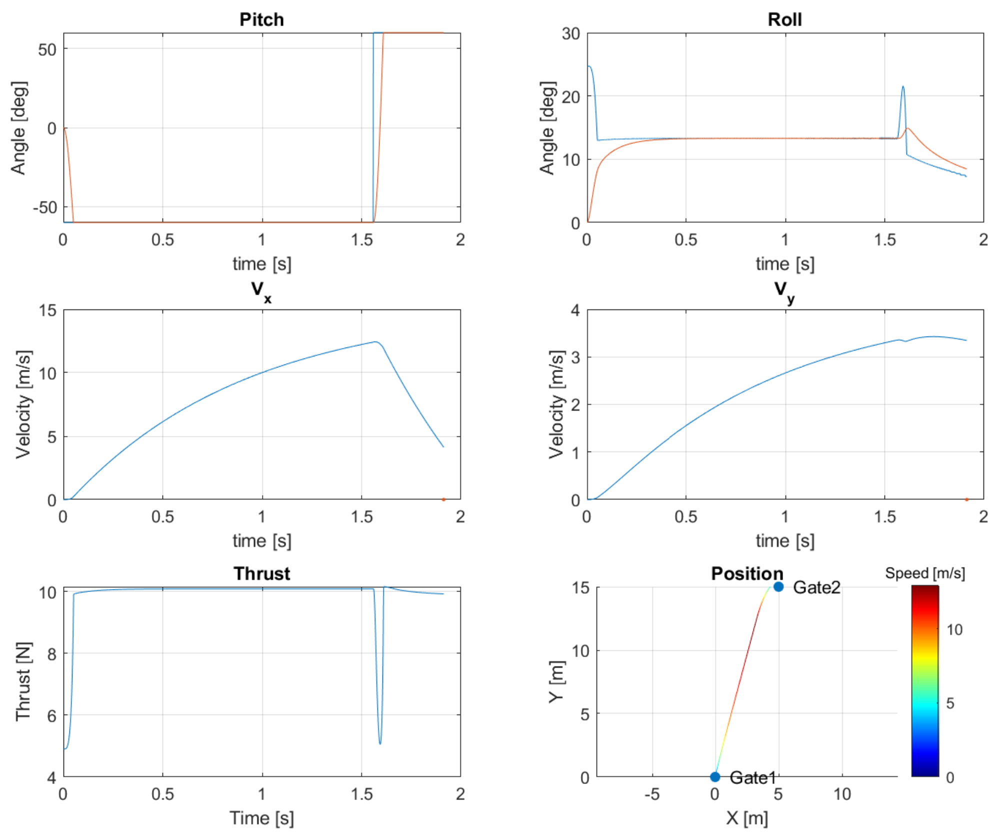


Figure 4.5: Resulting states and trajectory of a skewed maneuver

Prediction Stability

An effective measure to indicate how well the quadcopters follows the predicted trajectory is the estimated time-of-arrival. Which is calculated from Equation 4.9b. If the quadcopter would follow the prediction exactly, the predicted time-of-arrival would be constant. In Figure 4.6 the calculated time-of-arrival for the skewed flight can be seen. From this it becomes clear that the time of arrival increases when the pitch angle changes at the start and at the switching instant. The smaller fluctuations during the stable part are caused by the simulation's time-step-size, which governs the stopping rule of the bisection optimization.

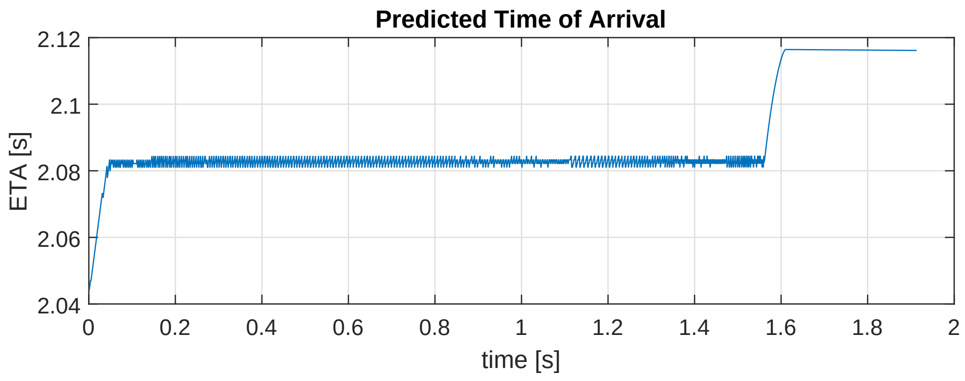


Figure 4.6: Estimated time of arrival through out the flight

4.4. Trajectory Comparisons

To evaluate the feasibility of the suggested controller time-optimal control comparisons have been made with approximated time-optimal trajectories. To this end the ICLOCS toolbox for MATLAB was used [90]. This toolbox uses direct collocation to solve OCPs. The same dynamic model as in section 4.3 was used to simulate four different types of trajectories. In subsection 4.3.1 no rotational inertia has been implemented in the dynamical model and no limits on the rotational rates were applied, consequently the optimal control solution will yield infinitely high rotational rates. Therefore the upper and lower limits of the pitch and roll rates were set to $\pm 360°$.

Additionally, trajectories that are optimized for minimum snap were calculated as well. Note that the minimum-snap methods described in subsection 3.3.1 were derived from different dynamical models. So those approaches could not be implemented directly for a fair comparison. Instead minimum snap is approximated in ICLOCS by numerically adding snap to the cost function. In ICLOCS it was not possible to incorporate snap continuity between multiple waypoints so it can be assumed that the resulting trajectories do not approach the theoretical minimum-snap optimality.
Finally, a high-gain PID controller was simulated too. This simple controller uses one set of gains which is used for all maneuvers. Both the PID and Bang-Bang base their desired heading on the position error such that the quadcopter is always pointing towards the waypoint.
An overview of the relevant simulation settings can be found in Table 4.1. The mass and drag coefficient are based on the Parrot Bebop quadcopter.

Setting	*Value*
Mass	0.452[kg]
Max $\frac{T}{W}$	3 [-]
Max roll and pitch rate	360 [°/s]
Max yaw rate	180 [°/s]
C_D	0.57

Table 4.1: Simulation Settings

4.4.1. Straight Flight

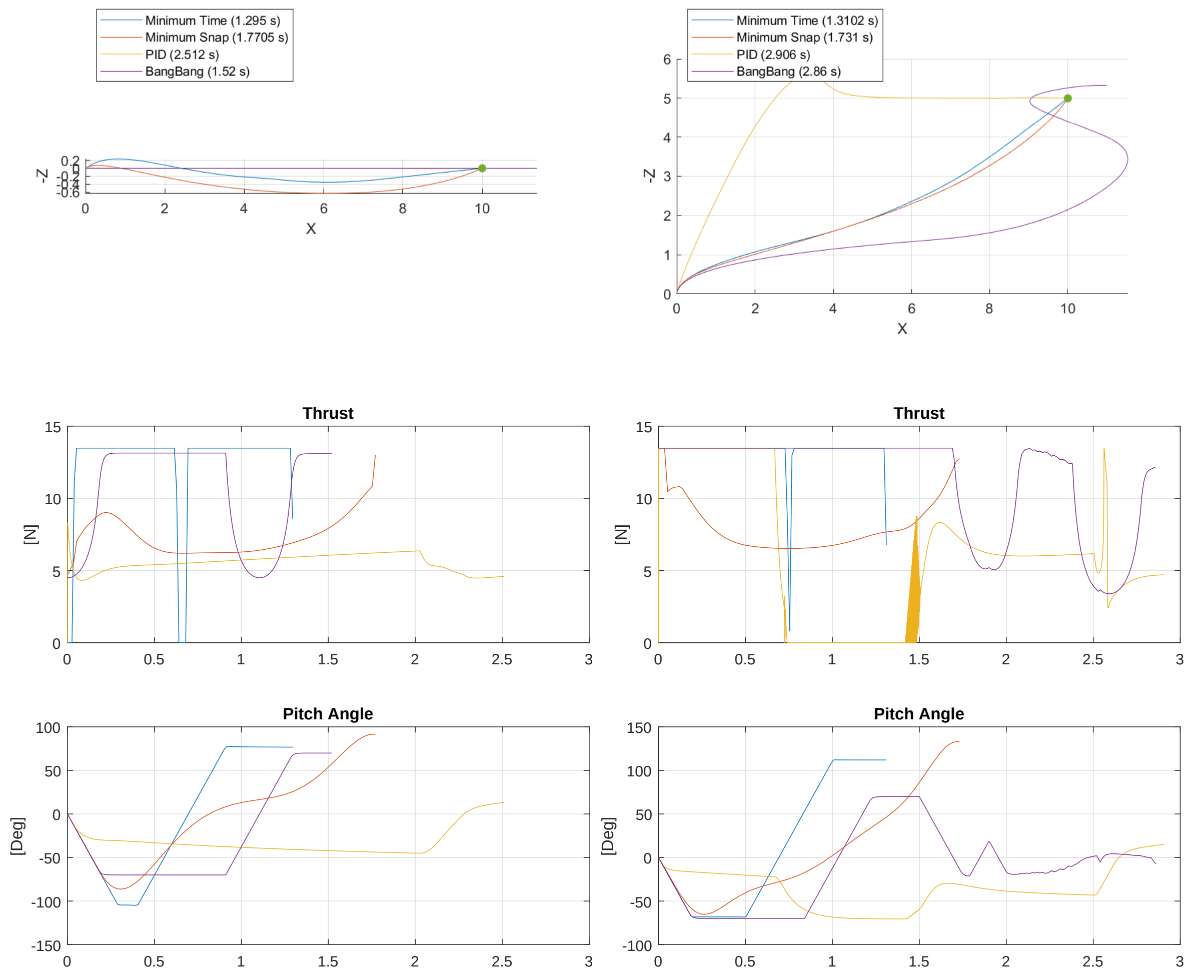
The first set of maneuvers consist of longitudinal flights only. The first maneuver consists of only one waypoint that lies 10 meters in front. Figure 4.7a shows the resulting trajectories, thrust inputs and resulting pitch angles.

It can be seen that both the bang-bang and minimum-time give bang-zero-bang inputs to the pitch rate. The minimum-time controller also gives a bang-zero-bang input to the thrust whereas the bang-bang and PID controller scales the thrust with the pitch angle for perfect altitude control.
Although, the bang-bang controller reaches the waypoint the second-fastest, it shows some overshoot.

