Diss. ETH No. 21881

Optimality-Based Trajectory Generation and Periodic Learning Control for Quadrocopters

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by

MARKUS HEHN

Diplom-Ingenieur, Technische Universität Darmstadt

born 14th March 1985
citizen of Germany

accepted on the recommendation of

Prof. Dr. Raffaello D'Andrea, examiner
Prof. Dr. Vijay Kumar, co-examiner
Prof. Dr. Roy Smith, co-examiner

2014

Optimality-Based Trajectory Generation and Periodic Learning Control for Quadrocopters

Markus Hehn

Institute for Dynamic Systems and Control
ETH Zurich
Zurich, 2014

Institute for Dynamic Systems and Control
ETH Zurich
Switzerland

© 2014 Markus Hehn. All rights reserved.

Abstract

This thesis considers the design of algorithms for the autonomous flight of quadrocopters and related multi-rotor vehicles. These flying vehicles have proven popular as small-scale aerial robotics platforms because they combine mechanical simplicity and robustness, the ability to hover, and exceptional agility.

The algorithms proposed herein perform two tasks: 1) they generate high-performance flight trajectories that satisfy the dynamic and input constraints of quadrotor vehicles; and 2) they track a given, periodic trajectory with high accuracy by using past data.

The first part of this thesis presents several algorithms centered around the problem of generating flight trajectories for quadrocopters. The first algorithm presented here is a benchmarking tool (Paper I). Its aim is to quantitatively answer questions surrounding achievable performance, influence of design parameters, and performance assessment of control strategies in the design process of quadrotor systems. The algorithm allows the computation of quadrotor maneuvers that satisfy Pontryagin's minimum principle with respect to time-optimality. Such maneuvers provide a useful lower bound on the duration of maneuvers, which can be used to assess performance of controllers and vehicle design parameters. Computations are based on a two-dimensional first-principles quadrotor model. The minimum principle is applied to this model to find that time-optimal trajectories are bang-bang in the thrust command, and bang-singular in the rotational rate command. The algorithm allows the computation of time-optimal maneuvers for arbitrary initial and final states by solving the boundary value problem induced by the minimum principle.

While the benchmarking algorithm provides valuable insight in the design stage, its computational complexity and use of a two-dimensional model disqualifies it from planning flight trajectories dynamically. Dynamic trajectory planning is required for tasks subject to dynamic changes of the objective, where significant disturbances in the flight path may occur, or where knowledge of the environment becomes available only mid-flight. In such circumstances, it is necessary to quickly and robustly plan a new trajectory whenever new information becomes available. The second algorithm in this thesis is designed to satisfy such real-time requirements (Paper II). Its aim is to combine computational efficiency and the ability to plan fast motions from a large range of initial conditions to a target point that will be reached at rest. The approach consists of planning separate trajectories in each of the three translational degrees of freedom, and to ensure feasibility by deriving decoupled constraints for each degree of freedom through approximations that preserve feasibility. This algorithm can compute a feasible trajectory within tens of microseconds on a laptop computer, and remaining computation time can be used to iteratively improve the trajectory. By replanning the trajectory at a high rate, the trajectory generator can be used as an implicit feedback law similar to model predictive control. The approach of decoupling the trajectory generation problem through feasibility-preserving approximations of the vehicle motion constraints is applicable to a broader class of trajectory generation problems, as is demonstrated by an extension to interception maneuvers (Paper III).

The second part of this thesis presents algorithms that use past data (measurements of past executions of the same motion) to improve tracking performance during the execution of periodic maneuvers under feedback control. The complex high-speed flight dynamics of quadrocopters are typically simplified when designing feedback controllers, and are subject to large parameter uncertainties. This can lead to significant tracking errors during high-performance flight, and repeated execution typically leads to a majority of the tracking errors being repeated. This part of the thesis introduces iterative learning schemes that non-causally compensate repeatable trajectory tracking errors during the repeated execution of periodic flight maneuvers (Papers IV & V). The schemes augment conventional feedback controllers by providing additional feedforward correction inputs based on data gathered during past executions. The learning is carried out in the frequency domain, and is based on a Fourier series decomposition of the input and output signals. The algorithm is extended by a time scaling method that allows the transfer of learnt maneuvers to different execution speeds through a prediction of the disturbance change. This allows the initial learning to occur at reduced speeds, and thereby extends the applicability of the algorithm to high-performance maneuvers.

All algorithms presented in this thesis have been verified experimentally on small-scale quadrocopters in the Flying Machine Arena test bed. These tests confirm the validity of modeling assumptions and show the real-world performance of the methods presented. The second part of this thesis features complimentary experiments designed to demonstrate the applicability of the learning algorithms developed earlier to more complex tasks: in this case, a quadrocopter balancing an inverted pendulum (Paper VI).

These algorithms have also been featured in more than 150 live demonstrations at ETH Zurich and various international events, highlighting their robustness and reliability in real-world settings.

Kurzfassung

Diese Dissertation präsentiert Algorithmen für den autonomen Flug von Quadrokoptern und ähnlichen Vielrotorfluggeräten. Auf Grund ihres simplen und robusten mechanischen Aufbaus, ihrer Schwebefähigkeit und ihrer ausserordentlichen Agilität haben sich derartige Fluggeräte zu beliebten Plattformen in der Flugrobotik entwickelt, und sind insbesondere im Bereich relativ niedriger Nutzlasten stark vertreten.

Die in dieser Arbeit präsentierten Algorithmen erfüllen zwei Aufgaben: 1) Die Berechnung von Flugtrajektorien unter Berücksichtigung der Dynamik und Aktuatorlimitationen von Quadrokoptern, sowie 2) das akkurate Folgen einer gegebenen periodischen Trajektorie durch Verwendung von Daten vorheriger Ausführungen.

Im ersten Teil der Dissertation werden mehrere Algorithmen zur Berechnung von Flugtrajektorien für Quadrokopter beschrieben. Ein erster Algorithmus dieser Kategorie dient zum Leistungsvergleich von Flugsystemen (Paper I). Ziel ist es hier, quantitative Antworten auf Fragen zur Leistungsfähigkeit von Flugsystemen und zum Einfluss von Systemparametern und -architektur auf die erzielbare Leistung zu berechnen. Der Algorithmus erlaubt die Berechnung von Flugmanövern, die die Optimalitätsbedingungen des Pontrjaginschen Maximumprinzips für die Minimierung der Manöverdauer erfüllen. Die so errechnete Manöverdauer stellt die Untergrenze für die Dauer des Manövers dar, und kann als Referenzwert für die Leistungsfähigkeit des Systems verwendet werden. Zur Berechnung der Manöver wird ein zweidimensionales Quadrokoptermodell verwendet. Anhand des Pontrjaginschen Maximumprinzips wird hergeleitet, dass die optimalen Steuersignale für den Gesamtschub immer maximal oder minimal sind, während jene für die Drehrate maximal, minimal oder durch eine Singularitätsbedingung gegeben sind. Der Algorithmus erlaubt die Berechnung von zeitoptimalen Flugmanövern für beliebige Anfangs- und Endzustände, indem das sich durch das Maximumprinzip ergebende Randwertproblem gelöst wird.

Der obige Algorithmus liefert wertvolle Informationen zur Auslegung von Flugsystemen, ist jedoch aufgrund seines hohen Rechenaufwandes und seiner Beschränkung auf ein zweidimensionales Modell für die dynamische Planung von Flugbahnen ungeeignet. Solch eine dynamiche Planung ist nötig, wenn während der Ausführung eines Manövers Änderungen des Zielpunktes oder signifikante Störungen auftreten, oder neue Informationen zu Hindernissen in der Umgebung verfügbar werden. In solchen Szenarien ist es nötig, schnell und zuverlässig neue Flugbahnen zu planen sobald neue Informationen zur Verfügung stehen. Der zweite in dieser Arbeit vorgestellte Algorithmus (Paper II) wurde in Hinsicht auf solche Echtzeitbedüfnisse entwickelt. Das Ziel ist in diesem Kontext, bei geringem Rechenaufwand schnelle Flugtrajektorien aus einem grossen Bereich von Anfangszuständen zu einem Zielpunkt, der im Stillstand erreicht wird, zu planen. Der hierfür verwendete Ansatz besteht aus einer Entkopplung der drei translatorischen Freiheitsgrade, wobei mittels Begrenzungen der Zustände und Stellgrössen der einzelnen Freiheitsgrade die dynamischen Eigenschaften des Quadrokopters berücksichtigt werden. So kann eine gültige Flugtrajektorie auf einem Laptop innerhalb von wenigen Mikrosekunden berechnet werden, und zusätzlich verfügbare Rechenzeit kann zur iterativen Verbesserung der Trajektorie verwendet werden. Das wiederholte Planen der Trajektorie mit hoher Frequenz erlaubt die Nutzung der Trajektoriengenerierung als impliziten Regelungsalgorithmus, und funktioniert in diesem Fall ähnlich wie eine modellprädiktive Regelung. Die Anwendbarkeit des Entkopplungsansatzes auf andere Trajektoriengenerierungsaufgaben wird in einer Erweiterung des Algorithmus anhand von Abfangtrajektorien demonstriert (Paper III).

Der zweite Teil dieser Dissertation präsentiert Forschungsergebnisse zu Algorithmen, die Daten aus vorherigen Experimenten verwenden um Fehler beim Folgen einer periodischen Referenztrajektorie zu reduzieren. Zur Auslegung von Flugreglern werden typischerweise vereinfachte Modelle der komplexen Flugdynamik von Quadrokoptern verwendet, die häufig mit signifikanten Parameterunsicherheiten behaftet sind. Hierdurch können sich grosse Folgefehler beim Flug von Hochleistungstrajektorien ergeben, die über mehrere Experimente hinweg zu grossem Teile wiederholbar sind. In diesem Teil der Arbeit werden Algorithmen präsentiert, die in diesem Falle die nicht-kausale Korrektur solcher wiederholbarer Folgefehler erlauben (Paper IV & V). Hierzu werden konventionelle Regler durch eine zusätzliche Vorsteuerung ergänzt, die Messdaten vorheriger Experimente verwendet. Das Lernen des Vorsteuerungssignals erfolgt im Frequenzbereich und basiert auf einer Fourierzerlegung der Eingangs- und Ausgangssignale des Systems. Zusätzlich wurde eine Zeitskalierungsmethode entwickelt, mit der der Lernprozess während des langsamen Ausführens des Manövers beginnen kann, und das gelernte Korrektursignal dann auf höhere Manövergeschwindigkeiten übertragen werden kann.

Alle in dieser Dissertation vorgestellten Algorithmen wurden mit Quadrokoptern in der Flying Machine Arena experimentell validiert. Diese Tests demonstrieren ihre Leistungsfähigkeit und bestätigen die in der Entwicklung der Algorithmen verwendeten Annahmen. Die Lernalgorithmen des zweiten Teils wurden weiterhin anhand von einem Experiment validiert, bei dem ein Quadrokopter ein inverses Pendel stabilisiert (Paper VI). Diese Experimente dienen dazu, die Anwendbarkeit der Methoden auf komplexere Fluganwendungen zu bestätigen.

Zusätzlich wurden die Algorithmen auch im Rahmen von Vorführungen sowohl an der ETH Zürich als auch an internationalen Veranstaltungen präsentiert. Über 150 Live-Demonstrationen bestätigen die Robustheit und Zuverlässigkeit der dargestellten Algorithmen.

Acknowledgments

Many people have contributed directly and indirectly to the research presented in this thesis, and this thesis would not have been possible without them. I would like to use this opportunity to thank some of them, knowing full well that this list cannot be comprehensive.

My sincerest gratitude goes to my doctoral advisor Professor Raffaello D'Andrea. He put his trust in me by accepting me as a doctoral candidate as well as giving me the freedom to work on the topics I considered interesting. I have benefited immensely from our numerous discussions and his guidance, insight, and inspiration. For this, I wish to say a heartfelt thank you to him. Raff has created a truly exceptional environment at the Institute for Dynamic Systems and Control (IDSC), working in which felt more like fun than work and where there always were more ideas than time.

I also sincerely thank Professor Vijay Kumar and Professor Roy Smith for agreeing to be members of my doctoral committee and their willingness to spend considerable time reviewing this thesis. I greatly appreciated their insightful questions and comments.

My time at the IDSC was spent in the company of many friendly, helpful, and extremely intelligent colleagues. Our countless discussions were always interesting, and often helped a great deal in progressing our individual or joint research projects. I have learnt a lot from these colleagues, and now consider many of them to be good friends.

My closest collaborators throughout the doctorate were the other doctoral candidates involved in the Flying Machine Arena project: Sergei Lupashin, Angela Schöllig, Mark Müller, Robin Ritz, Federico Augugliaro, and Dario Brescianini. To work in a group of highly motivated, skillful, and talented people has been a great pleasure and a hugely educational and rewarding experience. I feel that our work often showed that the whole really can be greater than the sum of its parts.

While we did not work on the same project, my office mate Philipp Reist and I shared teaching assistant duties and, by merit of starting at IDSC on the same day, progress and struggles of our doctorates. I truly appreciated working with him as well as our discussions and out-of-work projects. I also thoroughly enjoyed sharing teaching tasks with Sebastian Trimpe, who in his methodic, thought-through and structured approach to problems was a role model to me throughout this doctorate.

The IDSC supports its doctoral candidates with its excellent support staff. I would like to thank Katharina Munz for her organizational work, Markus Waibel for his management

of numerous demos and his help in disseminating our research results, Hallie Siegel for proof-reading and improving our publications, Carolina Flores for her design and media work, and Igor Thommen, Marc-André Corzillius, and Hans Ulrich Honegger for their technical support in the Flying Machine Arena.

I am greatly indebted to the great students that I had the pleasure to work with in the past four years. Many of them contributed significantly to the results of this thesis, and I am happy to see that many now pursue a doctorate themselves. I would like to particularly thank Luzius Brodbeck, Robin Ritz, and Dario Brescianini for their excellent work that directly improved and extended this research.

A great number of people have supported and guided me throughout the years, and the influence of many has shaped my life. Of all of them, the most important ones were certainly my parents Silvia and Wolfgang, who always supported and encouraged me.

Finally, I owe a huge thank you to Anna Hänsli, who has accompanied me during the past five years. She has always stood by my side unconditionally, and gave me whatever support I needed.

Zürich, Spring 2014 *Markus Hehn*

Financial Support

The author gratefully acknowledges partial funding of this research by the Swiss National Science Foundation (SNSF) under the grant “High-Performance Maneuvers and Trajectory Generation for Quadrotor Flying Vehicles” and the equipment grant “Optical Motion Capture System for Robot Experiments in Real World Environments”.

Contents

Preface 15

1. Introduction 17
- 1.1 Part A: Trajectory Generation 19
- 1.2 Part B: Iterative Learning of Periodic Motions 20

2. Contributions 23
- 2.1 Part A. Trajectory Generation 23
- 2.2 Part B. Iterative Learning of Periodic Motions 27
- 2.3 List of Publications 30
- 2.4 List of Supervised Student Projects 32
- 2.5 Outreach 33

3. Future Directions 35
- 3.1 Part A. Trajectory Generation 35
- 3.2 Part B. Iterative Learning of Periodic Motions 37

References 38

A. TRAJECTORY GENERATION 41

Paper I. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control **43**
- 1. Introduction 44
- 2. Modeling of Vehicle Dynamics 45
- 3. Minimum Principle for Time-Optimal Maneuvers 48
- 4. Algorithm for Calculation of Time-Optimal Maneuvers 53
- 5. Numerical Results 56
- 6. Benchmarking of Quadrotor Designs and Controllers 61
- 7. Experimental Results 65
- 8. Conclusion and Future Work 68
- A. Algorithm for Calculation of Time-Optimal Maneuvers 69
- References 79

Paper II. Real-Time Trajectory Generation for Quadrocopters 83
1. Introduction 84
2. Dynamic Model 86
3. Problem Statement 89
4. Feasibility Conditions and Decoupling 89
5. Decoupled Trajectory Planning 96
6. Choice of Design Parameters 100
7. Use as a Feedback Law 103
8. Performance Evaluation 104
9. Experimental Validation 108
10. Conclusion 111
A. Feasibility of Time-Varying Acceleration Constraints 113
References 115

Paper III. Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters 119
1. Introduction 120
2. Vehicle Dynamics 121
3. Trajectory Generation Algorithm 122
4. The Interception Maneuver 124
5. Computation and Verification of Interception Maneuvers 128
6. Experimental Results 129
7. Conclusion & Outlook 131
References 132

B. ITERATIVE LEARNING OF PERIODIC MOTIONS 135

Paper IV. A Frequency Domain Iterative Learning Algorithm for High-Performance, Periodic Quadrocopter Maneuvers 137
1. Introduction 138
2. Quadrocopter Dynamics and Closed-Loop Control 141
3. Learning Algorithm 145
4. Experiments 154
5. Advantages and Limitations 162
6. Conclusion 165
References 165

Paper V. An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories **171**
1. Introduction 172
2. The Flying Inverted Pendulum Experiment 174
3. Learning Algorithm 177
4. Experimental results 181
5. Conclusion and Outlook 183
References 184

Paper VI. A Flying Inverted Pendulum **187**
1. Introduction 188
2. Dynamics 188
3. Nominal trajectories 192
4. Dynamics about nominal trajectories 195
5. Controller Design 197
6. Experimental results 197
7. Conclusion and Outlook 203
References 204

Curriculum Vitae 207

Preface

This thesis documents the research carried out by the author during his doctoral studies under the supervision of Professor Raffaello D'Andrea at the Institute for Dynamic Systems and Control (IDSC) at ETH Zurich between September 2009 and April 2014.

The thesis is presented in the form of a cumulative dissertation: its main content consists of six self-contained research articles (of which three are journal articles and three are conference contributions) that have been published or submitted for publication during the doctoral studies. The research is structured into two classes of problems: The generation of flight trajectories for quadrocopters (Part A), and the precise tracking of periodic trajectories through learning methods (Part B). The articles are put into context by three introductory chapters, which are structured as follows:

Chapter 1 sketches out the motivation for this work. It also describes the problems considered in this thesis, related work, and the approaches used. Chapter 2 describes the key contributions of the research papers included in this thesis and how the individual papers relate to each other. This chapter also includes references to related papers, created in the context of the author's doctoral studies, that are not included in this thesis. Chapter 3 then provides a discussion of potential extensions and new directions of this research.

1

Introduction

This thesis presents algorithms for the autonomous high-performance flight of quadrocopters. The algorithms provide means to plan flight paths and to accurately track periodic motions, with a particular focus on the combination of computational efficiency and high flight performance.

Quadrocopters are robotic platforms that are frequently used in applications where the ability to move freely in three-dimensional space and to overcome obstacles on the ground are crucial [32]. Their high mechanical robustness (due to few moving parts) [27], their safety (due to comparatively small rotor sizes) [15], and their ability to engage in high-performance maneuvers and carry large payloads (due to their typically high thrust-to-weight ratio) [18] are significant advantages over other small robotic platforms.

To further increase payload capabilities without increasing the rotor size, and to increase robustness to actuator failures, other multi-rotor configurations are also being designed and used [18]. Popular designs include hexacopters (with six propellers) and octocopters (eight propellers). While this thesis is focused on quadrocopter applications, the algorithms also apply directly to configurations with more rotors mounted in parallel, in particular typical hexacopters and octocopters [1].

This thesis considers the trajectory generation and control of quadrocopters during high-performance flight. In this context, high performance is used to highlight that the algorithms are designed to allow full use of a vehicle's capabilities. While this is desirable for many applications (for example, it may allow faster execution of a given task or higher accuracy), the ability to exploit the dynamic potential of the vehicle can also be seen as a safety feature: even if the planned motion does not require the use of a vehicle's full dynamic capabilities, their efficient use may be crucial in recovering from unexpected disturbances or avoiding collisions when obstacles are detected mid-flight.

From a controls perspective, high-performance quadrocopter flight is a challenging proposition for several reasons:

- The **inherently unstable and nonlinear dynamics** typically limit the use of linear control laws to flight near hover or near a trajectory planned ahead of time (see e.g. [17], and references therein).

- Quadrocopters exhibit **fast dynamics**, in particular in the rotational degrees of freedom due to their typically low rotational inertia combined with the off-center mounting of the propellers. Flying fast maneuvers and effectively suppressing disturbances thus requires fast control loops and therefore limits the amount of computation that can be carried out.

- Significant **model uncertainties** arise when the maneuvering speed increases, for example due to the varying drag and lift behaviour of the propellers under unsteady inflow conditions [6], the aerodynamic effects of a vehicle moving through the turbulent wake of its propellers [2], and external influences such as wind or ground and wall effects when operating in proximity to the environment [28]. These uncertainties diminish the effectiveness of model-based control strategies.

The research presented in this thesis addresses two problems related to the control of quadrocopters. The key objective of the research presented in part A of this thesis is the generation of dynamic trajectories for quadrotor vehicles. Part B then focuses on the design of algorithms that can be used to track a predefined, periodic trajectory with high accuracy by using data from past executions.

A central part of this research project is the experimental validation of the developed algorithms in order to demonstrate their effectiveness under real-world conditions and provide insight into their characteristics. Each of the algorithms presented in this thesis was applied to small quadrocopters (see Figure 1.1), and experimental results are presented for each of the research contributions. The experiments were carried out in the Flying Machine Arena [20], a platform for aerial robotics research that was co-developed by the author throughout the course of his doctoral studies.

The following sections specify the objectives and context of each of the two parts of the research in more detail, and discuss the different approaches presented in the individual contributions.

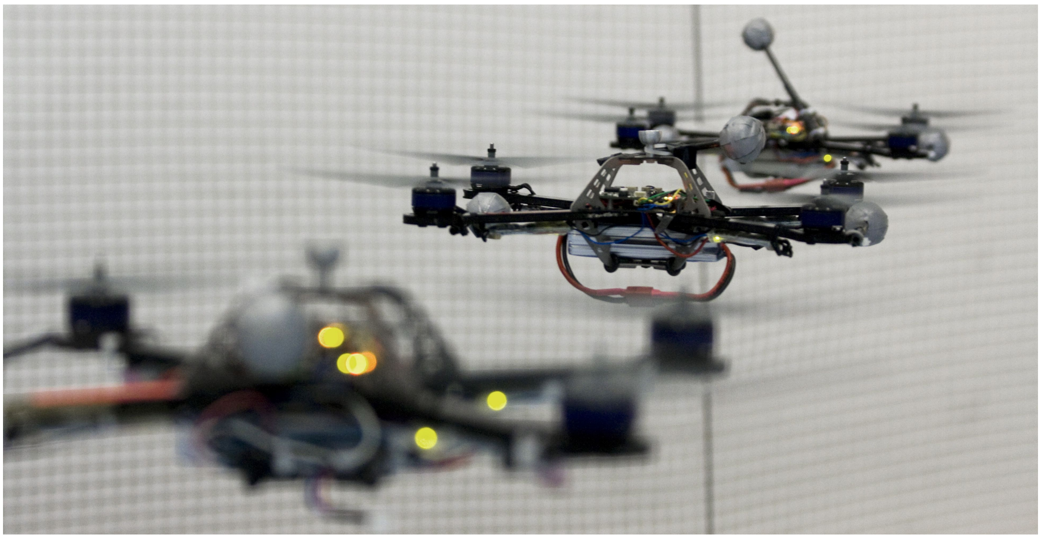

Photo by Raymond Oung

Figure 1.1 Quadrocopters used for the experimental validation of algorithms. These vehicles measure approximately 35 cm from propeller center to propeller center, and weigh about 500 g.

1.1 Part A: Trajectory Generation

Trajectory generation algorithms form a core component in unmanned aerial vehicle flight systems. They are closely interlinked with the control of the vehicle, and provide crucial information (such as arrival time at a specified point or flight paths) to high-level planning systems, e.g. mission planners or collision avoidance systems. Their impact on overall system performance and efficiency is significant, as they typically determine the speed of task execution. This is made even more important by severe limitations in achievable flight time of state-of-the-art quadrocopters.

The problem can be stated as follows: Given the current state $\mathbf{x}_0$, plan a trajectory to a target state $\mathbf{x}_T$. The trajectory must be feasible with respect to the dynamics of the quadrocopter. A multitude of trajectory design criteria may be desirable, e.g. reaching the target state at a specified time, minimizing energy consumption, avoiding a certain state space region, or arriving at the target state as quickly as possible. In its general form, the trajectory generation problem is the search for the control input trajectory that minimizes the performance metric, subject to constraints of the system dynamics and possibly other design constraints, and terminates at the specified terminal state.

Related Work

The problem of quadrocopter trajectory generation has received significant attention in recent years, and a number of algorithms have been presented. Broadly speaking, a possible categorization of the algorithms is as follows:

A first group of algorithms can be considered as primarily geometric. The trajectory generation process consists of first generating a path in space from a class of path primitives, and thereafter parameterizing the generated path in time such that the dynamic constraints of the quadrocopter are enforced. Examples of such algorithms have been presented using path primitives such as lines [16], polynomials [9], or splines [5].

A second group of algorithms is based on the design of trajectories that minimize a derivative of the position trajectory (or combinations thereof). Because these derivatives can be related to the control input constraints of the quadrocopter through its differential flatness property, the feasibility of the trajectory depends on these derivatives. Examples of such methods include minimum snap trajectory generation [21] and the minimization of a weighted sum of derivatives [30]. More real-time-focused implementations are based on model predictive control methods with learnt linear dynamics [4] or linear dynamics based on a decoupling of the system [24].

Research Directions

The specific trajectory design criterion that this thesis focuses on are minimum-time trajectories, meaning that the design objective is to minimize the maneuver's duration until the target state is reached. For high-performance flight, minimum-time maneuvers are of particular interest because they determine a fundamental performance limit of the system considered. They can provide definitive answers to questions such as "If a maneuver must start at the state $\mathbf{x}_0$ and end at the state $\mathbf{x}_T$, what is the minimum time required to execute it?" and "Is it possible to reach state $\mathbf{x}_T$ from the initial state $\mathbf{x}_0$ within some given duration?" [14].

The first direction of research presented herein is the design of algorithms that compute time-optimal flight maneuvers for quadrocopters. The computed trajectories can be used not only to execute maneuvers, but also as a benchmarking tool that provides valuable insight into the capabilities of the quadrotor system. Time-optimal maneuvers can highlight the influence on performance of such varied design parameters as the vehicle weight, the measurement range of the internal sensors, or the feedback control law. The application of state-of-the-art optimal control techniques (e.g. [3, 11]) to compute these time-optimal maneuvers is made difficult by the nonlinearity and high order of the quadrotor dynamics. A key challenge when designing algorithms that are capable of computing these maneuvers is thus to find suitable combinations of problem class restrictions (to reduce complexity) and numerical methods (to improve computational tractability). For these results to maintain their value as a benchmarking tool, it is particularly important that the generated trajectories are indeed time-optimal.

The second focus of this research is the design of algorithms that can generate trajectories quickly enough to be deployed in real-time scenarios. This allows a system to cope with changing requirements, which in the case of quadrotor vehicles may be caused by rapidly changing destinations, suddenly appearing obstacles, or the recovery from arbitrarily perturbed states (caused, for example, by large wind gusts). Running the trajectory generation algorithm at every controller update allows its use as an implicit feedback law: closed-loop control is achieved by re-generating a trajectory at each controller update, and applying the first control inputs of it in the manner of model predictive control [10]. The combination of long, variable planning horizons, nonlinear dynamics, and short available calculation times makes this a challenging optimization problem. Compared to the benchmarking problem, the focus here is shifted away from finding truly time-optimal maneuvers, towards finding maneuvers of relatively short duration under stringent real-time requirements. In this context, it is acceptable for the generated trajectories to not satisfy any optimality conditions; for the use in real-time scenarios, the absolute performance of planned trajectories is secondary to guaranteeing the finding of a feasible trajectory of reasonable performance within the given computation time constraints.

1.2 Part B: Iterative Learning of Periodic Motions

The second part of this thesis considers problems where the flight trajectory is defined ahead of flight, and the objective is to track this trajectory as accurately as possible. Accurate trajectory following is particularly relevant for operations where the tracking performance of the system determines the quality of the task. Examples include carrying a hanging payload with one or multiple quadrocopters, inspecting an object (like a bridge) along a pre-computed path, painting surfaces that are difficult to access, or filming a scene with a camera mounted on the vehicle.

The tracking problem complements the research projects presented in Part A: whereas real-time trajectory generation algorithms are useful for dynamically changing tasks or environments, we now consider problems where the trajectory is fully defined ahead of flight. The large model uncertainties, in particular at high maneuvering speeds, make the accurate tracking of high-performance trajectories challenging because they effectively act as disturbances to the closed-loop trajectory tracking system.

In the research presented herein, we consider the specific case where the same trajectory can be flown repeatedly. This offers an opportunity to improve tracking performance because many of the disturbances that degrade tracking performance will be similar each time the vehicle performs the motion. It is thus possible to non-causally correct for these repeatable disturbances: by using data from previous executions to characterize them, model-based correction inputs can be computed before executing the motion again. Ideally, these correction inputs are able to fully compensate for the repeatable disturbances, such that the feedback controller is required only to compensate for non-repeatable disturbances.

Related Work

While the application of learning algorithms (and specifically non-causal strategies) to stationary systems is well-established, its use for flying vehicles is less mature and has only been actively researched in recent years. Several high-performance maneuvers for multi-rotor vehicles have been demonstrated with the use of learning algorithms. Broadly speaking, these learning approaches can be categorized in two groups:

The first group is characterized by its ability to learn motions that are parameterized. The motion is thus described by a (finite) set of design parameters chosen by the user. After the execution of the motion, these parameters are adapted to compensate for observed disturbances. The direction and magnitude of the correction may be model-based, or based on the user's intuition. A discussion on the importance of choosing 'good' design parameters may be found in [19], where a learning algorithm for this kind of parameterized motion is demonstrated for multiple flips and fast translations with quadrocopters. A further demonstration of this class of learning algorithms is provided in [22]. The ability to shape the tracking performance strongly depends on the number of parameters that are optimized; in the above examples, the objective is to minimize the error at specific time instants ('key frames'), and a relatively small number of parameters is sufficient to do so. This makes the methods computationally lightweight.