In the second test the waypoint is now located at an altitude of 5 meters. Since the prediction model of the bang-bang controller assumes constant altitude for predicting the thrust it is expected that the performance will deteriorate in this setting.
Figure 4.7b shows the resulting trajectories. Because of the high saturation angles the bang-bang controller struggles to reach the desired altitude. The optimal-time solution decreases the pitch angle about a second earlier to trade in forward velocity for more vertical velocity.
Interestingly, despite that the minimum-snap and minimum-time trajectories look very similar, the profiles of the thrust and pitch angle look very different.

4.4.2. Longitudinal + Lateral Flight

The final set tests are calculated to test to what extend decoupling the longitudinal and lateral predictions in the bang-bang controller affect the performance with respect to the other controllers. Here the quadcopter is simulated to fly through two waypoints: the first 10 meters ahead and the second 5 meters to the right of the first one.



(a) Constant Altitude

(b) Variable Altitude

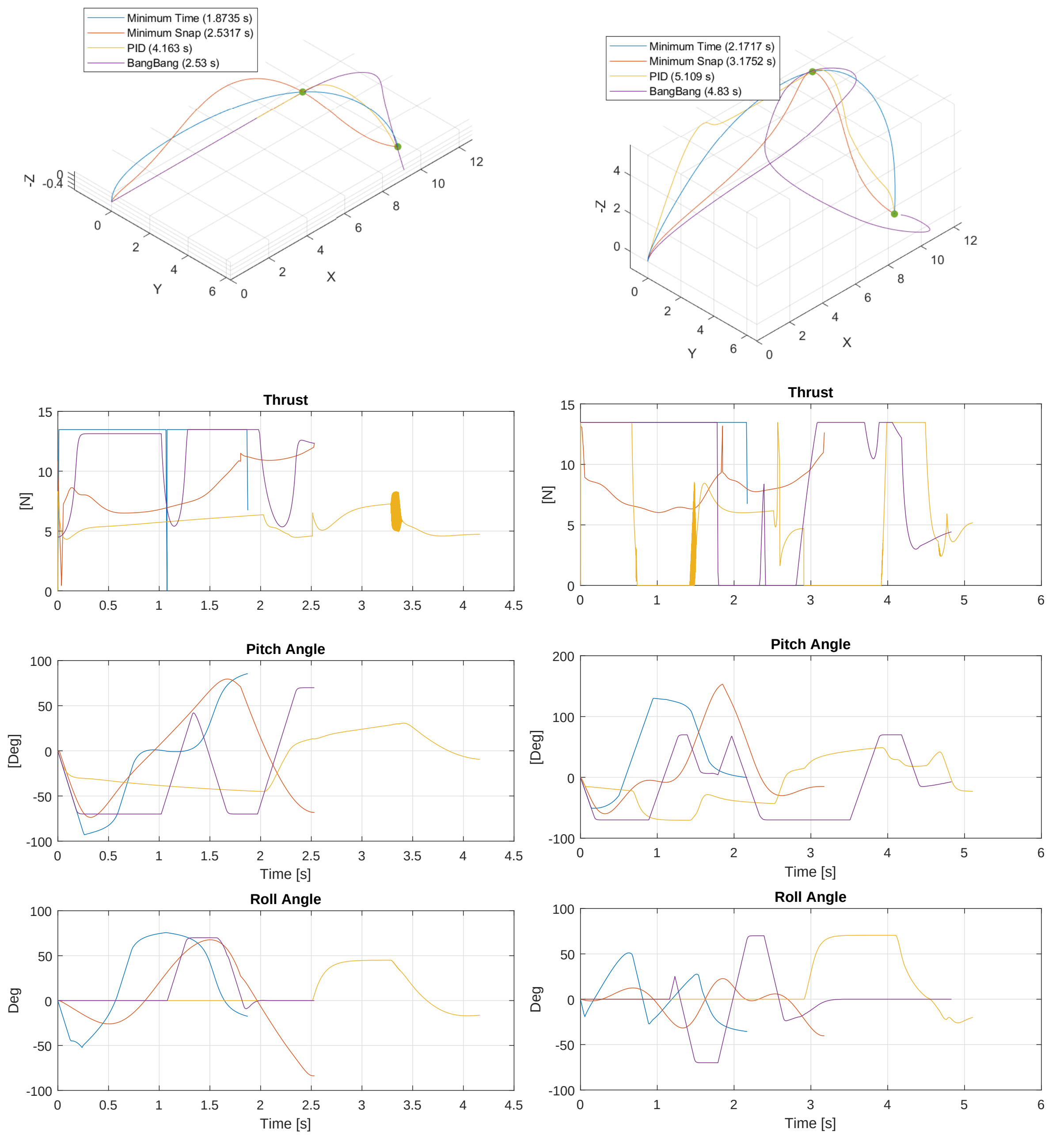
Figure 4.7: Longitudinal trajectory comparisons

Just as in the longitudinal test it is desired for the quadcopter to be at rest at the final waypoint. However, the bang-bang controller requires a desired velocity for the first waypoint as well in order to optimize for a switching time. This value has been chosen to be equal to the velocity at the first waypoint from the optimal-time solution. It is expected that this velocity will be too high because the optimal-time solution optimizes for the complete trajectory and will likely pass through the first waypoint at an angle that points closer to the final waypoint. The bang-bang controller on the other hand only predicts in straight lines.

The first comparison can be found in Figure 4.8a. The minimum-time trajectory takes the form of a continuous turn, first rolling to the left to better line up for the final waypoint. As expected, the bang-bang controller overshoots the first waypoint by a lot before turning and flying to the final waypoint.

Finally, for the final trajectories the first waypoint is set 5 meters higher in altitude. Which will result in a similar trajectory to Figure 4.8b for the PID and Bang-Bang controller while flying towards the first waypoint. However, while flying to the final waypoint the altitude controller will give less thrust to lower the altitude. The optimal-time solution on the other hand will keep the thrust saturated and use more extreme pitch and roll angles to control the altitude such that all power is available to keep accelerating or decelerating, which was also demonstrated by Hehn, Ritz, and D'Andrea [91].

Figure 4.8b shows the resulting trajectories. It can be seen that the time-optimal solution indeed keeps



(a) Constant altitude

(b) Variable Altitude

Figure 4.8: Longitudinal + lateral trajectory comparisons

the thrust saturated throughout the entire flight. Its pitch angle going as high as $130°$. Since the PID and Bang-Bang only takes one waypoint in account at a time the first part of the trajectory looks similar to Figure 4.8b. During the last phase the Bang-Bang controller still suffers from the overshoot of the first phase and cannot accelerate towards the final waypoint while the thrust is zero during the descent.

4.5. Transition Compensation

From the results in Figure 4.3.1 and section 4.4 it could be seen that the largest prediction inaccuracies are caused by neglecting the rotational dynamics in the prediction model, resulting in large positional overshoots. In this section steps are being taken in developing an adaptive method to compensate for these inaccuracies and improve trajectory prediction.

The prediction trajectory can be improved if it is known how much time, velocity and speed are being lost during the transition from accelerating to braking. Since these dynamics are not included in the prediction model. By compensating for these losses to the second section (the braking section) of the predicted trajectory a more accurate path can be calculated.

However, these losses are not known beforehand and are likely to vary with the state of the quadcopter at the switching instant. Therefore, the intention is to create a dataset that summarizes how much time, position and speed are lost during the rotational motion. This done by measuring how much time it takes to switch from one extreme saturation angle to the other, and what the position and speed difference is at the end of the rotation. Subsequently, a model is derived to estimate what compensation values should be added to improve the prediction to the next waypoint. The end result is that the switching time will converge to a better value when the compensation measures are in place. Moreover, by continuously measuring the remaining losses during flight the model can be improved on-line.

Figure 4.9 demonstrates what a compensation case could look like. In this example it is assumed that during the rotation 1 m/s in velocity, 2 m in position and 0.3s in time are lost. When comparing to the uncompensated trajectory it can be seen that as result the quadcopter is instructed to brake at 1.4 seconds instead of at 1.5 seconds.

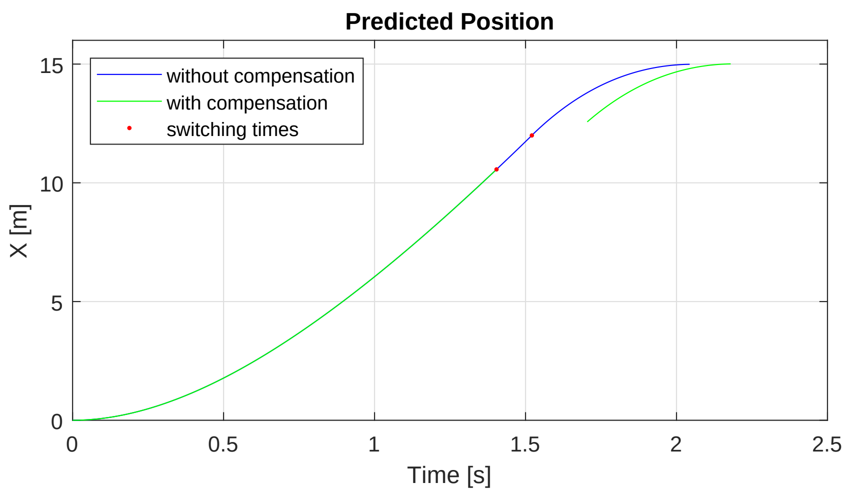


Figure 4.9: Predicted position with and without transition compensation applied.

One issue that arises is that, in order to avoid unreliable results, the optimization should be turned off during the transition from accelerating to braking. Otherwise the controller will immediately start optimizing the angle, as described in subsection 4.2.4. This optimization is based on the predicted braking trajectory that does not take into account the transition, which will yield inaccurate results.
This implies that the quadcopter is practically flying blind during this period which can be hazardous in real-life scenarios.

In Figure 4.10 the results of three different simulations can be found. In each case the quadcopter is controlled to fly straight to a target 15 meters ahead and try to reach the target at zero speed. In the first scenario no compensation is added. In the second scenario a compensation of -0.128 seconds is added. Which was the result of measuring the rotation losses in the no compensation scenario. The compensation periods are indicated by the dashed vertical lines.
The final scenario is one with -0.6 seconds compensations which much more than the measured compensation.