The second group of learning approaches considers more generic motions that need not be specified by parameters. The system dynamics are considered in discrete time, and the correction consists of correction values (typically control inputs or set points) for each discrete time step. After executing the motion, the correction values are numerically optimized in order to minimize a performance metric related to the tracking error. In this optimization, a model of the system dynamics provides the mapping from

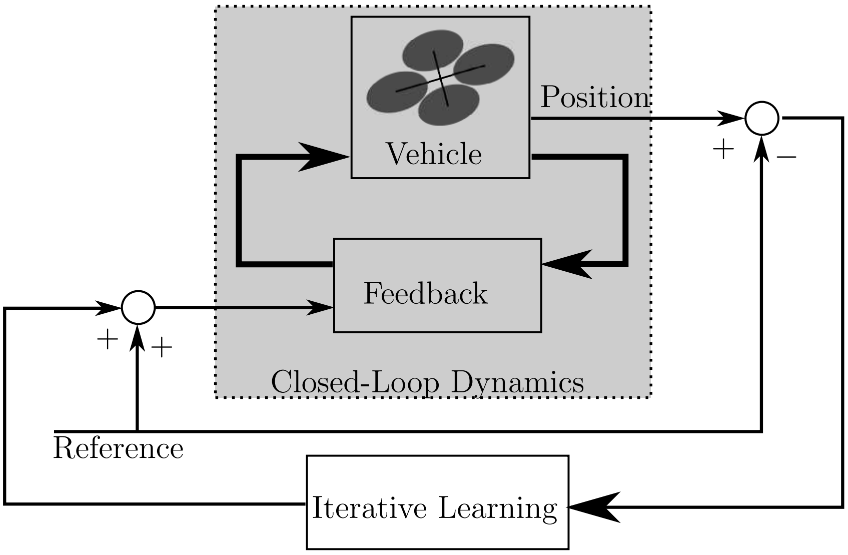

Figure 1.2 Architecture of a learning algorithm augmenting feedback control. The reference to the feedback controller is augmented by adding the learning control input to it. The feedback control runs in real-time, while the iterative learning applies feedback based on past executions. In this configuration, the iterative learning component can non-causally compensate repeatable disturbances, while the feedback control compensates non-repeatable disturbances.

corrections to the tracking error. This approach is commonly known as a form of iterative learning control [7], and its application to high performance quadrocopter flight has been demonstrated by several researchers [23, 25, 29, 31].

The delimitation between the two groups is not strict. Indeed, the second group of learning approaches could be seen as using a very large number of values to parameterize the correction.

Research Directions

The research in this thesis is centered on the development of algorithms for the learning of non-causal correction inputs for maneuvers that are periodic. It explores how the limitation to periodic maneuvers can be exploited in a learning control context, and how computationally inexpensive algorithms can be designed for this class of problems. To combine the strengths of feedback control and learning control, the learning control is used to augment the feedback controller as seen in Figure 1.2.

A particular focus of the algorithms in this thesis is the application of learning control to high-performance maneuvers. In this context, traditional learning control strategies may encounter a problem: in order to learn from past executions, the system must be able to execute the maneuver before learning control is applied. Due to the unstable dynamics of quadrocopters, and to physical constraints such as limited operating spaces, high-performance maneuvers may not be executable without the appropriate correction inputs from the learning control algorithm. This initialization problem requires a novel approach that provides a sufficiently good initial guess of the correction inputs.

2

Contributions

This chapter briefly summarizes the key contributions of the papers included in this thesis and explains the relationship between the individual results. Many of the results were obtained in close collaboration with other researchers at the Institute for Dynamic Systems and Control, as indicated below.

2.1 Part A. Trajectory Generation

Paper I

[P1] Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control. Autonomous Robots, Volume 33, Issue 1-2, pp 69-88, 2012.

Context In this paper, we consider the problem of generating time-optimal trajectories, i.e. maneuvers that minimize the time required for the transition from an initial to a final state. As discussed in Chapter 1, such maneuvers are of particular interest because they can be used as a benchmarking tool to analyze the performance and guide the development of quadrotor systems by answering performance-related questions such as "How do the physical parameters of the vehicle influence its performance?", "How much of the theoretical performance potential does a certain control strategy utilize?", and "Are there specific maneuvers for which a given controller performs particularly well or poorly?". The paper considers the optimal control problem for a two-dimensional normalized quadrotor model with two control inputs that are subject to saturation (the collective thrust and the rotational body rate). The omission of the third dimension makes the problem tractable by reducing its dimensionality, while still covering a class of maneuvers that is large enough to provide performance insight. In contrast to the other trajectory generation research presented in this thesis, computational efficiency is not of particular importance in this context because real-time trajectory generation is not necessary in a benchmarking context.

Contribution The paper presents the structure of time-optimal maneuvers that arises from the necessary optimality conditions given by Pontryagin's minimum principle [3]. It shows that time-optimal maneuvers are bang-singular: the thrust control is always maximal or minimal, and the rotational rate control input is maximal, minimal, or defined by a singular arc constraint. Based on this structure, the paper introduces an algorithm for computing the time-optimal maneuver for arbitrary initial and final states. To show that the approach yields results that are meaningful for real-life quadrocopters, the computed maneuvers are validated experimentally. The method's use as a benchmarking tool is demonstrated through the analysis of the performance implications of using a cascaded linear controller, and the influence of the maximum thrust the vehicle can produce.

Related Publications

[R5] Robin Ritz, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Performance Benchmarking Using Optimal Control. Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.

This paper presents an abridged version of the optimality conditions and trajectory generation algorithm, including preliminary results.

Paper II

[P2] Markus Hehn and Raffaello D'Andrea. Real-Time Trajectory Generation for Quadrocopters. IEEE Transactions on Robotics (under review).

Context This paper considers the problem of generating high-performance flight trajectories for quadrocopters under hard real-time constraints. As discussed in Chapter 1, such computational constraints typically arise when quadrocopters are flown in dynamically changing environments. The objective of the trajectory generation considered in this paper is to plan trajectories from a large range of initial conditions reaching a target point at rest, and for these trajectories to reach this target point as quickly as possible. As opposed to [P1], the full three-dimensional planning problem is considered, and the design objective is to use the trajectory generator as a feedback control law by executing it repeatedly in the fashion of model predictive control.

Contribution The contribution of this paper is the introduction and analysis of a trajectory generation algorithm that combines the ability to plan high-performance trajectories with computational efficiency. The approach is to plan separate trajectories in each of the three degrees of freedom. This is made possible by deriving decoupled feasibility constraints for each degree of freedom through approximations that preserve feasibility. The presented algorithm can compute a feasible trajectory within tens of microseconds on a laptop computer, and remaining computing time can be used to iteratively improve the trajectory. The paper includes the full description of the trajectory generation algorithm, an analysis of its computational and flight performance, and an experimental validation.

Related Publications

[R2] Angela Schoellig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasibility of Motion Primitives for Choreographed Quadrocopter Flight. Proceedings of the 2011 American Control Conference (ACC), 2011.

This paper presents the derivation of fundamental feasibility conditions for translational trajectories from the dynamics and input constraints of quadrocopters.

[R3] Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. Proceedings of the 2011 IFAC World Congress, 2011.

This paper presents the decoupling methodology used in this trajectory generation approach, and presents preliminary results on the use of the basic decoupled trajectory generation mechanism.

[R7] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter Pole Acrobatics. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

This paper presents a system that allows quadrocopters to balance an inverted pendulum, throw it into the air, and catch and balance it again on a second vehicle. The catching maneuver is planned in real time using the presented decoupling approach, and part of the motion is planned using the the decoupled trajectory generator.

[R11] Federico Augugliaro, Sergei Lupashin, Michael Hamer, Cason Male, Markus Hehn, Mark W. Mueller, Jan Willmann, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. The Flight Assembled Architecture Installation: Cooperative construction with flying machines. IEEE Control Systems Magazine (under review).

This paper presents the design and realization of a system that autonomously assembled a six-meter-tall tower consisting of 1500 foam modules with quadrocopters over six days in front of a live audience. The trajectory generation algorithm used in this system is an adapted version of the one presented in [P2].

Paper III

[P3] Markus Hehn and Raffaello D'Andrea. Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

Context This paper represents a continuation of the work in [P2]. While the results in [P2] focus on trajectory generation problems where the objective is to reach a target point at rest as quickly as possible, this work considers the problem of intercepting a position, i.e. planning a trajectory such that the vehicle is at a given position at a specified time. In addition to the interception constraint, the planned trajectories are designed to come to rest as quickly as possible after interception.

Contribution The main contribution of this paper is the adaptation of the trajectory generation algorithm presented in [P2] to the interception problem. Using the same decoupling approach, Pontryagin's minimum principle is used to show that the structure of an interception maneuver with minimal time to rest after the interception is identical to the structure of a maneuver that guides the vehicle to a target point. Thus, the trajectory generation algorithm from [P2] can be adapted to interception problems without increasing its computational complexity. The algorithm is validated experimentally by intercepting a ball thrown in the air.

Related Publications

[R6] Robin Ritz, Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. Cooperative Quadrocopter Ball Throwing and Catching. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

This paper presents the system design of an experiment in which multiple quadrocopters are attached to an elastic net, cooperatively launch a ball into the air by stretching the net, and catch it again. The real-time trajectory generation problem of intercepting the flying ball with the net is solved by applying a one-dimensional interception algorithm that is based on the same decoupling approach and shares the same maneuver structure.

[R9] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. A Computationally Efficient Algorithm for State-to-State Quadrocopter Trajectory Generation and Feasibility Verification. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

This paper considers the related problem of state interceptions, i.e. planning trajectories to a (partially or fully constrained) arbitrary target state at a specified target time. The method presented in this paper is based on a similar decoupling approach, but then implements an entirely different approach to solving the decoupled planning problem (in this paper, trajectories are designed to minimize jerk instead of time). This results in an algorithm of much lower computational complexity, but which can only verify feasibility *a posteriori*.

2.2 Part B. Iterative Learning of Periodic Motions

Paper IV

[P4] Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Learning Algorithm for High-Performance, Periodic Quadrocopter Maneuvers. Mechatronics (under review).

Context This paper presents a frequency-domain-based learning approach for periodic maneuvers that uses a serial architecture [7] (as shown in Figure 1.2). The objective of the iterative learning scheme is to minimize the position error of the quadrocopter by changing the set point of the position control loop. The approach exploits the ability to shape the closed-loop system dynamics by designing the feedback law appropriately; this in turn simplifies the design of the iterative learning algorithm. In particular, the closed-loop dynamics of a quadrocopter under feedback control can be approximated with sufficient accuracy by a linear time-invariant model. Furthermore, this paper addresses the problem of generating initial corrections for high-performance maneuvers in order to initialize the learning process and allow further improvements.

Contribution The contribution of this paper is the presentation and analysis of an iterative learning algorithm for quadrocopters that is formulated in the frequency domain and is applicable to periodic maneuvers. The algorithm combines fast convergence with low computational complexity. The statistical properties of repeatable and non-repeatable disturbances are used to derive the optimal correction inputs for each step of the learning process. For maneuvers that cannot be learnt directly, a time scaling approach is introduced. In this approach, the maneuver is initially learnt at reduced speed, and the learnt corrections from the reduced speed are used to initialize the learning at higher speeds. Experimental results show that the approach can reduce tracking errors nearly to the repeatability limit of the system and that it can be applied to more complex systems (as demonstrated by the flying inverted pendulum experiment in particular).

Related Publications

[R8] Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Feed-Forward Learning Scheme for High-Performance Periodic Quadrocopter Maneuvers. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

This paper presents an abridged version of the learning scheme including preliminary results. As opposed to [P4], the convergence properties are derived in a deterministic context.

[R10] Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A Platform for Aerial Robotics Research and Demonstration: The Flying Machine Arena. Mechatronics, Volume 24, Issue 1, pp 41-54, 2014.

This paper includes a more detailed presentation and analysis of the trajectory tracking controller that is used in the learning process, as well as describing the overall hardware and software infrastructure that is used for the experiments in the Flying Machine Arena.

Paper V

[P5] Markus Hehn and Raffaello D'Andrea. An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories. Proceedings of the 2013 European Control Conference (ECC), 2013.

Context This paper considers a similar problem setup to [P4]. The architecture of the learning system differs in that the learnt corrections are added directly to the control inputs instead of changing the reference signal to the feedback controller. A similar frequency domain learning approach is used, but the problem is formulated in a deterministic setting.

Contribution This paper extends [P4] to focus on the convergence of learning systems where the dimension of the tracking error and the correction input differ. The algorithm is designed to minimize an error output that consists of linear combinations of the system states, and error convergence conditions are derived. For systems where the correction input is larger or smaller than the number of error outputs, it is shown that the correction signal energy is minimized or that partial convergence is achieved, respectively. An experimental validation based on the flying inverted pendulum experiment confirms the effectiveness of the algorithm.

Paper VI

[P6] Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011.

Context The flying inverted pendulum is an experimental test bed that consists of a quadrocopter balancing an inverted pendulum. The feedback controller used to stabilize the inverted pendulum on the quadrocopter is an example of more complex, task-specific quadrotor control laws. This makes it an interesting experiment to verify the applicability of learning algorithms to more complex tasks. The experiment highlights the limitations of model-based control that can be overcome by learning control: the system is subject to large, repeatable disturbances caused by unmodeled dynamics (such as aerodynamic effects and the pendulum attachment point not coinciding with the quadrocopter's center of mass) and parameter uncertainties (such as the length of the pendulum). Both [P4] and [P5] present experimental results demonstrating the ability of the learning algorithms to significantly improve tracking performance of the flying inverted pendulum.

Contribution This paper details the system design of the flying inverted pendulum experiment. It contains the derivation of the combined dynamics of quadrocopter and pendulum, the derivation of equilibrium conditions and linearized dynamics for stationary and circular flight, and the design of LQR controllers. The system design is verified experimentally.

Related Publications

[R7] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter Pole Acrobatics. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

This paper extends the flying inverted pendulum experiment to a system that allows quadrocopters to balance an inverted pendulum, throw it into the air, and catch and balance it again on a second vehicle.

2.3 List of Publications

This section lists all peer-reviewed journal articles and conference contributions that were published or submitted by the author during his doctoral studies. The publications [P1]-[P6] are featured in this thesis. The related publications [R1]-[R11] present preliminary, related, or additional research results from the author's doctoral studies. Where applicable, their respective connection to the publications [P1]-[P6] is highlighted in the descriptions of the contributions of individual papers (Sections 2.1 and 2.2).

The journal manuscripts [P2], [P4], and [R11] are currently under review. Preliminary results of [P2] and [P4] have been published in [R3] and [R8], respectively.

Publications included in this thesis

(listed in the order in which they appear in this thesis)

[P1] Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control. Autonomous Robots, Volume 33, Issue 1-2, pp 69-88, 2012.

[P2] Markus Hehn and Raffaello D'Andrea. Real-Time Trajectory Generation for Quadrocopters. IEEE Transactions on Robotics (under review).

[P3] Markus Hehn and Raffaello D'Andrea. Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

[P4] Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Learning Algorithm for High-Performance, Periodic Quadrocopter Maneuvers. Mechatronics (under review).

[P5] Markus Hehn and Raffaello D'Andrea. An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories. Proceedings of the 2013 European Control Conference (ECC), 2013.

[P6] Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011.

Related publications

(listed in chronological order)

[R1] Sergei Lupashin, Angela P. Schoellig, Markus Hehn, and Raffaello D'Andrea. The Flying Machine Arena as of 2010. Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA) - Video Submission, 2011.

[R2] Angela Schoellig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasibility of Motion Primitives for Choreographed Quadrocopter Flight. Proceedings of the 2011 American Control Conference (ACC), 2011.

[R3] Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. Proceedings of the 2011 IFAC World Congress, 2011.

[R4] Stefania Tonetti, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Distributed Control of Antenna Array with Formation of UAVs. Proceedings of the 2011 IFAC World Congress, 2011.

[R5] Robin Ritz, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Performance Benchmarking Using Optimal Control. Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.

[R6] Robin Ritz, Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. Cooperative Quadrocopter Ball Throwing and Catching. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

[R7] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter Pole Acrobatics. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[R8] Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Feed-Forward Learning Scheme for High-Performance Periodic Quadrocopter Maneuvers. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[R9] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. A Computationally Efficient Algorithm for State-to-State Quadrocopter Trajectory Generation and Feasibility Verification. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[R10] Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A Platform for Aerial Robotics Research and Demonstration: The Flying Machine Arena. Mechatronics, Volume 24, Issue 1, pp 41-54, 2014.

[R11] Federico Augugliaro, Sergei Lupashin, Michael Hamer, Cason Male, Markus Hehn, Mark W. Mueller, Jan Willmann, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. The Flight Assembled Architecture Installation: Cooperative construction with flying machines. IEEE Control Systems Magazine (under review).

2.4 List of Supervised Student Projects

The following is a list of student projects that the author supervised at ETH Zurich as part of his doctoral studies.

Master's thesis

The master's thesis is a six-month, full-time project.

[M1] Luzius Brodbeck, Quadrotor Collision Avoidance, Fall 2010.

[M2] Matthias Fässler, Modeling, Control and Trajectory Tracking with a CoaX helicopter, Spring 2011

(co-supervision of external master's thesis carried out with Prof. Vijay Kumar at the University of Pennsylvania)

[M3] Robin Ritz, Cooperative Quadrocopter Ball Manipulation, Fall 2011.

(co-supervised with Mark Müller)

[M4] Dario Brescianini, Quadrocopter Pole Acrobatics, Spring 2012.

Semester project

The semester project is a semester-long, part-time project.

[S1] Mark Müller, Automatic Tuning of PID Controllers for Flight Control, Fall 2009.

(co-supervised with Guillaume Ducard)

[S2] Robin Ritz, Time-Optimal Quadrotor Control, Spring 2010.

[S3] Michael Blösch, Quadrotor Ball Launching, Spring 2010.

[S4] Dario Brescianini, Nonlinear Quadrocopter Attitude Control, Fall 2011.

[S5] Sara Spedicato, Quadrotor Attitude Control, Spring 2012.

[S6] Simon Berger, UAV Compliance Monitoring, Fall 2012.

(co-supervised with Mark Müller)

Bachelor's thesis

The bachelor's thesis is a three-month, full-time project.

[B1] Matthias Hofer, Computation of Optimal Quadrotor Maneuvers, Fall 2012.

(co-supervised with Robin Ritz)

2.5 Outreach

Throughout the duration of the author's doctoral studies, the results of the research were presented both to visitors at ETH Zurich and at various events. Some of the research resulted in stand-alone demonstrations, while other algorithms were used as components in more complex systems.

Lab Demonstrations

The algorithms developed during the author's doctoral studies have been demonstrated to more than 3000 visitors from academia, industry, governments, schools, and the general public at over 150 separate demonstrations of the Flying Machine Arena at ETH Zurich.

Exhibitions

The technology developed in the Flying Machine Arena was also exhibited at several events held outside ETH Zurich (as an example, see Figure 2.1). Further details about the requirements and execution of external exhibitions may be found in [20]. All of the following exhibitions included algorithms that are part of the research presented herein:

Flight Assembled Architecture, Orléans, France, December 2011.

Hannover Messe, Hannover, Germany, April 2012.

Google I/O After Hours, San Francisco, USA, June 2012.

Zurich Minds, Zürich, Switzerland, December 2012.

TED Global, Edinburgh, United Kingdom, June 2013.

www.FlyingMachineArena.org
Mobile Robots & Autonomous Systems
NEXT SHOW

Figure 2.1 Exhibition of the Flying Machine Arena at Hannover Messe 2012, the largest industrial fair in the world. The demonstrations presented to the audience included algorithms presented in this thesis as well as in the related publications listed in Section 2.3.

Selected Media Coverage

NZZ Format, “Die Intelligenz der Roboter”, SF1 (Swiss national television), April 2010.

IEEE Automaton Blog, “Pendulum-Balancing Quadrotor Learns Some New Tricks”, May 2011.

Daily Planet, “Robots build without human interference”, Discovery Channel, January 2012.

engine, “Those Magnificent Flying Machines”, March 2012.

c’t Magazin, “Flugzirkus: Spielerische Forschung mit Quadrokoptern”, July 2012.

Frankfurter Allgemeine Zeitung, “Perfekte Teamarbeit in der Luft”, October 2012.

Huffington Post UK, “Quadrocopter Pole Acrobatics: Robots Play Catch With Javelins”, February 2013.

Blick am Abend, “Roboter beim Stoeckli-Werfen”, February 2013.

The Art of Movement, “Flying robots perform amazing acrobatics”, CNN, November 2013.

Stephen Hawking’s Brave New World, “Hyper Connections”, Discovery Channel, to air.

3

Future Directions

This chapter discusses the current state of the research projects and potential future research directions.

3.1 Part A. Trajectory Generation

The first part of the trajectory generation research focused on the development of a benchmarking tool to determine the performance potential of quadrotor systems. The existing tool can be used to compute two-dimensional maneuvers between arbitrary states. It does, however, require computation times on the order of hours for a single maneuver, and the user must provide a reasonable initial guess for the method to converge.

Efficient computation of time-optimal trajectories: To perform statistical performance evaluations for a full range of motions, the ability to more quickly compute time-optimal trajectories is desirable. While real-time capabilities are not required, computing thousands of trajectories within useful time limits without user interaction would be a significant improvement. Preliminary results of the application of pseudospectral optimal control methods [8] show large improvement potential, however further research is required to ensure that the method always finds results that satisfy the known optimality conditions.

Benchmarking of general three-dimensional trajectories: Extending the time-optimal trajectory generator to accommodate a full three-dimensional quadrocopter model would enable the performance analysis of arbitrary maneuvers. Due to the computational complexity of the current method, this extension would likely require the use of other, more efficient ways to solve the problem.

The second part of this research focused on the design of efficient real-time trajectory generation methods. The presented algorithms have been implemented and used extensively in the Flying Machine Arena project, both in individual experiments and as building blocks in larger systems. They form a solid and proven base for further research and development.

Higher performance of computed maneuvers: The comparison to time-optimal maneuvers shows that the trajectories generated by these methods perform well, but improvement potential remains. Further research could be aimed at further closing the gap to truly time-optimal maneuvers. A potential way to achieve this would be to plan trajectories that may violate the decoupled feasibility constraints, and then verify the feasiblity of the planned trajectory after planning.

Increasing computational efficiency: The presented algorithms include the iterative computation of design parameters in order to maximize performance. While straightforward optimization methods were implemented in the current version, more sophisticated methods (see e.g. [12]) could further reduce the computational complexity of the trajectory generator.

Other classes of trajectory generation problems: The current algorithms are designed to solve trajectory generation problems where either a target point is to be reached at rest as quickly as possible, or a given point is to be intercepted at a given time. It would be interesting to extend the approach presented herein to other classes of trajectories, for example reaching a given point at a given speed, or visiting a set of points.

Augmenting trajectory-generation-based control with learning control: The trajectory generation methods presented in this thesis are based on first-principles models of quadrocopters, and ignore well-known aerodynamic effects acting on quadrocopters (such as rotor damping [26] and drag-like effects [2]). An interesting approach to account for such effects would be to apply learning methods in addition to the real-time trajectory generation. This could compensate for such unmodeled effects based on past executions of similar trajectories without significantly increasing the computational complexity of the trajectory generation algorithm.

3.2 Part B. Iterative Learning of Periodic Motions

The frequency-domain iterative learning methods presented in Papers IV and V form a solid basis for further research. The algorithms are computationally lightweight and allow the non-causal compensation of repeatable tracking errors. In our experiments, the tracking performance after successful learning was close to the limit of repeatability; however, there are still a number of potential improvements:

Choice of correction signal Fourier order: The order of the Fourier series used to represent the tracking error and controller set point corrections is currently chosen by the user. An interesting extension would be to automatically analyze the reference trajectory in order to determine (in combination with the known closed-loop system dynamics) the required order of the Fourier series.

Transfer to similar maneuvers: The focus of the learning algorithms in this research was to improve the performance of one maneuver based on previous executions of the same maneuver. A change of the maneuver would then require re-starting the learning process entirely. One would however expect that similar maneuvers require similar corrections, and that the data of past executions of one maneuver could serve as a starting point to learn a similar maneuver. For more traditional time-domain iterative learning control methods, recent results of such knowledge transfer algorithms applied to quadrocopters have been promising [13].

Transfer to higher execution speeds: The current method of initializing the learning at higher speeds from correction data of lower speeds is based on a first-order extrapolation of the past corrections and the known closed-loop system transfer function. It would be interesting to explore other mechanisms to predict the corrections at higher speeds from low-speed measurement data, perhaps by including a disturbance model to predict speed-dependent disturbance properties.

References

[1] Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Inversion Based Direct Position Control and Trajectory Following for Micro Aerial Vehicles. In *International Conference on Intelligent Robots and Systems*, 2013.

[2] Moses Bangura and Robert Mahony. Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor. In *Australasian Conference on Robotics and Automation*, 2012.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I.* Athena Scientific, third edition, 2005.

[4] Patrick Bouffard, Anil Aswani, and Claire J Tomlin. Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *International Conference on Robotics and Automation*, 2012.

[5] Y. Bouktir, M. Haddad, and T. Chettibi. Trajectory Planning for a Quadrotor Helicopter. In *Mediterranean Conference on Control and Automation*, 2008.

[6] Pierre-jean Bristeau, Philippe Martin, Erwan Salaün, and Nicolas Petit. The Role of Propeller Aerodynamics in the Model of a Quadrotor UAV. In *European Control Conference*, 2009.

[7] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A Survey of Iterative Learning Control. *Control Systems Magazine*, 26(3):96–114, 2006.

[8] Bruce A. Conway. A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems. *Journal of Optimization Theory and Applications*, 152(2):271–306, 2011.

[9] Ian D. Cowling, Oleg A. Yakimenko, and James F. Whidborne. A Prototype of an Autonomous Controller for a Quadrotor UAV. In *European Control Conference*, 2007.

[10] Carlos E. García, David M. Prett, and Manfred Morari. Model Predictive Control: Theory and Practice - A Survey. *Automatica*, 25(3):335–348, 1989.

[11] Hans Peter Geering. *Optimal Control with Engineering Applications.* Springer, 2007.

[12] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic Press, October 2003.

[13] Michael Hamer, Markus Waibel, and Raffaello D'Andrea. Knowledge Transfer for High-Performance Quadrocopter Maneuvers. In *International Conference on Intelligent Robots and Systems*, 2013.

[14] Henry Hermes and Joseph P Lasalle. Functional Analysis and Time Optimal Control. *Mathematics in Science and Engineering*, 56, 1969.

[15] Gabriel M Hoffmann, Hao Huang, Steven L Waslander, and Claire J Tomlin. Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In *AIAA Guidance, Navigation and Control Conference*, 2007.

[16] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor Helicopter Trajectory Tracking Control. In *Conference on Decision and Control*, 2008.

[17] Jonathan P. How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.

[18] Qimi Jiang, Daniel Mellinger, Christine Kappeyne, and Vijay Kumar. Analysis and Synthesis of Multi-Rotor Aerial Vehicles. In *International Design Engineering Technical Conference*, 2011.

[19] Sergei Lupashin and Raffaello D'Andrea. Adaptive Fast Open-Loop Maneuvers for Quadrocopters. *Autonomous Robots*, 33(1-2):89–102, 2012.

[20] Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A Platform for Aerial Robotics Research and Demonstration: The Flying Machine Arena. *Mechatronics*, 24(1):41–54, 2014.

[21] Daniel Mellinger and Vijay Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *International Conference on Robotics and Automation*, 2011.

[22] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *International Symposium on Experimental Robotics*, 2010.

[23] Fabian L. Mueller, Angela P. Schoellig, and Raffaello D'Andrea. Iterative Learning of Feed-Forward Corrections for High-Performance Tracking. In *International Conference on Intelligent Robots and Systems*, 2012.

[24] Mark W. Mueller and Raffaello D'Andrea. A Model Predictive Controller for Quadrocopter State Interception. In *European Control Conference*, 2013.

[25] Pong-in Pipatpaibul and P R Ouyang. Application of Online Iterative Learning Tracking Control for Quadrotor UAVs. *ISRN Robotics*, 2013:Article ID 476153, 2013.

[26] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot. In *Australasian Conference on Robotics and Automation*, 2006.

[27] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, July 2010.

[28] Caitlin Powers, Daniel Mellinger, Aleksandr Kushleyev, and Bruce Kothmann. Influence of Aerodynamics and Proximity Effects in Quadrotor Flight. In *International Symposium on Experimental Robotics*, 2012.

[29] Oliver Purwin and Raffaello D'Andrea. Performing and Extending Aggressive Maneuvers Using Iterative Learning Control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[30] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial Trajectory Planning for Quadrotor Flight. In *International Conference on Robotics and Automation*, 2013.

[31] Angela P Schoellig, Fabian L Mueller, and Raffaello D'Andrea. Optimization-Based Iterative Learning for Precise Quadrocopter Trajectory Tracking. *Autonomous Robots*, 33(1-2):103–127, 2012.

[32] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Autonomous Mobile Robots*. The MIT Press, second edition, 2011.

Part A

TRAJECTORY GENERATION

Paper I

Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control

Markus Hehn, Robin Ritz, and Raffaello D'Andrea

Abstract

Frequently hailed for their dynamical capabilities, quadrotor vehicles are often employed as experimental platforms. However, questions surrounding achievable performance, influence of design parameters, and performance assessment of control strategies have remained largely unanswered. This paper presents an algorithm that allows the computation of quadrotor maneuvers that satisfy Pontryagin's minimum principle with respect to time-optimality. Such maneuvers provide a useful lower bound on the duration of maneuvers, which can be used to assess performance of controllers and vehicle design parameters. Computations are based on a two-dimensional first-principles quadrotor model. The minimum principle is applied to this model to find that time-optimal trajectories are bang-bang in the thrust command, and bang-singular in the rotational rate control. This paper presents a procedure allowing the computation of time-optimal maneuvers for arbitrary initial and final states by solving the boundary value problem induced by the minimum principle. The usage of the computed maneuvers as a benchmark is demonstrated by evaluating quadrotor design parameters, and a linear feedback control law as an example of a control strategy. Computed maneuvers are verified experimentally by applying them to quadrocopters in the ETH Zurich Flying Machine Arena testbed.