It can be seen from the final velocity values that in uncompensated case the quadcopter start braking too late and is unable to reach the zero velocity.
Even in the compensated cases zero velocity is never reached exactly at the target position. But it comes close. The overcompensated case brakes too early and even starts accelerating again after the

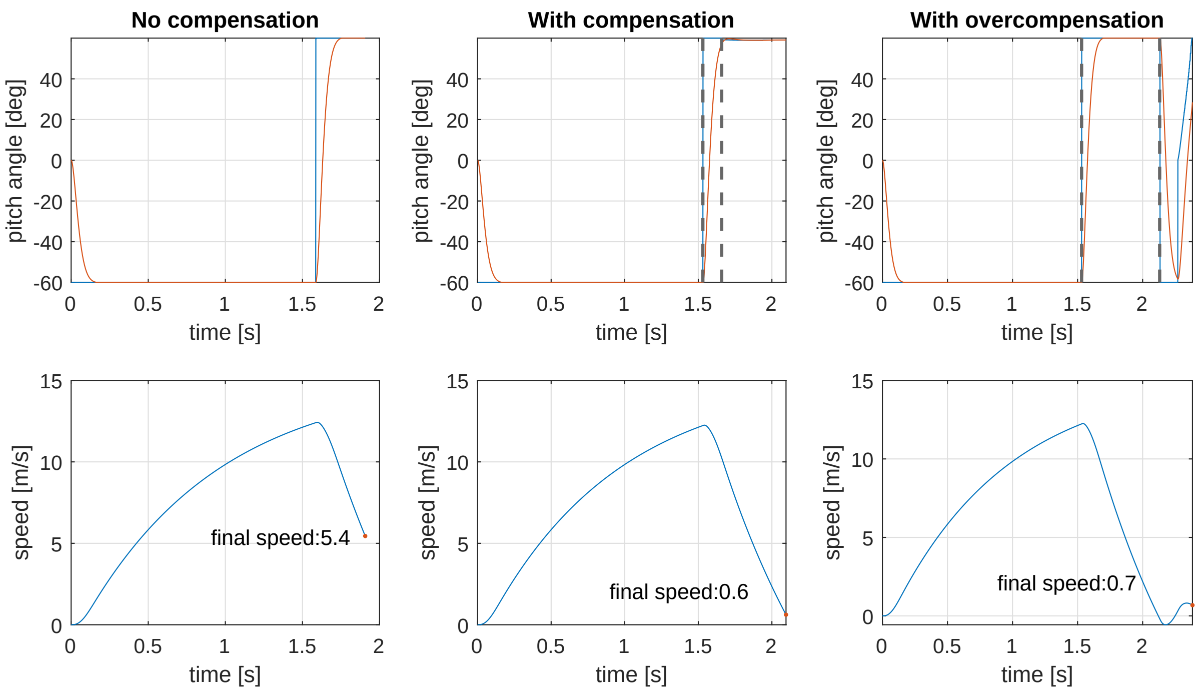


Figure 4.10: Resulting pitch angle and speed for different levels of transition compensation.

transition. The final velocity is close to the correctly compensated case but the target is reached at a later time.

4.6. Discussion of the Preliminary Results

In section 4.1 a bang-bang controller has been suggested as a solution to approaching time-optimal control while maintaining low computational effort to solve for optimal control inputs. This approach is based on the fact that the time-optimal solution to an unconstrained OCP has either a bang-bang or bang-singular-bang shaped control input. Which can be proven using PMP as shown in section 4.1.
An important consequence of this result is that the OCP reduces to a problem in which only the switching times will need to be solved. This is already beneficial to reducing the computational effort of the solving method.

Subsequently, a path predictor was created to calculate the position and velocity trajectories the quadcopter would fly towards its target. A key feature of this predictor is that the position and its derivatives can be evaluated analytically. No state propagation is required. Which again is in favor of computational efficiency.

4.6.1. The Effects of Simplifications

However, some simplification had to be introduced in order for the differential equations to be solved analytically. First, a simple, two-dimensional quadcopter was used to approximate all forces acting on the quadcopter. In this model it was assumed that drag is only linearly proportional with the body's velocity. Furthermore it is assumed that the thrust and attitude angle are known and constant. This resulted in assuming that the thrust would always counteract the drones weight to maintain a constant altitude.
Furthermore, because of the constant angle assumption it was impossible to use the predictor to calculate the dynamics at the switching instances. Therefore these dynamics where neglected and it was assumed that the transition from one saturated attitude angle to the other happens instantly.
Finally, because a two-dimensional dynamic model with constant altitude has been used, it is assumed that the translational dynamics are decoupled in the longitudinal, lateral direction, and vertical direction

which is similar to the work of Hehn and D'Andrea [40]. In fact, the path predictor does not take into account the vertical dynamics at all.

Decoupled Translational Dynamics

The results of section 4.3 show that in simulation the quadcopters flight deviates the most from the prediction during the transition phases. Which can be expected because the MATLAB simulation uses a dynamical model that can be considered as a 3-dimensional version of the model used by the path predictor with added rotational dynamics. Which is a flaw of the analysis. However, since the 3-dimensional model does not decouple the longitudinal and lateral dynamics it can still be used to conclude that the decoupled dynamics assumption of the path predictor does only have a small effect on the prediction quality.

Moreover, the path predictor derives the available thrust from the assumption that altitude is constant throughout the flight. Consequently, the comparisons performed in section 4.4 have shown that the bang-bang controller can only approach the 'true' time-optimal trajectory when the waypoints are all on the same altitude. In cases where the quadcopter has to change its height the path predictor is unaware of the vertical motion of the quadcopter and the actual available thrust. Simulations have shown that the trajectory quality degrades significantly compared to the trajectories of true time-optimal and even PID control.
Also in the constant altitude simulations it can be seen that the time-optimal solution is in fact a trajectory that is not constant in altitude, but one that keeps the thrust saturated and uses the attitude to control attitude angle and acceleration simultaneously. While the bang-bang controller only saturates thrust when the attitude angle is saturated.

Constant Angle Assumption

This shows also that the constant angle assumption will contribute to deviating from true time-optimality. In addition to the fact that in the time-optimal solution the attitude angle is not constant, there is also an issue of determining the value of the constant angle. When flying in a straight line this value be derived of the maximum thrust-over-weight ratio the quadcopter can provide. However, this means that all available thrust is used to accelerate in one direction and keep the vertical acceleration at zero. As a result there is no thrust left to compensate for tracking errors and to control the lateral direction as well. Therefore, in practice the constant angle has to be smaller in order to have some 'spare thrust' available. But it is unknown how much spare thrust is sufficient, and each Newton of unused thrust is a step away from time-optimality.

Instantaneous Transition Assumption

Furthermore, from Figure 4.6 it became clear that a large cause for prediction inaccuracy comes from the assumption of instantaneous change of attitude angles at the switching instances. In simulation rotational dynamics are included whereas the predictor cannot take those into account. During the transition there is less thrust available for acceleration or braking. Which is especially a problem when finding the optimal braking switch time. The optimal braking switch time the latest instance the quadcopter can brake and still reach its target with the desired speed. The predictor does not know that time is lost during the transition before the thrust is available for braking and will always predict optimal switching time too far into the future. Resulting in overshooting the target.
Fortunately, the method of predicting the transition dynamics which was presented in section 4.5 shows promise in mitigating this problem and will be further investigated in the main thesis phase.

4.6.2. Computational efficiency

It can be seen that some severe simplifications had to be made to minimize the required computational power of the solver. Which in fact has become significantly efficient. The solver uses a bisection scheme to iterate the switching time based on the path predictions. For the saturation direction only four evaluations will need to be made per iteration. That is, position and velocity at the switching time and at the time of arrival. For the secondary direction only the position needs to be evaluated at the time of arrival per iteration.

The total number of iterations required depends on the initial guesses and the set thresholds. Simulations experiments have shown that the switch time optimizer requires on average 9 iterations for the switching time to converge to a precision of $< 0.01s$. The angle optimizer for the secondary direction requires on average 12 iterations for it to converge to a value with a precision of $< 0.01°$. This comes down to a total of only $4 \cdot 9 + 1 \cdot 12 = 48$ path value evaluations for calculating the optimal switching time and secondary attitude angle. The MATLAB code (which also include a lot of unnecessary operations for debugging and analysis purposes) requires around 1ms for the entire optimization function to run on a consumer desktop computer (i7-3770k 4.2GHz quadcore CPU, 16GB RAM). Which is more than sufficient to run in real-time and be implemented as an MPC. This results shows great promise that the optimizer will be able to run fast enough onboard a quadcopter. Which will be further investigated in the main phase of the thesis.

Bibliography

[1] *What is Drone Racing?* 2020. URL: https://thedroneracingleague.com/ (visited on 12/03/2020).

[2] Lockheed Martin Corporation. *AlphaPilot — Lockheed Martin AI Drone Racing Innovation Challenge*. 2019. URL: https://www.lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html (visited on 01/30/2020).

[3] Richard Bellman. *Dynamic programming*. Dover Publications, 1957. ISBN: 9780486428093.

[4] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[5] Donald E Kirk. *Optimal control theory: An introduction*. Vol. 1. Dover Publications, 1998, pp. 86–90. ISBN: 9780486434841.

[6] RE Kalman. “The theory of optimal control and the calculus of variations”. In: *Mathematical optimization techniques*. University of California Press Los Angeles, CA, 1963, pp. 309–331.

[7] John T. Betts. “Survey of numerical methods for trajectory optimization”. In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207. ISSN: 07315090. DOI: 10.2514/2.4231.

[8] Francesco Biral, Enrico Bertolazzi, and Paolo Bosetti. “Notes on numerical methods for solving optimal control problems”. In: *IEEJ Journal of Industry Applications* 5.2 (2016), pp. 154–166. ISSN: 21871108. DOI: 10.1541/ieejjia.5.154.

[9] Marco Frego. “Numerical Methods for Optimal Control Problems with application to autonomous vehicles”. In: *PhD dissertation* (2014), pp. 1–84. URL: http://eprints-phd.biblio.unitn.it/1227/1/MFT.pdf.

[10] Anil V. Rao. “A survey of numerical methods for optimal control”. In: *Advances in the Astronautical Sciences*. 2010. ISBN: 9780877035572.

[11] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality: Second Edition*. 2011. ISBN: 9781118029176. DOI: 10.1002/9781118029176.

[12] Fei Yue Wang, Huaguang Zhang, and Derong Liu. “Adaptive dynamic programming: An introduction”. In: *IEEE Computational Intelligence Magazine* (2009). ISSN: 1556603X. DOI: 10.1109/MCI.2009.932261.

[13] Asma Al-Tamimi, Frank L. Lewis, and Murad Abu-Khalaf. “Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 38.4 (2008), pp. 943–949. ISSN: 10834419. DOI: 10.1109/TSMCB.2008.926614. URL: http://ieeexplore.ieee.org/document/4554208/.

[14] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2010. DOI: 10.1137/1.9780898718577.

[15] H. Kwakernaak and R. Sivan. “Linear Optimal Control Systems”. In: *IEEE Transactions on Automatic Control* (1974). ISSN: 15582523. DOI: 10.1109/TAC.1974.1100628.