Published in *Autonomous Robots* Volume 33, Issue 1-2. The publication includes two electronic attachments available at `www.hehn.be` and `www.springerlink.com`.
DOI: 10.1007/s10514-012-9282-3

©2012 Springer Science+Business Media, LLC. The final publication is available at `springerlink.com`.

1. Introduction

Quadrotor vehicles are an increasingly popular aerial vehicle platform. Advantages compared to other micro unmanned aerial vehicle (UAV) platforms are the ability to hover, robustness and straightforwardness of design due to mechanical simplicity [20], and safety benefits due to smaller rotor sizes compared to most other hover-capable UAVs [10].

Most early research on quadrocopter control focused on near-hover operation. This operation was commonly achieved using linear controllers, which were tuned through various strategies (see, for example, [3], and references therein). Trajectories are typically generated by connecting waypoints by lines or circles, and choosing flight velocity sufficiently low to satisfy near-hover assumptions (e.g. [12]).

With the above advantages making quadrocopters attractive platforms for numerous research fields, another key strength of these vehicles is increasingly being exploited: They offer exceptional agility in the rotational degrees of freedom due to the off-center mounting of the propellers, which allows the generation of large torques. Most platforms also provide high thrust-to-weight ratios, allowing fast translational dynamics. This has lead to numerous more complex control strategies that leverage these capabilities by explicitly considering the feasibility of trajectories. Examples include fast translations [9, 18, 21], flips [17], and dancing motions [23]. Several authors have also introduced algorithms that generate trajectories from a class of motion primitives (lines, polynomials, or splines) and respect the dynamic constraints of quadrocopters [4, 5, 11]. These algorithms enforce dynamic feasibility by designing trajectories in a two-step process, first determining the shape of the trajectory and subsequently determining an appropriate speed profile such that feasibility constraints are not violated.

While this work has lead to impressive results, it has not answered one key question: Given the specifications of a quadrotor vehicle, how fast can a certain maneuver be completed? More formally, this can be stated as: What is the time-optimal trajectory for given initial and final states? The ability to compute accurate answers to this question offers insight that may be helpful for many performance-related topics: How much of the theoretical speed potential does a certain control strategy utilize? Are there specific maneuvers for which there exists a large improvement potential for a given controller? How do the physical parameters of the vehicle, such as mass or maximum thrust, influence performance?

The computation of time-optimal trajectories provides a useful tool to answer these questions: The duration of such trajectories provides an absolute lower bound, which may be used for comparisons. Several approaches have been proposed to compute time-optimal trajectories: [14] proposed a direct method using control inputs discretized in time, and solved the resulting optimization problem using genetic algorithms and non-linear programming. [4] presented a different direct method, exploiting the differential flatness of the system dynamics and parameterizing the output trajectory by a set of control points, which are then connected by B-spline functions. While both methods have been successfully demonstrated, the optimality of solutions remained unanswered.

In this paper, we present an algorithm that allows the computation of state and input trajectories between two states. Pontryagin's minimum principle is employed to determine the structure of time-optimal trajectories, and computed trajectories can be verified to satisfy the minimum principle conditions. While these conditions are necessary - but not sufficient - for optimality, they do provide a strong argument for the found trajectories.

The computation of maneuvers that satisfy the minimum principle using the algorithm presented herein requires considerable computational effort. This makes it an unattractive proposition in real-time scenarios. However, the off-line computation of a selection of maneuvers, chosen to be representative for the motion of the vehicle in a specific application, provides a lower bound on the maneuver duration, to which real-time capable controllers may be compared as a means of benchmarking their performance. If certain motions are expected to be carried out repeatedly, this method may be used to design nominal trajectories, for which stabilizing controllers could then be designed.

We base our calculations on a two-dimensional first-principles model of the quadrotor vehicle. Because the algorithm presented herein is meant as a benchmarking tool rather than a control algorithm, the choice of a two-dimensional model provides a reasonable trade-off: The omission of the third dimension makes the problem tractable by reducing its dimensionality. At the same time, two-dimensional problems cover a large class of maneuvers that are interesting to benchmark. Through rotations of the coordinate system, all maneuvers that start and end at rest may be treated as two-dimensional problems, as well as maneuvers where initial and final velocities and rotations are aligned with the rotated coordinate system. This provides a large number of control scenarios that may be benchmarked, such that the benchmarking results can be considered representative of the performance.

The quadrotor model is presented in Section 2. The structure of time-optimal trajectories is determined in Section 3, and the algorithm used to solve the induced boundary value problem is presented in Section 4. Numerous sample maneuvers have been computed with this algorithm, a selection of which are shown in Section 5. The application of time-optimal maneuvers as a benchmarking tool is demonstrated by computing the influence of model parameters and evaluating the performance of a linear controller (Section 6). The computed trajectories are validated in actual flight tests, with the experimental results (Section 7) demonstrating their validity.

2. Modeling of Vehicle Dynamics

This section introduces the two-dimensional first-principles model, on which we base our calculations. Furthermore, a non-dimensionalizing coordinate transformation is employed to reduce the number of model parameters to two.

2.1 First-Principles Model

The two-dimensional model has three degrees of freedom: the horizontal position x, the vertical position z, and the pitch angle θ, as shown in Figure 1.

The quadrocopter is controlled by two inputs: the total thrust force F_T and the pitch rate ω, shown in Figure 1. The control inputs are subject to saturation. In particular, the pitch rate is limited to a maximum allowable magnitude $\overline{\omega}$, and the thrust is constrained to be between $\underline{F_\mathrm{T}}$ and $\overline{F_\mathrm{T}}$:

$$|\omega| \leq \overline{\omega}, \tag{1}$$

$$\underline{F_\mathrm{T}} \leq F_\mathrm{T} \leq \overline{F_\mathrm{T}} \tag{2}$$

The pitch rate is limited by the range of the gyroscopic on-board sensors, and the maximum collective thrust is determined by the maximum thrust each propeller can produce. Because commonly available motor drivers do not allow changes of the direction of rotation mid-flight, and because the propellers are of fixed-pitch type, it is assumed that the collective thrust is always positive: $\underline{F_T} > 0$.

The equations of motion are

$$\ddot{x} = \frac{F_\mathrm{T}}{m} \sin\theta, \tag{3}$$

$$\ddot{z} = \frac{F_\mathrm{T}}{m} \cos\theta - \mathrm{g}, \tag{4}$$

$$\dot{\theta} = \omega, \tag{5}$$

where g denotes the gravitational acceleration and m is the mass of the quadrocopter.

We assume that the angular velocity $\dot{\theta}$ can be set directly without dynamics and delay. This is motivated by the very high angular accelerations that quadrotors can reach (typically on the order of several hundred $\mathrm{rad\,s^{-2}}$), while the angular velocity is usually limited by the gyroscopic sensors used for feedback control on the vehicle [17]. This means that aggressive maneuvering is often more severely limited by the achievable rotational rate than the available angular acceleration. The simplification of the model to a constrained rotational rate makes the derivations and numerical computations in this paper tractable.

It should be noted that the first-principles model presented above not only assumes there is direct control of the rotational rate, but also neglects numerous dynamic effects, such as propeller speed change dynamics [17], blade flapping [20], changes in the angle of attack of the propellers [13], and drag forces. Considering that we seek to compute very fast maneuvers, it is to be expected that such effects are significant. However, the purpose of the method presented herein is to allow the benchmarking of quadrotor performance. Because such benchmarks typically serve comparative purposes, modeling inaccuracies play a less significant role than in other applications: As long as the model captures

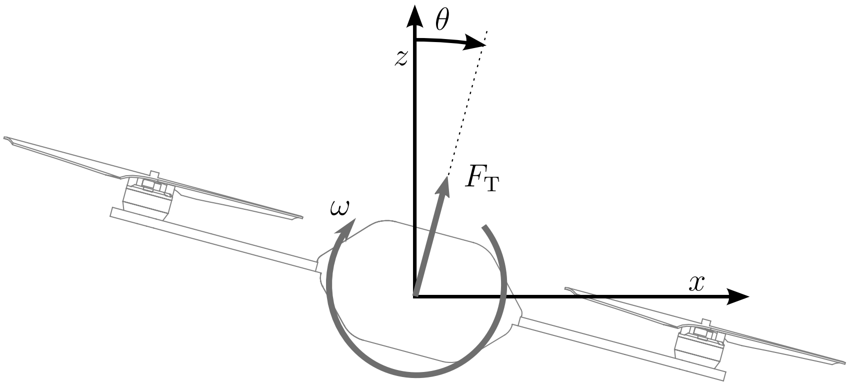

Figure 1. Coordinate system and control inputs of the quadrotor model.

the dominant dynamics, one can expect relative comparisons based on the model to provide meaningful answers. Differences of results based on different models, for example between simulative and experimental results, highlight that the model does not capture all effects; however, when comparing results obtained using the same dynamics, one can expect the neglected effects to, for the most part, cancel out. We therefore believe the trade-off between modeling complexity (and therefore tractability of the optimal control problem) and accuracy to be reasonable. This is further supported by the comparison of numerical and experimental results, which show a reasonable qualitative match with some quantitative discrepancies between simulations and reality (see Section 7).

2.2 Non-Dimensional Model

In order to allow simple comparisons between different configurations, it is beneficial to describe the quadrotor model with as few parameters as possible. We therefore introduce a non-dimensionalizing transformation:

$$\hat{t} := \overline{\omega} t, \tag{6}$$

$$\hat{x} := \frac{\overline{\omega}^2 x}{\mathrm{g}}, \tag{7}$$

$$\hat{z} := \frac{\overline{\omega}^2 z}{\mathrm{g}}. \tag{8}$$

In the transformed coordinates and time, the gravitational acceleration is unity. Defining the state vector

$$\mathbf{x} := (\hat{x}, \dot{\hat{x}}, \hat{z}, \dot{\hat{z}}, \theta), \tag{9}$$

and the transformed control input vector

$$\mathbf{u} = (u_R, u_T) := \left(\frac{\omega}{\overline{\omega}}, \frac{F_\mathrm{T}}{m\mathrm{g}}\right), \tag{10}$$

the quadrotor dynamics may be written as

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\hat{x}} \\ \ddot{\hat{x}} \\ \dot{\hat{z}} \\ \ddot{\hat{z}} \\ \dot{\theta} \end{pmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \dot{\hat{x}} \\ u_T \sin\theta \\ \dot{\hat{z}} \\ u_T \cos\theta - 1 \\ u_R \end{pmatrix}. \tag{11}$$

It is important to note that in the above equations, the derivative $\dot{\mathbf{x}}$ is taken with respect to the transformed time $\hat{t}$.

The transformed control inputs are bounded by the minimum and maximum thrust in units of gravitational acceleration, and unit allowable rotational rate:

$$\begin{gathered} \underline{u_T} = \frac{\underline{F_\mathrm{T}}}{m\mathrm{g}} \leq u_T \leq \overline{u_T} = \frac{\overline{F_\mathrm{T}}}{m\mathrm{g}}, \\ |u_R| \leq 1. \end{gathered} \tag{12}$$

The dimensionless model contains two model parameters: The lower and upper limit of the collective thrust input ($\underline{u_T}$ and $\overline{u_T}$).

For the remainder of the paper, the hat notation will be omitted. It is understood that calculations are carried out using the dimensionless coordinates, but could equivalently be done in the original system.

3. Minimum Principle for Time-Optimal Maneuvers

This section demonstrates how Pontryagin's minimum principle for time-optimality is applied to the vehicle dynamics. It is shown that the thrust input is bang-bang and that the rotational control is bang-singular, meaning that the control input is always at full positive or negative saturation, except during singular arcs. The system is augmented by the switching function of the rotational rate input, leading to a boundary value problem containing five unknowns. The existence of optimal trajectories is shown.

The time-optimal quadrocopter trajectory between the initial state $\mathbf{x}_0$ and the final state $\mathbf{x}_T$ is characterized by its state trajectory $\mathbf{x}^*(t), t \in [0, T]$, or equivalently by the corresponding control inputs $\mathbf{u}^*(t), t \in [0, T]$. It is the solution to the optimization problem

$$\begin{aligned} \underset{\mathbf{u}\in\mathbf{U}}{\text{minimize}} \quad & T \\ \text{subject to} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \\ & \mathbf{x}(0) = \mathbf{x}_0, \\ & \mathbf{x}(T) = \mathbf{x}_T, \end{aligned} \tag{13}$$

where $\mathbf{U}$ denotes the set of all attainable control vectors, as defined by Equation (12).

Minima to this problem may be found using Pontryagin's minimum principle, which provides necessary conditions for optimality [2, 7]. With a cost function equal to 1, i.e., $g(\mathbf{x}, \mathbf{u}) = 1$, the Hamiltonian for the problem yields

$$\begin{aligned} H(\mathbf{x}, \mathbf{u}, \mathbf{p}) &= g(\mathbf{x}, \mathbf{u}) + \mathbf{p}^T f(\mathbf{x}, \mathbf{u}) \\ &= 1 + p_1\dot{x} + p_2 u_T \sin\theta \\ &\quad + p_3\dot{z} + p_4(u_T\cos\theta - 1) + p_5 u_R, \end{aligned} \tag{14}$$

where p_i denotes the elements of the costate vector $\mathbf{p}$. Note that, because the terminal time of the maneuver is free, the Hamiltonian is always zero along an optimal trajectory [2]:

$$H(\mathbf{x}^*, \mathbf{u}^*, \mathbf{p}) \equiv 0. \tag{15}$$

Applying the adjoint equation to the Hamiltonian

$$\dot{\mathbf{p}} = -\nabla_{\mathbf{x}} H(\mathbf{x}^*, \mathbf{u}^*, \mathbf{p}), \tag{16}$$

the first four costates may be expressed explicitly as

$$\begin{aligned} \dot{p}_1 &= 0, & &\Rightarrow & p_1 &= c_1, \\ \dot{p}_2 &= -p_1, & &\Rightarrow & p_2 &= c_2 - c_1 t, \\ \dot{p}_3 &= 0, & &\Rightarrow & p_3 &= c_3, \\ \dot{p}_4 &= -p_3, & &\Rightarrow & p_4 &= c_4 - c_3 t, \end{aligned} \tag{17}$$

where the constants $\mathbf{c} = (c_1, c_2, c_3, c_4)$ remain to be determined.

The last element of the costate vector, p_5, is given by the adjoint equation (16) to be

$$\dot{p}_5 = -p_2 u_T^* \cos\theta^* + p_4 u_T^* \sin\theta^*. \tag{18}$$

The above equation depends on the control input u_T, the trajectory of which is not known a priori. It is therefore not easily possible to express p_5 explicitly.

3.1 Optimal Control Inputs

The minimum principle states that the optimal control input trajectory minimizes the Hamiltonian (14) over all possible values of $\mathbf{u}$. Since the two control inputs do not appear in the same summand in the Hamiltonian, it can be minimized separately for u_R and u_T:

Optimal Control Input u_R^* For the rotational control input u_R, minimizing (14) results in

$$u_R^* = \underset{u_R \in [-1,+1]}{\operatorname{argmin}} \{p_5 u_R\}. \tag{19}$$

If p_5 changes sign, then u_R^* switches from -1 to $+1$ or vice versa. We define

$$\Phi_R := p_5 \tag{20}$$

as the switching function of u_R^*. If Φ_R is zero for a nontrivial interval of time, then the minimum condition (19) is insufficient to determine u_R^*. In these intervals, which are called singular arcs, u_R^* is determined using the condition that Φ_R remains zero: It follows that $\dot{\Phi}_R$ vanishes, which results in the condition

$$\dot{\Phi}_R = -p_2 u_T^* \cos\theta^* + p_4 u_T^* \sin\theta^* = 0. \tag{21}$$

Solving for θ^* using $u_T^* > 0$ (as discussed in Section 2.1) and the costate equations (17) yields

$$\theta^* = \arctan\left(\frac{p_2}{p_4}\right) = \arctan\left(\frac{c_2 - c_1 t}{c_4 - c_3 t}\right). \tag{22}$$

Differentiating (22) with respect to time gives the trajectory of the control input $u_R^* = \dot{\theta}^*$ in a singular arc:

$$u_{R,sing}^* = \frac{c_2 c_3 - c_1 c_4}{(c_1^2 + c_3^2)t^2 - 2(c_1 c_2 + c_3 c_4)t + c_2^2 + c_4^2} \tag{23}$$

The rotational control input u_R^* of a time-optimal maneuver of the quadrocopter can be written as:

$$u_R^* = \begin{cases} +1 & \text{if } \Phi_R < 0 \\ u_{R,sing}^* & \text{if } \Phi_R = 0 \\ -1 & \text{if } \Phi_R > 0 \end{cases}. \tag{24}$$

This type of optimal control trajectory is referred to as bang-bang singular control or bang-singular control [2, 15].

Optimal Control Input u_T^* To compute the optimal control trajectory for the thrust input u_T^*, the sum of all terms of the Hamiltonian containing u_T must be minimized:

$$u_T^* = \underset{u_T \in [\underline{u_T}, \overline{u_T}]}{\operatorname{argmin}} \{p_2 u_T \sin\theta^* + p_4 u_T \cos\theta^*\}. \tag{25}$$

Again, we define a switching function

$$\Phi_T := p_2 \sin\theta^* + p_4 \cos\theta^*. \tag{26}$$

For a singular arc to exist, Φ_T must be zero for a nontrivial interval of time. Setting Φ_T to zero and solving for θ^* yields

$$\theta^* = \arctan\left(-\frac{p_4}{p_2}\right) = \arctan\left(\frac{c_3 t - c_4}{c_2 - c_1 t}\right). \tag{27}$$

The pitch angle θ^* is determined by the rotational control input u_R^*. If u_R^* is in a regular interval, then the pitch angle is an affine function of time, and Equation (27) can not be satisfied over a nontrivial time interval. It can therefore be concluded that u_T^* cannot be in a singular arc when u_R^* is regular. If u_R^* is singular, θ^* is given by (22). It follows that, for a singular arc of u_T^* to exist, the pitch angle trajectory defined by (22) and by (27) must be identical:

$$\arctan\left(\frac{c_2 - c_1 t}{c_4 - c_3 t}\right) = \arctan\left(\frac{c_3 t - c_4}{c_2 - c_1 t}\right). \tag{28}$$

Taking the tangent of both sides and multiplying the constraint out yields

$$(c_1^2 + c_3^2)t^2 - 2(c_1 c_2 + c_3 c_4)t + c_2^2 + c_4^2 = 0. \tag{29}$$

Neglecting the trivial case $\mathbf{c} = (0,0,0,0)$, condition (29) cannot hold for a nontrivial interval of time. The thrust control input u_T^* therefore does not contain singular arcs and can be written as

$$u_T^* = \begin{cases} \overline{u_T} & \text{if } \Phi_T \le 0 \\ \underline{u_T} & \text{if } \Phi_T > 0 \end{cases}. \tag{30}$$

3.2 Augmented System

Because only the derivative of the switching function Φ_R is given, we augment the system equations (11) with an additional state, representing the switching function Φ_R. We define

$\mathbf{x_a} = (\mathbf{x}^*, \Phi_R)$, resulting in the augmented system dynamics

$$\dot{\mathbf{x}}_\mathbf{a} = f_a(t, \mathbf{x_a}) = \begin{pmatrix} \dot{x}^* \\ u_T^* \sin\theta^* \\ \dot{z}^* \\ u_T^* \cos\theta^* - 1 \\ u_R^* \\ (c_1 t - c_2) u_T^* \cos\theta^* + (c_4 - c_3 t) u_T^* \sin\theta^* \end{pmatrix}, \tag{31}$$

where the control inputs u_R^* and u_T^* are given by the control laws (24) and (30). A quadrotor maneuver from $\mathbf{x}_0$ to $\mathbf{x}_T$ that satisfies the minimum principle solves the boundary value problem (BVP)

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{a} &= f_a(t, \mathbf{x_a}), \\ \mathbf{x}^*(0) &= \mathbf{x}_0, \\ \mathbf{x}^*(T) &= \mathbf{x}_T. \end{aligned} \tag{32}$$

The BVP (32) contains six unknowns: The final time T, the four unknown constants $\mathbf{c}$, and the initial value of the switching function $\Phi_R(t = 0)$. However, the initial value of the switching function may not be chosen freely, but must satisfy Equation (15)[1].

3.3 Existence of Optimal Trajectories

While Pontryagin's minimum principle provides necessary conditions for optimality, it is useful to verify the existence of optimal trajectories. We apply Roxin's theorem [22] in order to show this.

We note that all assumptions on the system dynamics of Roxin's theorem hold, guaranteeing the existence and uniqueness of a solution to the system dynamics and the convexity of the system differential equation (11) with respect to the control inputs $\mathbf{u}$ (because the control inputs appear linearly, this is straightforward to see). All that remains is to show the reachability of the target state from the initial state.

We show that there is always a trajectory between two arbitrary states. We note that it is sufficient to find a trajectory from an arbitrary state to the origin. A possible maneuver between two states is then the motion from the initial state to the origin, and the reverse of the motion from the final state to the origin. A strategy to drive the system to the origin can easily be found, for example by successively applying the following steps:

1. Apply $\mathbf{u} = (\overline{u_R}, \overline{u_T})$ or $\mathbf{u} = (\underline{u_R}, \overline{u_T})$ until the pitch angle θ is zero. Thrust commands now influence only the vertical dynamics.

---

[1] It can be seen from Equation (15) that, depending on the initial state $\mathbf{x}_0$ and the constants $\mathbf{c}$, either $\Phi_R(t = 0) = 0$ is the only allowable initial value of the switching function ($u_R^*(t = 0)$ is then in a singular arc), or there are two allowable values that only differ in sign ($u_R^*(t = 0)$ is then in a regular arc, and its sign is dictated by the sign of $\Phi_R(t = 0)$).

2. Use the commands $\mathbf{u} = (0, \overline{u_T})$ and $\mathbf{u} = (0, \underline{u_T})$ to drive the states z and $\dot{z}$ to zero using a bang-bang strategy.

3. Apply rotational rate commands to drive the horizontal degree of freedom to zero, while adjusting the thrust such that the vertical degree of freedom remains at rest. The corresponding control inputs will be $\mathbf{u} = (\overline{u_R}, \mathrm{g}/\cos\theta)$, $\mathbf{u} = (\underline{u_R}, \mathrm{g}/\cos\theta)$, and $\mathbf{u} = (0, \mathrm{g}/\cos\theta)$. The allowable pitch angle during this step is limited by the available thrust, as $\mathrm{g}/\cos\theta \leq \overline{u_T}$ must hold.

As shown in [22], the existence of the minimum follows from the existence of an arbitrary trajectory to the target state.

4. Algorithm for Calculation of Time-Optimal Maneuvers

This section introduces a numerical algorithm that solves the boundary value problem for the augmented system (BVP (32)) between arbitrary initial and final states. The resulting maneuvers satisfy the minimum principle with respect to time-optimality. An implementation of the algorithm in MATLAB for free use is available on the first author's website, and is submitted along with this article. This section aims to provide the reader with a high-level overview of the algorithm used. A more detailed discussion of the individual steps may be found in Appendix A.

Finding a solution to the boundary value problem (32) is generally difficult: The state at the end of the maneuver is a non-convex function of the six unknowns, with many local minima and strongly varying sensitivities. With common boundary value problem solvers providing only local convergence under these conditions, it is necessary to provide a good initial guess for the unknowns or the solution trajectory $\mathbf{x_a}$. However, with no straightforward physical interpretation of the constant vector $\mathbf{c}$ and the switching function Φ_R, it is difficult to provide such an initial guess. The application of BVP solvers showed that convergence to the correct solution could only be achieved from initial guesses very close to the correct values, making it almost impossible to initialize the algorithm correctly. The problem is further aggravated by the fact that the numerical integration is highly sensitive to numerical errors when entering or leaving singular arcs, as will be discussed in Section 4.4.

The algorithm presented herein relaxes these problems by using more robust optimization methods to produce a good initial guess for the BVP solver. For this, we exploit the known bang-singular structure of maneuvers and parametrize a maneuver by the times at which the control inputs switch, and by the terminal time. This approach, commonly referred to as switching time optimization (STO, [24]), provides a significant advantage of requiring no initial guess of $\mathbf{x_a}$ or $\mathbf{c}$. Instead of requiring an initial guess of $\mathbf{x_a}$ or $\mathbf{c}$, it requires a guess of the switching times, which are easier to obtain, and which can lead to convergence from a much larger range of initial guesses.

The finding of a solution to the STO problem does not necessarily imply a solution to the conditions derived from the minimum principle. Therefore, the STO is used only as a first step of the algorithm. In a second step, the result of it is used to extract an initial guess of $\mathbf{c}$, which typically lies close enough to the correct values to allow a BVP solver to solve the boundary value problem as a third step, and therefore compute a maneuver that satisfies the minimum principle.

If it is assumed that the maneuver has no singular arcs, then the algorithm is less complex and more intuitive. Therefore, an algorithm assuming maneuvers with pure bang-bang behavior is introduced first, and then the modifications necessary for the computation of maneuvers with singular arcs are presented.

4.1 Switching Time Optimization

Under the assumption that the optimal solution is a pure bang-bang maneuver, the entire maneuver can be characterized by the initial values of the two control inputs, the times at which they switch, and the total duration of the maneuver. The switching time optimization algorithm optimizes over the switching times and maneuver duration, using the weighted sum of square state errors at the end of the maneuver as an objective function.

4.2 Parameter Extraction

The result of the switching time optimization is a bang-bang maneuver between the initial and the final state. It is necessary to verify that this maneuver does indeed satisfy the conditions of the minimum principle, as they were derived in Section 3. To do this, the constants $\mathbf{c} = (c_1, c_2, c_3, c_4)$ are computed based on the result of the STO, and then used as a starting point for a BVP solver.

To compute the constants, a set of equations is obtained from the known switching times: If the maneuver is to satisfy the minimum principle, the switching functions Φ_R and Φ_T must be zero when a switching time in the control inputs u_T and u_R occurs, respectively.

The thrust switching function Φ_T is known explicitly (Equation (26)), and it is straightforward to generate constraints from it. The trajectory of Φ_R, on the other hand, is not known explicitly (as shown in Section 3, only the derivative $\dot{\Phi}_R$ of the switching function is known a priori). However, once the state trajectories are known from the STO, the condition $H \equiv 0$ (which must hold if the maneuver is time-optimal) can be used to compute Φ_R.

Additional constraints are obtained from the fact that the switching function trajectory Φ_R must be a solution to the differential equation (18), which can be evaluated by numerically integrating the differential equation over time intervals, for example between two switching times.

As shown in Appendix A, all of the resulting constraints are linear in the constants $\mathbf{c}$. The resulting overconstrained system of equations can then be solved for the least-squares solution. If this solution does not satisfy all constraints to an acceptable accuracy, the found maneuver does not satisfy the minimum principle. Reasons for this may be an inappropriate choice of the number of switches in the control inputs, failure of the STO to converge to the correct maneuver, or the existence of singular arcs in the optimal maneuver.

4.3 BVP Solver

The constants extracted from the solution of the STO and the corresponding maneuver duration are used as initial guesses in a final step. Using the weighted sum of square final state errors as an objective function, the constants and the maneuver duration are optimized. In this step, the control inputs are determined from the control laws defined by the minimum principle (Equations (24) and (30)), which ensures that the maneuver indeed satisfies the minimum principle. Because the initial guess is typically very close to the optimal values, the BVP solver converges quickly in most cases.

4.4 Modified Algorithm for Bang-Singular Maneuvers

For maneuvers containing singular arcs, the switching time optimization is modified. In addition to the times at which the control inputs switch, the durations for which the rotational control input remains in a singular arc after each switching time are also introduced as optimization variables. It is no longer possible to optimize the switching times without knowledge of the constants $\mathbf{c}$, as they define the control input trajectory in singular arcs. The STO algorithm therefore optimizes over the switching times, the singular arc durations, and the values of the constants $\mathbf{c}$. While this overconstrains the optimization problem (the constants $\mathbf{c}$ as well as the switching times and singular arc durations define the times at which control inputs switch), the optimization was seen to be significantly more robust when using this parametrization.

Assuming that the thrust input u_T does not switch at the edges of the singular intervals[2], $\dot{\Phi}_R$ is continuous over the border of the singular arcs, as can be seen from (21). Consequently, the switching function Φ_R enters and leaves a singular arc tangentially. This makes the maneuver highly sensitive to numerical integration errors, and makes it difficult to determine the singular arc entry and exit points from the numerically integrated switching function. The switching times and singular arc durations are therefore retained as optimization variables in the BVP solver step. This makes it necessary to verify the match between the switching function and these optimization variables after convergence of the BVP solver.

---

[2] We conjecture that the assumption that u_T does not switch at the edges of the singular arcs is valid for almost all initial and final conditions, with an appropriately defined measure. For all maneuvers considered here, results have shown that this condition has been fulfilled.

5. Numerical Results

In this section, we present a selection of quadrotor maneuvers that were computed using the algorithm introduced in Section 4. While the algorithm allows the computation of motions between arbitrary states, the results presented herein focus on position changes where the quadrocopter is at rest at the beginning and at the end of the maneuver.

The quadrotor parameters for which the maneuvers have been computed are based on the ETH Zurich Flying Machine Arena vehicles, as described in [17]. Table 1 shows the used numerical model parameters in the dimensional form $(\underline{F_\mathrm{T}}/m, \overline{F_\mathrm{T}}/m, \overline{\omega})$. The non-dimensional parameters $(\underline{u_T}, \overline{u_T})$ can easily be calculated from these using the control input transformation (12). While computations were carried out in the dimensionless coordinate system, the maneuvers presented herein have been transformed back to the state variables representing physical dimensions, allowing a more intuitive interpretation.