[16] Philipp Foehn and Davide Scaramuzza. “Onboard State Dependent LQR for Agile Quadrotors”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2018), pp. 6566–6572. ISSN: 10504729. DOI: 10.1109/ICRA.2018.8460885.

[17] Arthur E. Bryson, Yu-Chi Ho, and George M. Siouris. “Applied Optimal Control: Optimization, Estimation, and Control”. In: *IEEE Transactions on Systems, Man, and Cybernetics* (2008). ISSN: 0018-9472. DOI: 10.1109/tsmc.1979.4310229.

[18] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality: Second Edition*. 2011. ISBN: 9781118029176. DOI: 10.1002/9781118029176.

[19] *IROS 2019 - Macau*. 2019. URL: https://www.iros2019.org/ (visited on 05/26/2020).

[20] Hyungpil Moon et al. “The IROS 2016 Competitions [Competitions]”. In: *IEEE Robotics and Automation Magazine* 24.1 (2017), pp. 20–29. ISSN: 10709932. DOI: 10.1109/MRA.2016.2646090.

[21] *International Micro Air Vehicles, Conferences and Competitions*. 2019. URL: http://www.imavs.org/ (visited on 11/02/2020).

[22] Mark W Mueller, Markus Hehn, and Raffaello D’Andrea. “A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification”. In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3480–3486. ISBN: 9781467363587. DOI: 10.1109/IROS.2013.6696852.

[23] Daniel Mellinger and Vijay Kumar. “Minimum snap trajectory generation and control for quadrotors”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525. ISBN: 9781612843865. DOI: 10.1109/ICRA.2011.5980409.

[24] Charles Richter, Adam Bry, and Nicholas Roy. “Polynomial trajectory planning for quadrotor flight”. In: *International Conference on Robotics and Automation* Isrr (2013), pp. 1–16. URL: http://www.michigancmes.org/papers/roy7.pdf.

[25] Michiel J. Van Nieuwstadt and Richard M Murray. “Real-time trajectory generation for differentially flat systems”. In: *International Journal of Robust and Nonlinear Control* 8.11 (1998), pp. 995–1020. ISSN: 10498923. DOI: 10.1002/(SICI)1099-1239(199809)8:11<995::AID-RNC373>3.0.CO;2-W.

[26] Charles Richter, Adam Bry, and Nicholas Roy. “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments”. In: *Springer Tracts in Advanced Robotics* 114.Isrr (2016), pp. 649–666. ISSN: 1610742X. DOI: 10.1007/978-3-319-28872-7_37.

[27] Mattia Landolfi et al. “Autonomous Guidance Navigation and Control for Agile Quadrotors Using Polynomial Trajectory Planning and L 1 Adaptive Control”. In: *2017 25th Mediterranean Conference on Control and Automation (MED)* 3 (2017), pp. 1041–1046. DOI: 10.1109/MED.2017.7984255.

[28] Francesca Baldini et al. “Fast Motion Planning for Agile Space Systems with Multiple Obstacles”. In: 2016, pp. 1–14. DOI: 10.2514/6.2016-5683.

[29] Steven M. LaValle. “Rapidly-Exploring Random Trees: A New Tool for Path Planning”. In: *In* (1998). ISSN: 1098-6596. DOI: 10.1.1.35.1853. arXiv: arXiv:1011.1669v3.

[30] Steven M. LaValle and James J. Kuffner. “Randomized kinodynamic planning”. In: *International Journal of Robotics Research* (2001). ISSN: 02783649. DOI: 10.1177/02783640122067453.

[31] *Bitcraze AB*. URL: https://bitcraze.io.

[32] Bryan Penin et al. “Vision-based minimum-time trajectory generation for a quadrotor UAV”. In: *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe (2017), pp. 6199–6206. ISSN: 21530866. DOI: 10.1109/IROS.2017.8206522.

[33] Steven G. Johnson. *The NLopt nonlinear-optimization package*. URL: http://github.com/stevengj/nlopt.

[34] Davide Falanga et al. “PAMPC: Perception-Aware Model Predictive Control for Quadrotors”. In: *IEEE International Conference on Intelligent Robots and Systems* (2018), pp. 5200–5207. ISSN: 21530866. DOI: 10.1109/IROS.2018.8593739.

[35] Gao Tang, Weidong Sun, and Kris Hauser. “Learning Trajectories for Real- Time Optimal Control of Quadrotors”. In: *IEEE International Conference on Intelligent Robots and Systems*. 2018, pp. 3620–3625. ISBN: 9781538680940. DOI: 10.1109/IROS.2018.8593536.

[36] Shuo Li et al. “Aggressive Online Control of a Quadrotor via Deep Network Representations of Optimality Principles”. In: (2019). arXiv: 1912.07067. URL: http://arxiv.org/abs/1912.07067.

[37] Elia Kaufmann et al. “Deep Drone Acrobatics”. In: *arXiv* (2020). ISSN: 23318422. DOI: 10.15607/rss.2020.xvi.040. arXiv: 2006.05768. URL: http://arxiv.org/abs/2006.05768.

[38] Efe Camci and Erdal Kayacan. “Learning motion primitives for planning swift maneuvers of quadrotor”. In: *Autonomous Robots* 43.7 (2019), pp. 1733–1745. ISSN: 15737527. DOI: 10.1007/s10514-019-09831-w. URL: https://doi.org/10.1007/s10514-019-09831-w.

[39] Antonio Loquercio et al. “Deep Drone Racing: From Simulation to Reality With Domain Randomization”. In: *IEEE Transactions on Robotics* (2019), pp. 1–14. ISSN: 1552-3098. DOI: 10.1109/tro.2019.2942989. arXiv: 1905.09727. URL: http://arxiv.org/abs/1905.09727.

[40] Markus Hehn and Raffaello D’Andrea. *Quadrocopter trajectory generation and control*. Vol. 44. 1 PART 1. IFAC, 2011, pp. 1485–1491. ISBN: 9783902661937. DOI: 10.3182/20110828-6-IT-1002.03178. URL: http://dx.doi.org/10.3182/20110828-6-IT-1002.03178.

[41] Richard F. Hartl, Suresh P. Sethi, and Raymond G. Vickson. “Survey of the maximum principles for optimal control problems with state constraints”. In: *SIAM Review* 37.2 (1995), pp. 181–218. ISSN: 00361445. DOI: 10.1137/1037043.

[42] H. Maurer and N. P. Osmolovskii. “Second order optimality conditions for bang-bang control problems”. In: *Control and Cybernetics* 32.3 SPEC. ISS. (2003), pp. 555–584. ISSN: 03248569.

[43] Stephen K. Lucas and C. Yalçin Kaya. “Switching-time computation for bang-bang control laws”. In: *Proceedings of the American Control Conference* 1 (2001), pp. 176–181. ISSN: 07431619. DOI: 10.1109/ACC.2001.945537.

[44] Zhaolong Shen and Sean B. Andersson. “Minimum time control of a second-order system”. In: *Proceedings of the IEEE Conference on Decision and Control* 2.1 (2010), pp. 4819–4824. ISSN: 01912216. DOI: 10.1109/CDC.2010.5717016.

[45] Markus Hehn, Robin Ritz, and Raffaello D’Andrea. “Performance benchmarking of quadrotor systems using time-optimal control”. In: *Autonomous Robots* 33.1-2 (2012), pp. 69–88. ISSN: 09295593. DOI: 10.1007/s10514-012-9282-3.

[46] Matthias Faessler, Davide Falanga, and Davide Scaramuzza. “Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 476–482. ISSN: 23773766. DOI: 10.1109/LRA.2016.2640362.

[47] Yuki Kawai and Kenji Uchiyama. “Design of frequency shaped LQR considering dynamic characteristics of the actuator”. In: *2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016* (2016), pp. 1235–1239. DOI: 10.1109/ICUAS.2016.7502630.

[48] Ban Wang, Khaled A. Ghamry, and Youmin Zhang. “Trajectory tracking and attitude control of an unmanned quadrotor helicopter considering actuator dynamics”. In: *Chinese Control Conference, CCC*. 2016. ISBN: 9789881563910. DOI: 10.1109/ChiCC.2016.7555068.

[49] Matthias Faessler et al. “Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2015. DOI: 10.1109/ICRA.2015.7139420.

[50] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. “Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories”. In: *IEEE Robotics and Automation Letters* 3.2 (Dec. 2017), pp. 620–626. ISSN: 23773766. DOI: 10.1109/LRA.2017.2776353. arXiv: 1712.02402. URL: http://arxiv.org/abs/1712.02402%20http://dx.doi.org/10.1109/LRA.2017.2776353.

[51] Ezra Tal and Sertac Karaman. “Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness”. In: *Proceedings of the IEEE Conference on Decision and Control* 2018-Decem (2019), pp. 4282–4288. ISSN: 07431546. DOI: 10.1109/CDC.2018.8619621. arXiv: 1809.04048.

[52] Ewoud J.J. Smeur, Qiping Chu, and Guido C.H.E. De Croon. “Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles”. In: *Journal of Guidance, Control, and Dynamics* 39.3 (2016), pp. 450–461. ISSN: 07315090. DOI: 10.2514/1.G001490. URL: http://resolver.tudelft.nl/uuid:31536cfa-89e1-4d44-873e-f2f398bd69ca.

[53] Giuseppe Loianno et al. “Estimation, Control, and Planning for Aggressive Flight with a Small Quadrotor with a Single Camera and IMU”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 404–411. ISSN: 23773766. DOI: 10.1109/LRA.2016.2633290.

[54] Friedrich Fraundorfer and Davide Scaramuzza. “Visual odometry: Part I: The First 30 Years and Fundamentals”. In: *IEEE Robotics and Automation Magazine* (2011). ISSN: 10709932.

[55] David G Low. “Distinctive image features from scale-invariant keypoints”. In: *International Journal of Computer Vision* (2004), pp. 91–110. URL: `https://www.cs.ubc.ca/%7B~%7Dlowe/papers/ijcv04.pdf`.

[56] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “LNCS 3951 - SURF: Speeded Up Robust Features”. In: *Computer Vision–ECCV 2006* (2006), pp. 404–417. URL: `http://link.springer.com/chapter/10.1007/11744023%7B%5C_%7D32`.

[57] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. “BRISK: Binary Robust invariant scalable keypoints”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2011. ISBN: 9781457711015. DOI: `10.1109/ICCV.2011.6126542`.

[58] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2564–2571. DOI: `10.1109/ICCV.2011.6126544`.

[59] Dibyendu Mukherjee, Q. M. Jonathan Wu, and Guanghui Wang. “A comparative experimental study of image feature detectors and descriptors”. In: *Machine Vision and Applications* 26.4 (2015), pp. 443–466. ISSN: 14321769. DOI: `10.1007/s00138-015-0679-9`.