5.1 Vertical Displacements

First, the special case of maneuvers with a purely vertical displacement is considered. At the beginning of the maneuver, the quadrotor is at rest and at a pitch angle of zero, and without loss of generality, the initial position of the quadrotor can be set to the origin:

$$\mathbf{x}_0 = (x(0), \dot{x}(0), z(0), \dot{z}(0), \theta(0)) = (0, 0, 0, 0, 0). \tag{33}$$

At the end of the vertical displacement maneuver, the quadrotor is at rest again, with no overall horizontal displacement, and a final pitch angle that is a multiple of 2π:

$$\mathbf{x}_T = (x(T), \dot{x}(T), z(T), \dot{z}(T), \theta(T)) = (0, 0, z_T, 0, N2\pi). \tag{34}$$

To determine time-optimal maneuvers, it is necessary to compare maneuvers for different values of full rotations N, as the equivalence of corresponding terminal states is not included in the maneuver description. The terminal times of these different maneuvers are then compared to determine the fastest one.

We limit the results shown herein to maneuvers with a positive z_T value. Maneuvers have been computed for vertical displacements z_T of 0.1 m to 10 m, with a step size of 0.1 m. In Figure 2, the maneuver duration T is plotted as a function of the vertical

Parameter	Value	Description
$\underline{F_\mathrm{T}}/m$	$1\,\mathrm{m\,s^{-2}}$	Minimum mass-normalized thrust
$\overline{F_\mathrm{T}}/m$	$20\,\mathrm{m\,s^{-2}}$	Maximum mass-normalized thrust
$\overline{\omega}$	$10\,\mathrm{rad\,s^{-1}}$	Maximum rotational rate

Table 1. Numerical parameters of the quadrotor model.

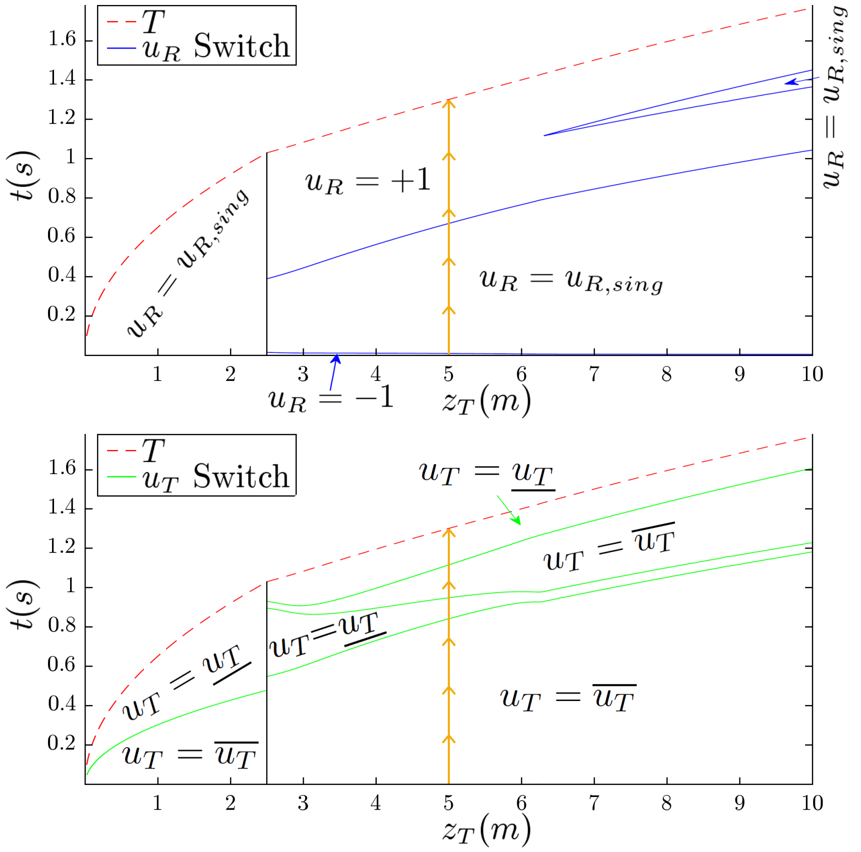

Figure 2. Maneuver duration T as function of the final vertical displacement z_T for rest-to-rest maneuvers with no horizontal displacement. Additionally, the switching times of u_R are drawn in the plot on the top, and the switching times of u_T in the plot on the bottom. The vertical solid line denotes where the structure of the minimum principle solution changes: on the left the solution without a flip, which is faster for small z_T, and on the right the solution where the quadrocopter performs a flip, which is faster for large z_T. The line with the arrows denotes the example maneuver of Figure 4.

displacement z_T. Furthermore, it shows the switching times for each maneuver. For a particular displacement z_T, the maneuver starts at the bottom of the graph ($t = 0\,\text{s}$) and, as time passes, moves up in the positive direction of the t-axis. Every time a switching line is crossed, the corresponding control input switches to the value specified in the diagram. The maneuver is finished when the T-curve is reached.

If the desired vertical displacement is small, i.e. for $z_T \leq 2.4\,\text{m}$, the quadrocopter is within a singular arc during the entire maneuver. The pitch angle remains at exactly $\theta = 0$. The thrust is at its maximum at the beginning and switches to its minimum at a time such that the quadrocopter comes to rest due to gravity at the desired height z_T. For $z_T \geq 2.5\,\text{m}$, it is beneficial to perform a flip and to make use of the thrust for braking while the pitch is around $\theta \approx \pm\pi$. For $z_T \geq 6.3\,\text{m}$, a singular arc (which keeps the pitch near $\theta \approx \pm\pi$ for a particular time) appears, as can be seen in Figure 2. Thus, the flip is stopped for an interval of deceleration. A selection of maneuvers is depicted in Figure 3, showing the different maneuver shapes for varying

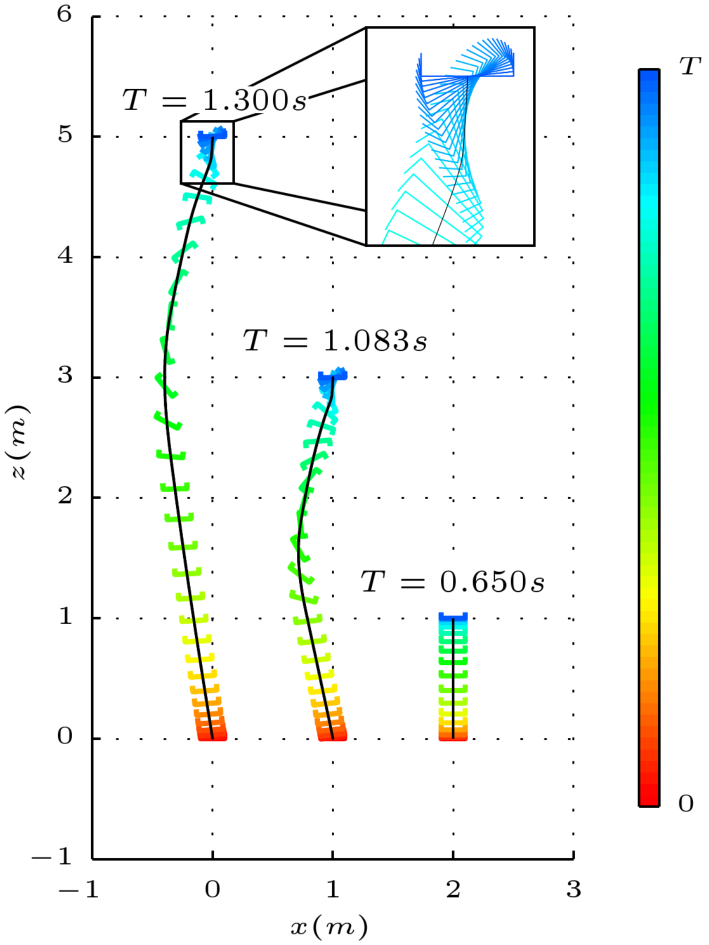

Figure 3. Illustration of maneuvers for a purely vertical displacement of 1 m, 3 m, and 5 m. The maneuvers satisfy the minimum principle and for each maneuver, a quadrotor is plotted every 0.04 s or every 0.01 s in the zoom box, respectively.

displacements.

The arrow line in the plots of Figure 2 denotes an example maneuver with a vertical displacement of $z_T = 5\,\mathrm{m}$. The state, input and switching function trajectories of this maneuver are shown in Figure 4. The switches in the control input trajectories in Figure 4 can be depicted by following the arrow line from $t = 0\,\mathrm{s}$ towards $t = T$ in Figure 2.

5.2 Horizontal Displacements

We now consider maneuvers that lead to a purely horizontal displacement. For this case, the initial and final state are

$$\mathbf{x}_0 = (0, 0, 0, 0, 0), \quad \mathbf{x}_T = (x_T, 0, 0, 0, 0). \tag{35}$$

Considerations of symmetry lead to the conclusion that, for purely horizontal displacements, the switching times are symmetric about the half-time of the maneuver. This

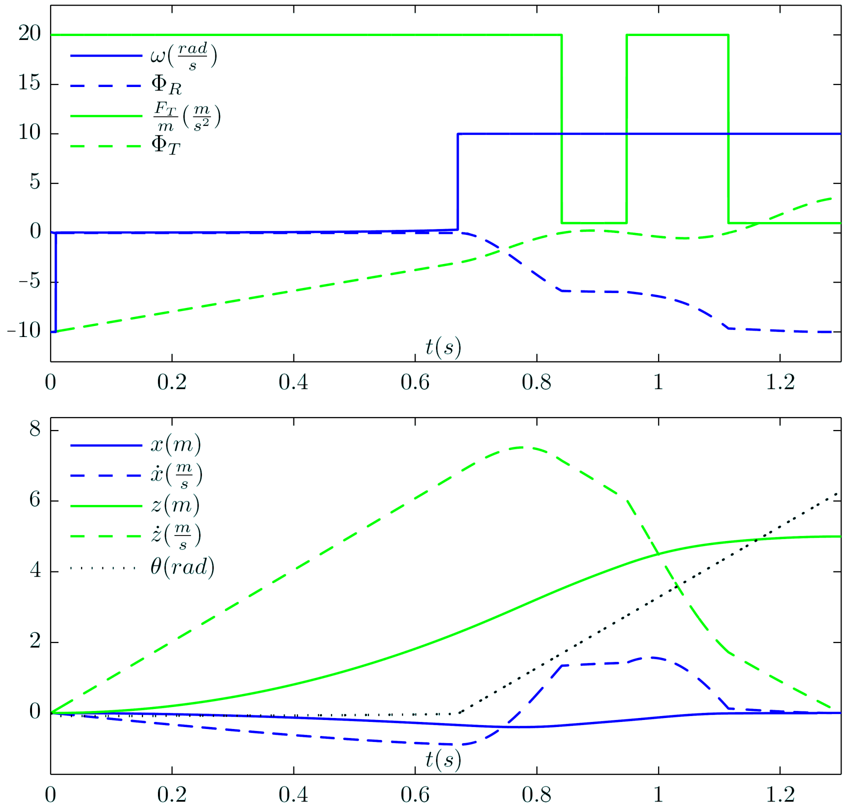

Figure 4. Input and state trajectories of a vertical displacement maneuver with $z_T = 5\,\text{m}$. In the plot of the control inputs, the switching functions are also drawn, but note that they are scaled to fit into the plot. The switches of the control input trajectories can be obtained by following the arrow line from $t = 0\,\text{s}$ towards $t = T$ in Figure 2.

consideration is not proven here, but has shown to always be correct for the computed horizontal maneuvers, with the resulting symmetric maneuvers satisfying the minimum principle. Using this assumption during the STO, the number of optimization variables can be reduced and the computation of horizontal displacements becomes particularly simple.

Maneuvers have been computed for a displacement x_T ranging from 0.1 m to 15 m, with a step size of 0.1 m. Figure 5 shows the maneuver duration T and the switching times as a function of x_T.

For $x_T \leq 1.5\,\text{m}$, the maneuver is bang-bang with no singular arcs. At the beginning the quadrotor turns at maximum rate, and around the maximum pitch angle the thrust is switched to its maximum for acceleration. Then it turns in the negative direction to decelerate around the minimum peak of θ, before it goes back to $\theta = 0$. At $x_T = 1.6\,\text{m}$, two singular arcs appear. Roughly speaking, the pitch angle is kept at $\theta \approx \pm\pi/2$ for acceleration and deceleration, respectively. Because a trade-off between fast acceleration in x and maintaining altitude in z is necessary, the pitch angle is not exactly $\theta = \pm\pi/2$ within the singular arcs, and is not constant. For $x_T \geq 7.9\,\text{m}$, the two singular arcs merge: The quadrotor turns smoothly to a negative θ for deceleration, instead of a sharp turn in

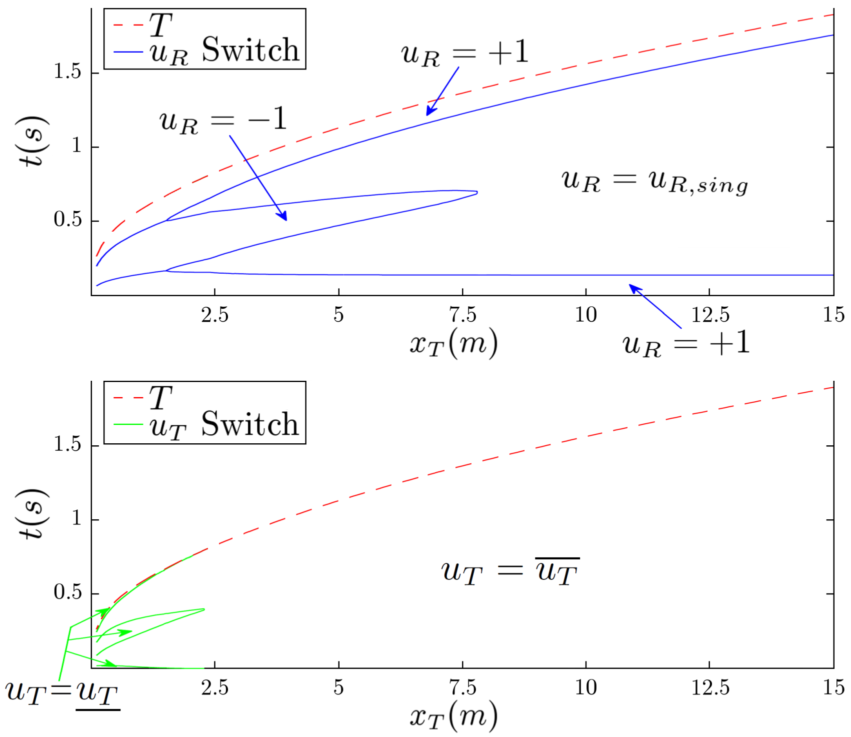

Figure 5. Maneuver duration T as function of the final displacement x_T for purely horizontal rest-to-rest maneuvers. Additionally, the switching times of u_R are drawn in the plot on the top, and the switching times of u_T in the plot on the bottom.