[60] Martin A Fischler and Robert C Bolles. “Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. ISSN: 15577317. DOI: `10.1145/358669.358692`.

[61] Friedrich Fraundorfer and Davide Scaramuzza. “Visual odometry: Part II: Matching, robustness, optimization, and applications”. In: *IEEE Robotics and Automation Magazine* (2012). ISSN: 10709932. DOI: `10.1109/MRA.2012.2182810`.

[62] Zhengyou Zhang et al. “A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry”. In: *Artificial Intelligence* (1995). ISSN: 00043702. DOI: `10.1016/0004-3702(95)00022-4`.

[63] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. “A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2011. ISBN: 9781457703942. DOI: `10.1109/CVPR.2011.5995464`.

[64] Shuo Li et al. “Visual Model-predictive Localization for Computationally Efficient Autonomous Racing of a 72-gram Drone”. In: (2019). arXiv: `1905.10110`. URL: `http://arxiv.org/abs/1905.10110`.

[65] J. Illingworth and J. Kittler. “A survey of the hough transform”. In: *Computer Vision, Graphics and Image Processing* (1988). ISSN: 0734189X. DOI: `10.1016/S0734-189X(88)80033-1`.

[66] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986). ISSN: 01628828. DOI: `10.1109/TPAMI.1986.4767851`.

[67] Chris Harris and Mike Stephens. “A combined edge and corner detector”. In: *Proc 4th Alvey Vision Conference* (1988).

[68] Sunggoo Jung et al. “A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge”. In: *Journal of Field Robotics* 35.1 (2018), pp. 146–166. ISSN: 15564967. DOI: `10.1002/rob.21743`.

[69] Shuo Li et al. “Autonomous drone race: A computationally efficient vision-based navigation and control strategy”. In: 1 (2018), pp. 1–2. arXiv: `1809.05958`. URL: `http://arxiv.org/abs/1809.05958`.

[70] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* (1998). ISSN: 00189219. DOI: `10.1109/5.726791`.

[71] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 14764687. DOI: `10.1038/nature14539`.

[72] Matthew D. Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014. ISBN: 9783319105895. DOI: 10.1007/978-3-319-10590-1_53. arXiv: 1311.2901.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* (2017). ISSN: 15577317. DOI: 10.1145/3065386.

[74] Sunggoo Jung et al. “Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2539–2544. ISSN: 23773766. DOI: 10.1109/LRA.2018.2808368.

[75] Wei Liu et al. “SSD: Single shot multibox detector”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9905 LNCS (2016), pp. 21–37. ISSN: 16113349. DOI: 10.1007/978-3-319-46448-0_2. arXiv: arXiv:1512.02325v5.

[76] Elia Kaufmann et al. “Beauty and the beast: Optimal methods meet learning for drone racing”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2019-May. 2019, pp. 690–696. ISBN: 9781538660263. DOI: 10.1109/ICRA.2019.8793631. arXiv: 1810.06224.

[77] Antonio Loquercio et al. “DroNet: Learning to Fly by Driving”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1088–1095. ISSN: 23773766. DOI: 10.1109/LRA.2018.2795643.

[78] Philipp Foehn et al. “AlphaPilot: Autonomous Drone Racing”. In: (2020). arXiv: 2005.12813. URL: http://arxiv.org/abs/2005.12813.

[79] Zhe Cao et al. “Realtime multi-person 2D pose estimation using part affinity fields”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.143.

[80] Robert Mahony, Vijay Kumar, and Peter Corke. “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor”. In: *IEEE Robotics and Automation Magazine* (2012). ISSN: 10709932. DOI: 10.1109/MRA.2012.2206474.

[81] Haomiao Huang et al. “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2009), pp. 3277–3282. ISSN: 10504729. DOI: 10.1109/ROBOT.2009.5152561.

[82] Robert Mahony et al. “Nonlinear complementary filters on the special linear group”. In: *International Journal of Control* 85.10 (2012), pp. 1557–1573. ISSN: 00207179. DOI: 10.1080/00207179.2012.693951.

[83] Walter T. Higgins. “A Comparison of Complementary and Kalman Filtering”. In: *IEEE Transactions on Aerospace and Electronic Systems* AES-11.3 (1975), pp. 321–325. ISSN: 00189251. DOI: 10.1109/TAES.1975.308081.

[84] R. E. Kalman. “A new approach to linear filtering and prediction problems”. In: *Journal of Fluids Engineering, Transactions of the ASME* 82.1 (1960), pp. 35–45. ISSN: 1528901X. DOI: 10.1115/1.3662552.

[85] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. “Kalman filters for non-linear systems: A comparison of performance”. In: *International Journal of Control* 77.7 (2004), pp. 639–653. ISSN: 00207179. DOI: 10.1080/00207170410001704998.

[86] Simon J. Julier and Jeffrey K. Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI* 3068.July 1997 (1997), p. 182. ISSN: 0277786X. DOI: 10.1117/12.280797.

[87] John L. Crassidis and F. Landis Markley. “Unscented filtering for spacecraft attitude estimation”. In: *Journal of Guidance, Control, and Dynamics* (2003). ISSN: 07315090. DOI: 10.2514/2.5102.

[88] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. “Using inertial sensors for position and orientation estimation”. In: *Foundations and Trends in Signal Processing* 11.1-2 (2017), pp. 1–153. ISSN: 19328354. DOI: 10.1561/2000000094. arXiv: 1704.06053.

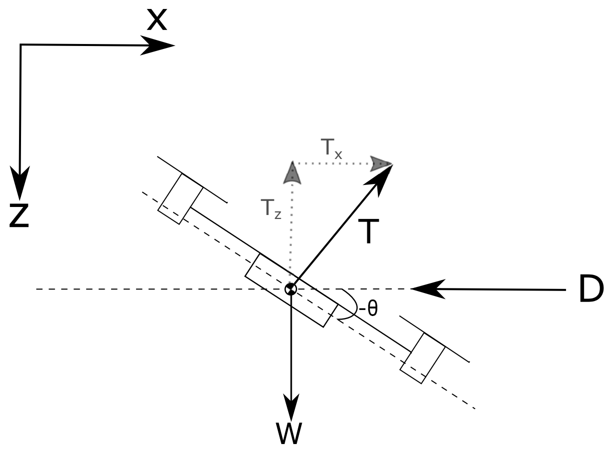
[89] Grzegorz Szafranski et al. “Altitude estimation for the UAV’s applications based on sensors fusion algorithm”. In: *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings* (2013), pp. 508–515. DOI: `10.1109/ICUAS.2013.6564727`.

[90] Yuanbo Nie, Omar Faqir, and Eric C. Kerrigan. “ICLOCS2: Try this Optimal Control Problem Solver before you Try the Rest”. In: *2018 UKACC 12th International Conference on Control, CONTROL 2018* 2.2017 (2018), p. 336. DOI: `10.1109/CONTROL.2018.8516795`.

[91] Markus Hehn, Robin Ritz, and Raffaello D’Andrea. “Performance benchmarking of quadrotor systems using time-optimal control”. In: *Autonomous Robots* 33.1-2 (2012), pp. 69–88. ISSN: 09295593. DOI: `10.1007/s10514-012-9282-3`.

B Bang-Bang Prediction Model

B.1. Decoupled Prediction Model



T = Thrust
D = Drag = $C_d \cdot V$
$W = mg$ =weight
m = mass
g = gravity
θ = attitude angle
x = position

Assume perfect constant altitude control:

$$\begin{aligned} T_z &= W \\ T &= \frac{W}{\cos\theta} \\ T_x &= T\sin\theta = W\frac{\sin\theta}{\cos\theta} = W\tan\theta \end{aligned} \tag{B.1}$$

Sum of forces:

$$\begin{aligned} m \cdot \ddot{x} &= T_x - D = W\tan\theta - C_d \cdot \dot{x} \\ \ddot{x} &= g\tan\theta - \frac{C_d}{m}\dot{x} \end{aligned} \tag{B.2}$$

Homogeneous equation:

$$\ddot{x} + \frac{Cd}{m}\dot{x} = 0 \tag{B.3}$$

Characteristic equation:

$$\begin{aligned}
r^2 + \frac{C_d}{m} r &= 0 \\
r\left(r + \frac{C_d}{m}\right) &= 0 \\
r_1 = 0, r_2 &= \frac{-C_d}{m} \\
x_h &= c_1 + c_2 e^{\frac{-C_d}{m} t}
\end{aligned} \tag{B.4}$$

Particular equation:
Assume $x_p = A \cdot t$, where A is a constant:

$$\begin{aligned}
x_p &= At \\
\dot{x_p} &= A \\
\ddot{x_p} &= 0 \\
0 &= g \tan\theta - \frac{C_d}{m} \cdot A \\
A &= \frac{W \tan\theta}{C_d} \\
x_p &= \frac{W \tan\theta}{C_d} t
\end{aligned} \tag{B.5}$$

$$\begin{aligned}
x &= x_h + x_p = c_1 e^{\frac{-C_d}{m} t} + c_2 + \frac{W \tan\theta}{C_d} t \\
\dot{x} &= c_1 \frac{-C_d}{m} e^{\frac{-C_d}{m} t} + \frac{W \tan\theta}{C_d}
\end{aligned} \tag{B.6}$$

Solve for initial conditions:

$$\begin{aligned}
\dot{x}_0 &= c_1 \frac{-C_d}{m} e^{\frac{-C_d}{m} t_0} + \frac{W \tan\theta}{C_d} \\
c_1 &= \left(\dot{x}_0 - \frac{W \tan\theta}{C_d}\right) \frac{m}{-C_d e^{\frac{-C_d}{m} t_0}} \\
x_0 &= c_1 e^{\frac{-C_d}{m} t_0} + c_2 + \frac{W \tan\theta}{C_d} t_0 \\
c_2 &= x_0 - c_1 e^{\frac{-C_d}{m} t_0} - \frac{W \tan\theta}{C_d} t_0 \\
&= x_0 - \left(\dot{x}_0 - \frac{W \tan\theta}{C_d}\right) \frac{m}{-C_d e^{\frac{-C_d}{m} t_0}} e^{\frac{-C_d}{m} t_0} - \frac{W \tan\theta}{C_d} t_0
\end{aligned} \tag{B.7}$$

In the code we always assume that t_0 = 0 which makes the constants:

$$\begin{aligned}
c_1 &= \left(\dot{x}_0 - \frac{W \tan\theta}{C_d}\right) \frac{m}{-C_d} \\
c_2 &= x_0 - \left(\dot{x}_0 - \frac{W \tan\theta}{C_d}\right) \frac{m}{-C_d} = x_0 - c_1
\end{aligned} \tag{B.8}$$

B.2. Combined Euler Angles

Note that in a 3-dimensional case the thrust components cannot always be represented as in Equation B.1. In fact, when the quadcopter has rotated both in pitch (θ) and roll (ϕ) the lateral thrust component in the velocity reference frame must be corrected by $\frac{1}{\cos\theta}$ which can be derived by rotating the thrust component in the body frame back to the velocity frame as shown in Equation B.9:

$$\begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{-W}{\cos\theta\cos\phi} \end{bmatrix} = \begin{bmatrix} -W\tan\theta \\ W\frac{\tan\phi}{\cos\theta} \\ W \end{bmatrix} \tag{B.9}$$

C

Flight Experiments Results

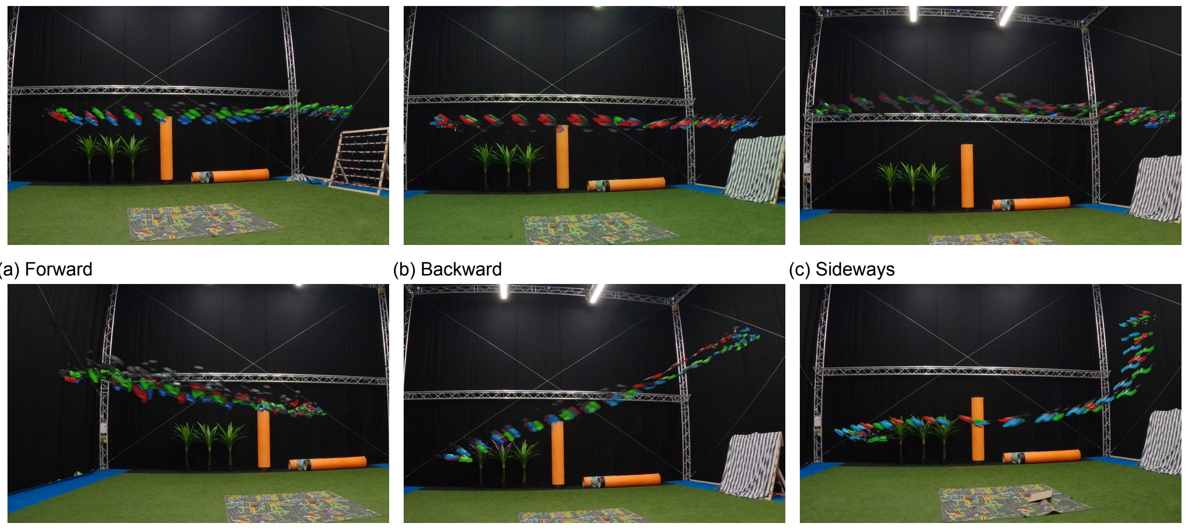
Flight experiments have been performed to test the Bang-Bang controller in practice. The experiments were performed indoors at the TU Delft's 'Cyberzoo'. The main results of the experiments are discussed in the paper in Part I. This chapter will summarize all performed experiments in plots, including the experiments not discussed in the paper.
Where applicable, comparisons between the bang-bang, bang-bang controller with transition compensation and a PID controller are made. section C.1 briefly describes the different sets of experiments and in section C.2 the relevant results are shown.

C.1. Flight Maneuvers

Several sets of maneuvers have been established to test the controller performance for different aspects.

Motion Primitives

This section describes the the experiments that test the controllers' performance for combinations of longitudinal and lateral flight. These consists of short flights, governed by two waypoints that are positioned either longitudinally, laterally or both longitudinally and laterally from each other.



(a) Forward (b) Backward (c) Sideways

(d) Forward-Sideways (e) Forward-Up (f) Forward-Down

Figure C.1: Comparison compositions of the motion primitive maneuvers. Each color represents a single run with a different controller: Red = PID, Blue = BangBang without compenstation, Green = BangBang with compensation

Figure C.1 shows a compilation of all motion primitive maneuvers that have been performed with the flights of different controllers. Because each flight should end in rest, the bang-bang controller flights switch to a PID controller after the first bang-bang motion has been completed. This is necessary to avoid endless oscillations around the target position.

Circular Flight

Additionally, to more closely approach a drone racing scenario, flights with multiple consecutive waypoints have been performed. That is, a set of rectangular positioned waypoints have been established for this maneuver. Since the bang-bang controller uses a decoupled model to predict paths in straight lines, it is unable to accommodate changes in heading. Therefore, most flight experiments have been performed with a fixed heading. Furthermore, the 'saturation' dimension is automatically varied between the longitudinal and lateral direction, based on the maximum component of the position error. That is, if the target position lies straight ahead the saturation dimension lies in the body's x-axis and therefore a bang-bang motion is planned in pitch. If the target lies to the left or to the right the saturation dimension lies to the body's y-axis and a bang-bang motion is planned for roll.

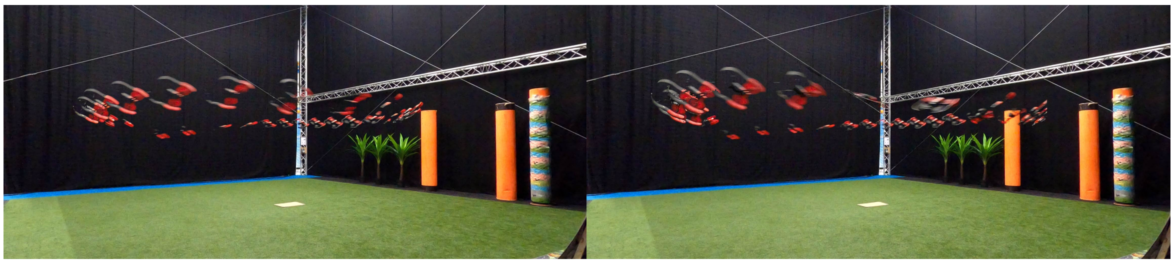

(a) Bang-Bang (b) PID

Figure C.2: Compositions of single circular runs for the bang-bang controller and PID controller.

Reduced Positional Feedback

The quadcopter's position is estimated by the Optitrack system. From which the velocity is derived on-board. For most experiments the Optitrack's sample rate was set to 120Hz. However, in drone racing the quadcopter often must estimate its position with expensive on-board solutions that are likely as accurate, nor as fast as Optitrack. To test the bang-bang controller for lower quality position and velocity estimations a few tests have been performed in which the positional reporting of Optitrack has been set to a lower frequency of 10Hz. The related filters still run as fast as the Bebop's main control loop (512Hz).
These tests have been performed for the Forward, Forward-Sideways and Circular maneuvers. And the results can be found in section C.2.
However, even though it can be seen that the controllers perform worse for this scenario. There haven't been performed enough flights to establish a statistically significant difference between the Bang-Bang controller and the PID controller.

Real Thrust Estimation

The final flight experiments concern the case in which the path prediction model is augmented with an estimated thrust force. In the nominal case this model assumes perfect hovering conditions and a constant thrust force in which the vertical component perfectly matches the quadcopter's weight in opposite direction. However, the real thrust force can be approximated with the thrust command that the altitude controller produces. This is done by first finding the thrust command value when the quadcopter is in hover condition (0.56 for the Bebop). It was then assumed that the produced thrust force scales linearly with the thrust command.
This experiment was only performed with the forward-up and forward-down maneuver.
No significant performance increases were expected because the path predictor assumes constant thrust during the entire bang-bang maneuver. Using real-time thrust estimations would therefore only be helpful if the vertical thrust component had a constant offset from the quadcopter's weight during the complete maneuver. Which is never the case.
The boxplots in Figure C.7 also show no significant differences between the bang-bang controller with and without real-time thrust estimations.

C.2. Plots

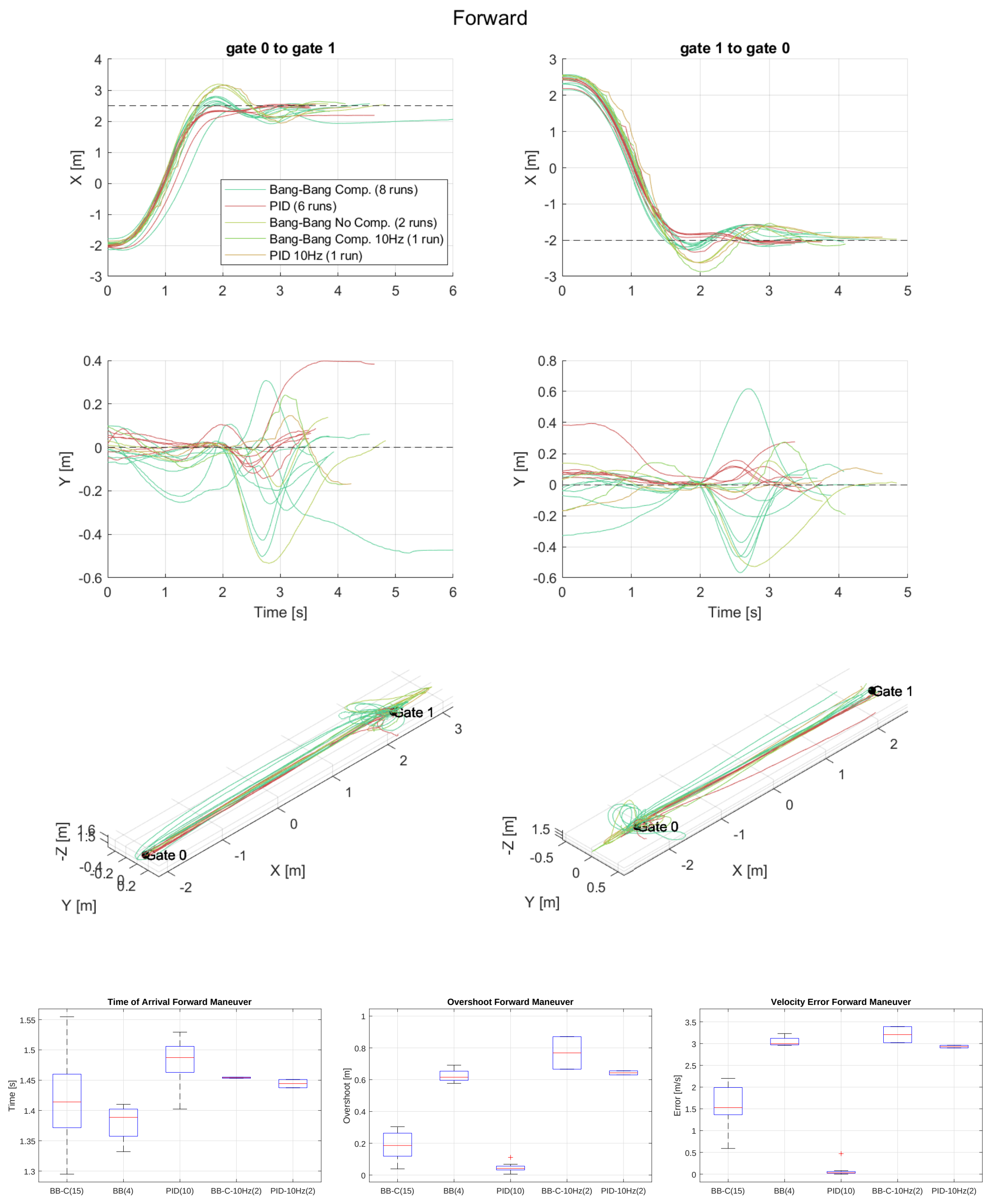


Figure C.3: Forward Maneuver

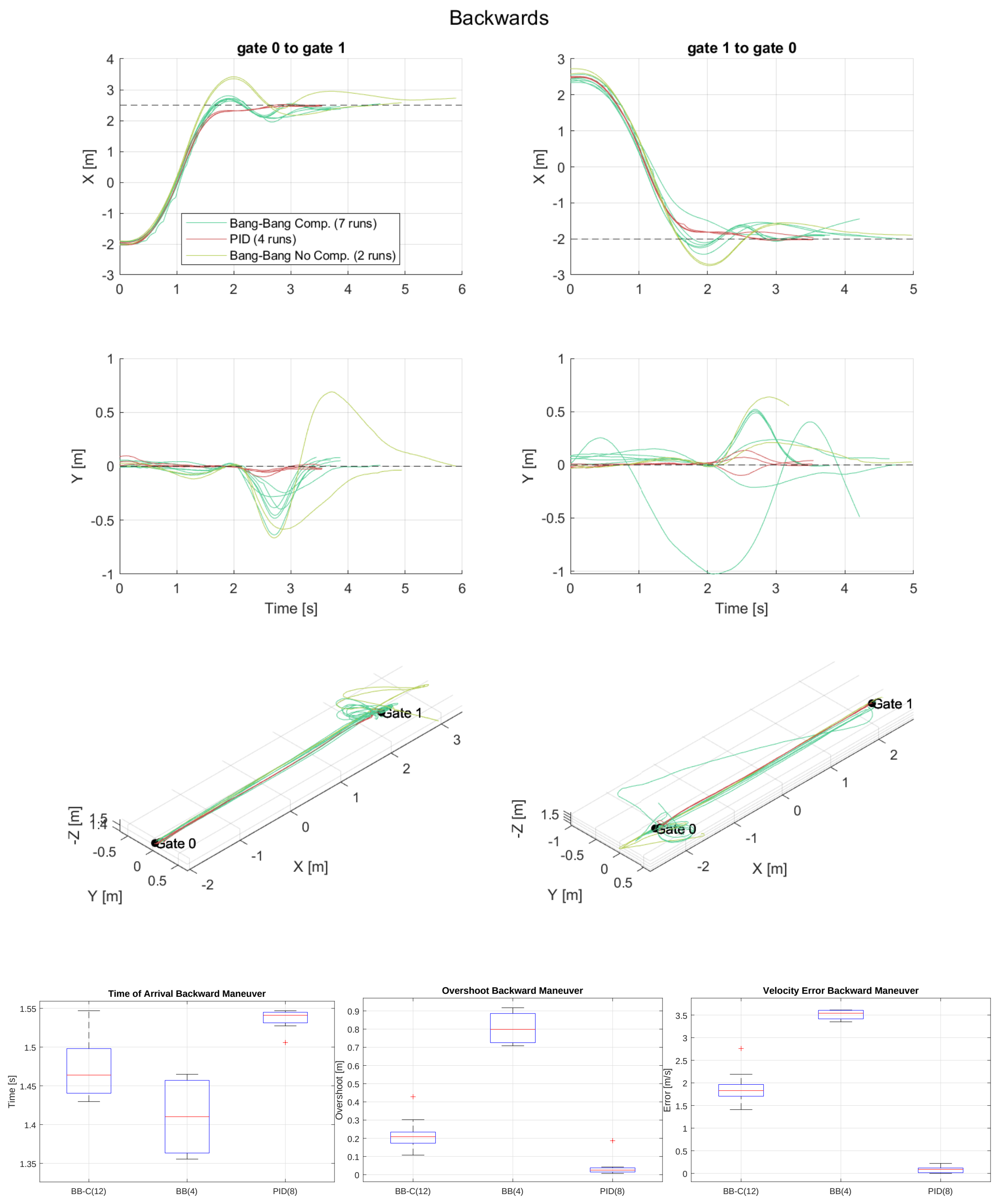


Figure C.4: Backward Maneuver

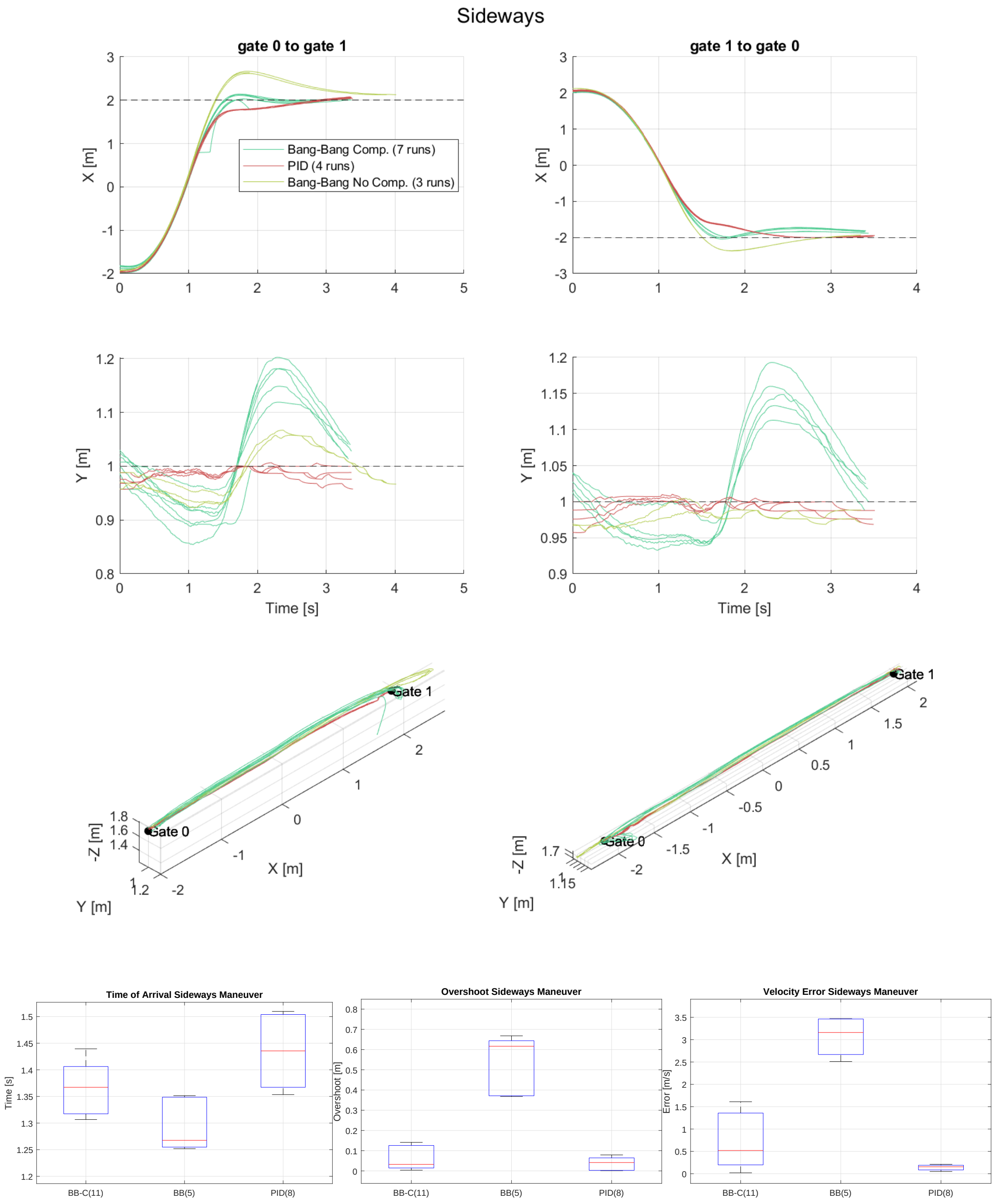


Figure C.5: Sideways Maneuver

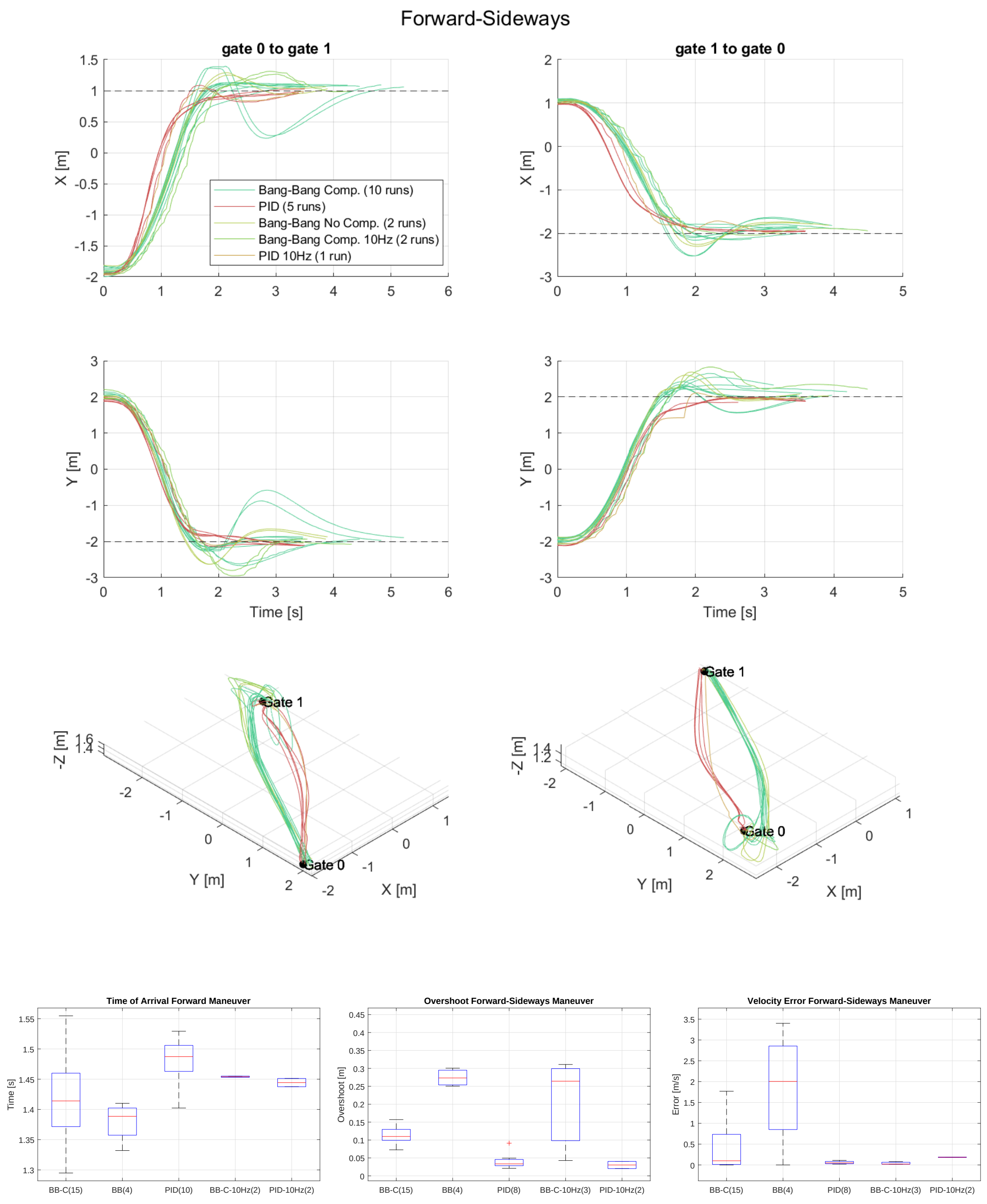


Figure C.6: Forward-Sideways

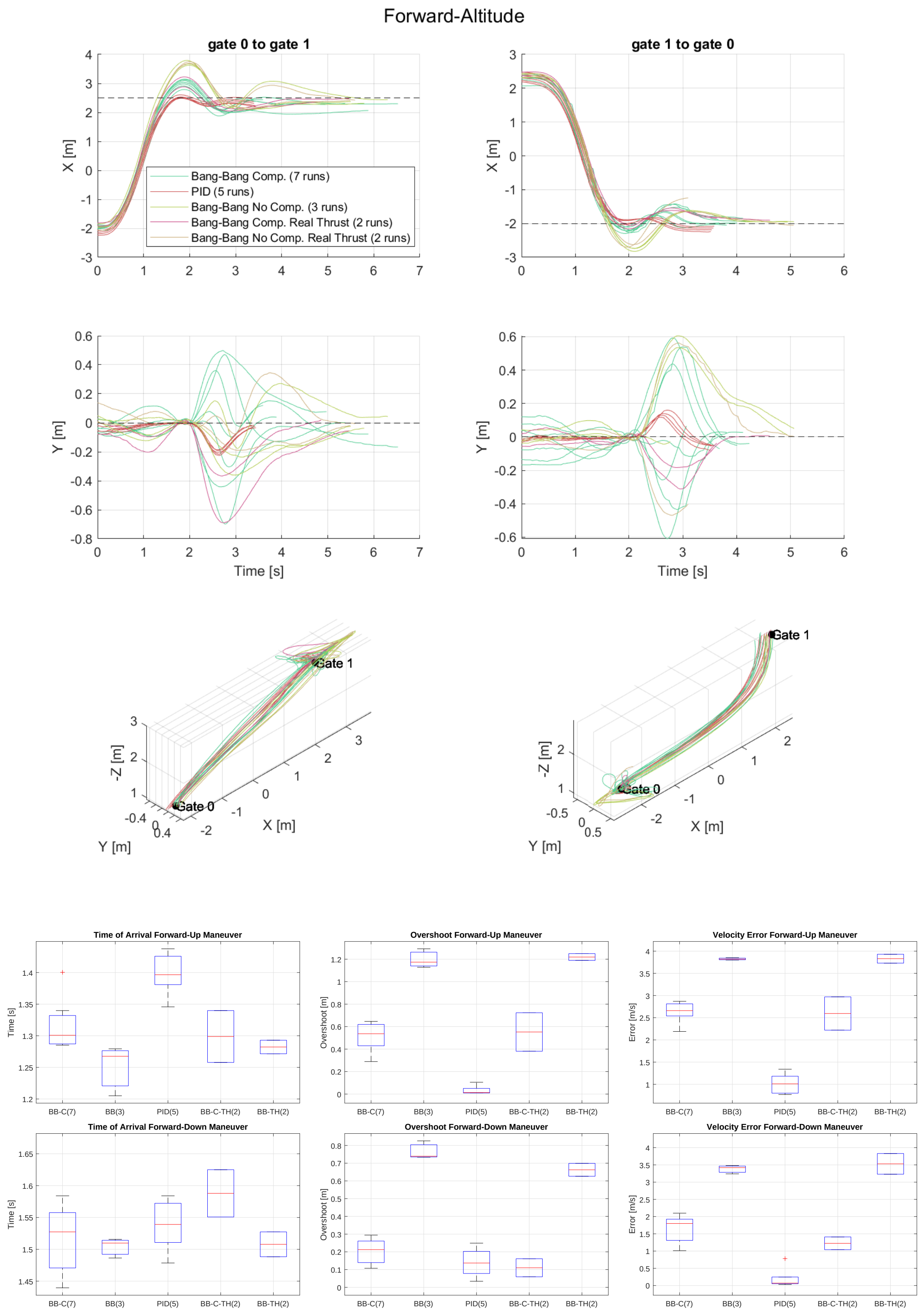
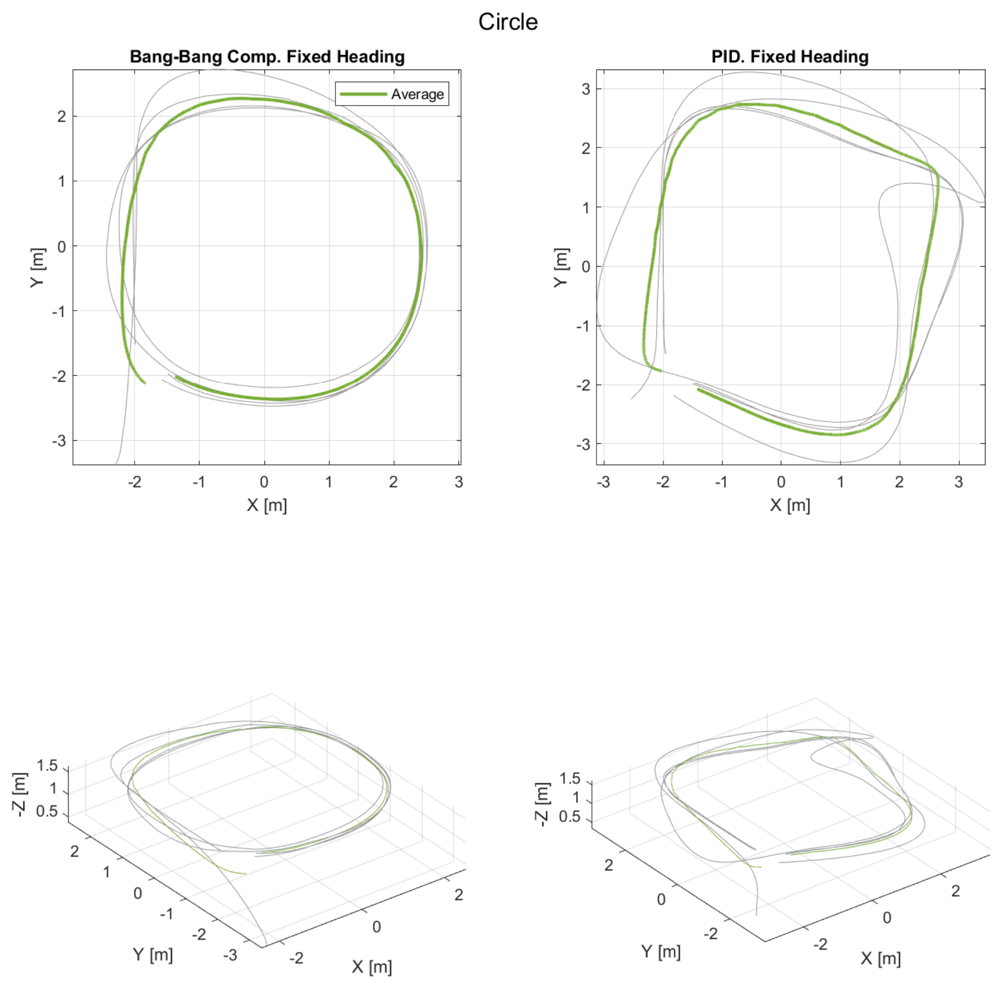
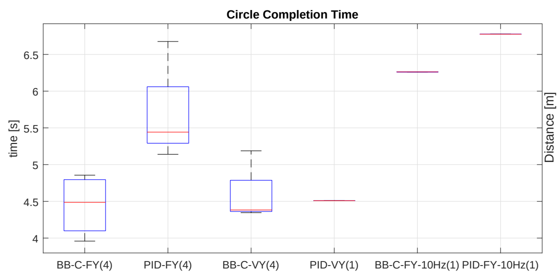
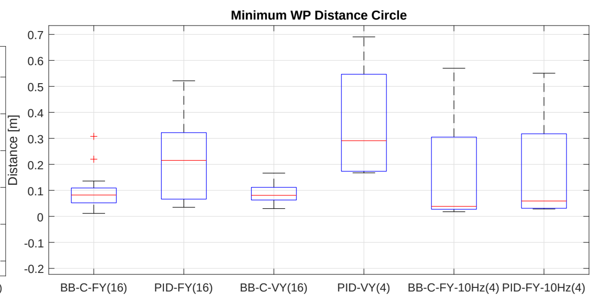


Figure C.7: Forward-Up & Forward-Down


Circle
Bang-Bang Comp. Fixed Heading
PID. Fixed Heading
Average
X [m]
Y [m]
-Z [m]


Circle Completion Time
time [s]
BB-C-FY(4)
PID-FY(4)
BB-C-VY(4)
PID-VY(1)
BB-C-FY-10Hz(1)
PID-FY-10Hz(1)


Minimum WP Distance Circle
Distance [m]
BB-C-FY(16)
PID-FY(16)
BB-C-VY(16)
PID-VY(4)
BB-C-FY-10Hz(4)
PID-FY-10Hz(4)