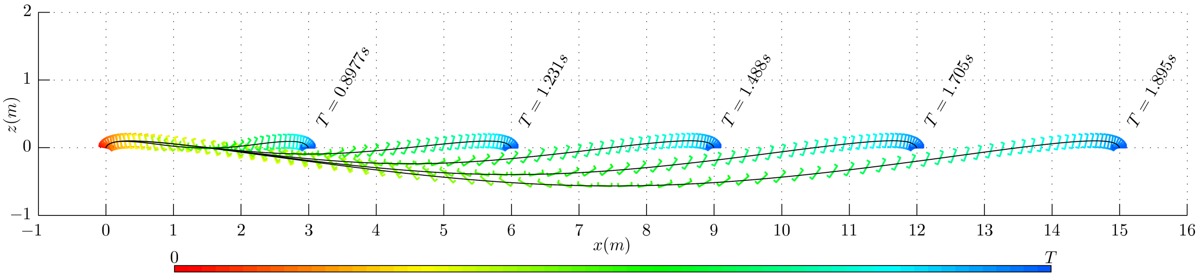

Figure 6. Illustration of maneuvers for a purely horizontal displacement between 3 m and 15 m. The maneuvers satisfy the minimum principle and for each maneuver, a quadrotor is plotted every 0.02 s.

the middle of the maneuver. For maneuvers with $x_T \geq 2.4\,\mathrm{m}$, the thrust control input is always at its maximum value. Figure 6 shows an illustration of some selected maneuvers.

5.3 General Displacements

For general two-dimensional displacements, the initial and final states are

$$\mathbf{x}_0 = (0,0,0,0,0), \quad \mathbf{x}_T = (x_T, 0, z_T, 0, 0). \tag{36}$$

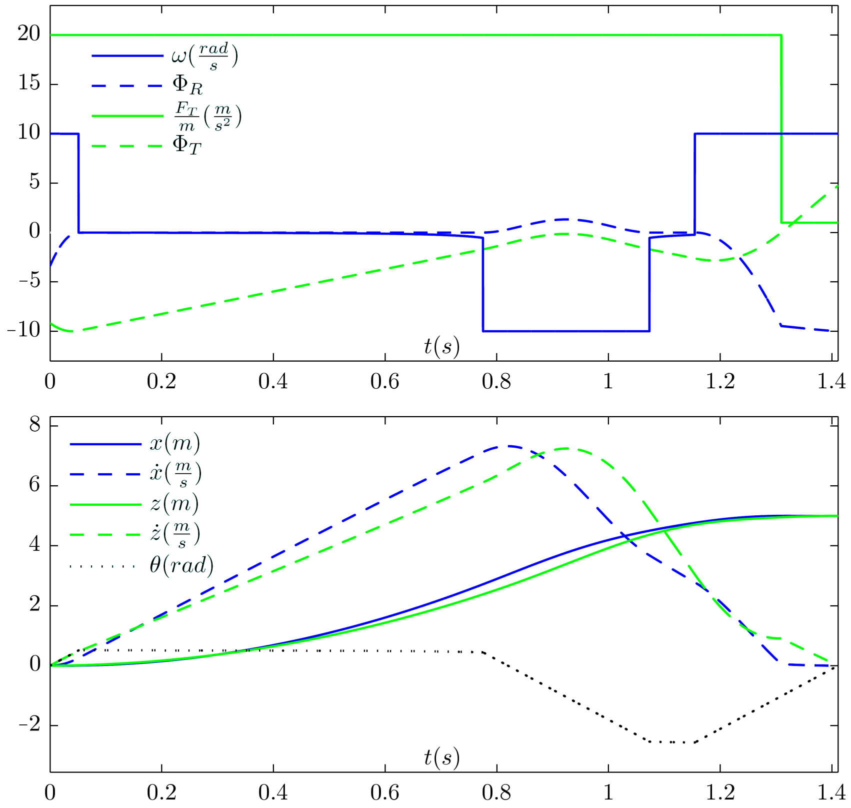

Figure 7. Input and state trajectories of an example maneuver with $x_T = 5\,\mathrm{m}$ and $z_T = 5\,\mathrm{m}$. The scaled switching functions are also drawn in the plot of the control inputs.

Since the final state of a general displacement contains two variables (x_T and z_T), the maneuver duration T cannot be easily plotted in a two-dimensional figure. As an example, a maneuver with a displacement of $5\,\mathrm{m}$ in horizontal and vertical direction, i.e. a maneuver with $x_T = 5\,\mathrm{m}$ and $z_T = 5\,\mathrm{m}$, is illustrated here. Figure 7 shows the resulting input, state, and switching function trajectories of this example maneuver. Note that the control inputs and the switching functions indeed fulfill the control laws (24) and (30). This implies that the minimum principle for time-optimality is satisfied.

6. Benchmarking of Quadrotor Designs and Controllers

This section demonstrates the usage of computed time-optimal maneuvers as a benchmarking tool for quadrotor designs and controllers.

6.1 Variation of the Quadrotor Design Parameters

The computation of time-optimal maneuvers allows the analysis of the impact of varying quadrotor parameters. These maneuvers allow the separation of effects of the physical parameters of the vehicle, from those of the control strategy by providing the achievable performance for a control law that fully utilizes the capabilities of the vehicle.

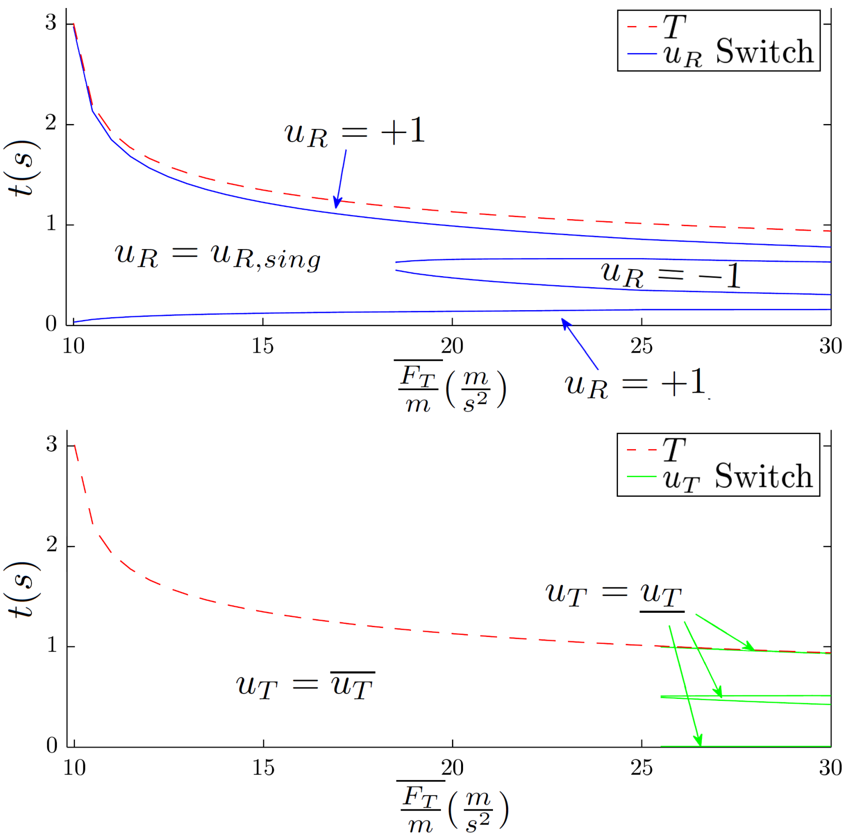

Figure 8. Maneuver duration T as function of the mass-normalized thrust $\overline{F_\mathrm{T}}/m$ for a horizontal maneuver with a displacement of $x_T = 5\,\mathrm{m}$. The switching times of u_R are drawn in the plot on the top, and the switching times of u_T in the plot on the bottom.

For a varying maximum pitch rate $\overline{\omega}$, the adjusted quadrotor performance can be obtained very easily: As the computations are done in the dimensionless coordinate system where the maximum pitch rate is normalized to unity, a changing $\overline{\omega}$ impacts only the back-transformation to the dimensional coordinates. Consequently, no recomputation of maneuvers is necessary and the structure of the switching times, as plotted in Figure 2 and Figure 5, does not change.

For varying thrust limits $\underline{F_\mathrm{T}}$ and $\overline{F_\mathrm{T}}$ (or equivalently $\underline{u_T}$ and $\overline{u_T}$), the impact of the changing parameters is not straight-forward. Since $\underline{u_T}$ and $\overline{u_T}$ are used during the computation of the maneuvers, a complete recalculation is required and the structure of the switching time evolution in Figure 2 and Figure 5 may change. As an example, maneuvers for a horizontal displacement of $x_T = 5\,\mathrm{m}$ have been computed with a maximum mass-normalized thrust $\overline{F_\mathrm{T}}/m$ between $10\,\mathrm{m\,s^{-2}}$ and $30\,\mathrm{m\,s^{-2}}$ at a step size of $0.5\,\mathrm{m\,s^{-2}}$, while the minimum thrust $\underline{F_\mathrm{T}}/m$ was kept constant at $1\,\mathrm{m\,s^{-2}}$. The resulting maneuver duration T and the switching times are shown in Figure 8.

The gravitational acceleration g poses a lower limit: If the mass-normalized thrust $\overline{F_\mathrm{T}}/m$ approaches g, the maneuver duration approaches infinity because the quadrotor needs all the available thrust force to maintain hover, and no horizontal displacement can be achieved without height loss. As $\overline{F_\mathrm{T}}/m$ increases, the slope of the T-curve decreases, which means that the performance gain per additional thrust becomes smaller. It follows that for large thrust-to-weight ratios, an increase of the available thrust force does not lead to significantly better performance with respect to the maneuver duration, with respect to horizontal displacements. To achieve faster displacements, the maximum pitch rate $\overline{\omega}$ would have to be increased as well.

6.2 Benchmarking of a Linear Controller

To demonstrate the use of time-optimal maneuvers as a controller benchmark, the performance of a linear controller, as commonly found in quadrotor literature, is evaluated. The time-optimal maneuvers provide a lower bound on the achievable maneuver duration, against which the maneuver duration using the linear controller is compared.

The linear controller consists of cascaded position and attitude control loops, as often used in quadrotor control (see, for example, [12, 19], and references therein).

The two-dimensional model presented in Section 2 can be linearized around the hover point $F_\mathrm{T} = m\mathrm{g}, \theta = 0$, yielding the linearized dynamics

$$\ddot{x} = \mathrm{g}\theta \tag{37}$$

$$\ddot{z} = \frac{F_\mathrm{T}}{m} - \mathrm{g} \tag{38}$$

$$\dot{\theta} = \omega. \tag{39}$$

The pitch angle is straightforward to control using a proportional controller:

$$\omega = \frac{1}{\tau_\theta}(\theta_d - \theta), \tag{40}$$

where θ_d is the desired pitch angle that is computed by the position control loop, and τ_θ is the time constant of the attitude control loop.

The two translational degrees of freedom decouple entirely in the linearization, allowing straightforward controller designs for each of them:

$$F_\mathrm{T} = m\left(-2\frac{\zeta_z}{\tau_z}\dot{z} - \frac{1}{\tau_z^2}(z - z_d)\right), \tag{41}$$

$$\theta_d = \frac{1}{\mathrm{g}}\left(-2\frac{\zeta_x}{\tau_x}\dot{x} - \frac{1}{\tau_x^2}(x - x_d)\right). \tag{42}$$

In the above equations, τ_x and τ_z are the respective closed loop time constants, and ζ_x and ζ_z are the respective damping ratios.

The saturations of the control inputs ω and F_T are applied with the same values as for the computation of time-optimal maneuvers. Additionally, the desired pitch angle θ_d is limited to $|\theta_d| \leq \pi/2$.

It is important to note that this controller is not designed for optimal performance, and performance cannot be expected to match the many more sophisticated controllers that have been presented (see, for example, [4, 5, 11, 17, 18, 21]). It is, however, a useful demonstration of the benchmarking of common 'everyday' controllers, and their performance.

The parameters of the controller are shown in Table 2, and are based on the parameters of controllers used in the Flying Machine Arena test bed. These parameters are based on manual tuning for all-round usability in the testbed, rather than optimizing simulated performance.

We compare the performance of the linear controller to the achievable performance when using the model dynamics (3)-(5) and performing translations that start and end at rest. To evaluate the performance of the linear controller, the target translation is provided as a setpoint x_d and z_d, and the closed-loop dynamics are simulated until the state settles. The duration of the maneuver is taken to be the time until the position error remains within 1% of the translation distance.

Figure 9 shows the duration of purely horizontal maneuvers as a function of the translation distance. The results show that this linear controller achieves maneuver times that, depending on the translation distance, are between approximately 175% and 880% of the minimal achievable time. As one expects, the maneuver duration is approximately constant for small translations, but varies significantly for large translations where model nonlinearities become more dominant. The non-monotonicity of the maneuver duration is caused by position oscillations that either lie within the 1% band defining the end of the maneuver, or slightly exceed it.

Figure 10 demonstrates the same comparison between the linear controller and time-optimal maneuvers, this time demonstrating the performance for purely vertical maneuvers. When using the linear controller, the pitch angle θ always remains at 0. The system dynamics are therefore fully linear, except for control input saturations. This is represented in the three distinct regions in the plot: Initially, the maneuver duration is constant. In a second region, the upper thrust constraint is reached, increasing the maneuver duration. In the third section, both upper and lower constraints are active during the maneuver, and the maneuver duration rises faster. The maneuver duration using the linear controller is between 155% and 1100% of the minimal time.

Parameter	Value	Description
τ_θ	0.18 s	Attitude control time constant
τ_x	0.35 s	Horizontal translation control time constant
ζ_x	0.95	Horizontal translation control damping ratio
τ_z	0.25 s	Vertical translation control time constant
ζ_z	0.8	Vertical translation control damping ratio

Table 2. Parameters of the linear controller

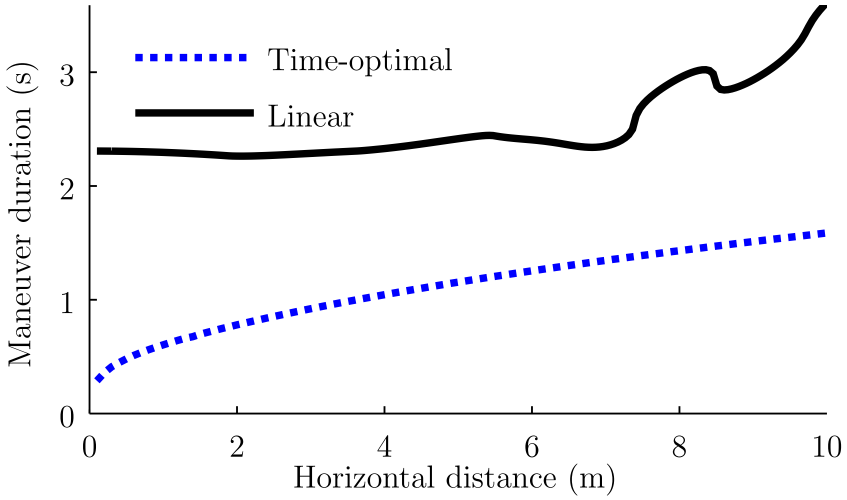

Figure 9. Comparison of maneuver times for purely horizontal maneuvers. The maneuver using the linear controller is considered finished when the position error remains within 1% of the desired translation.

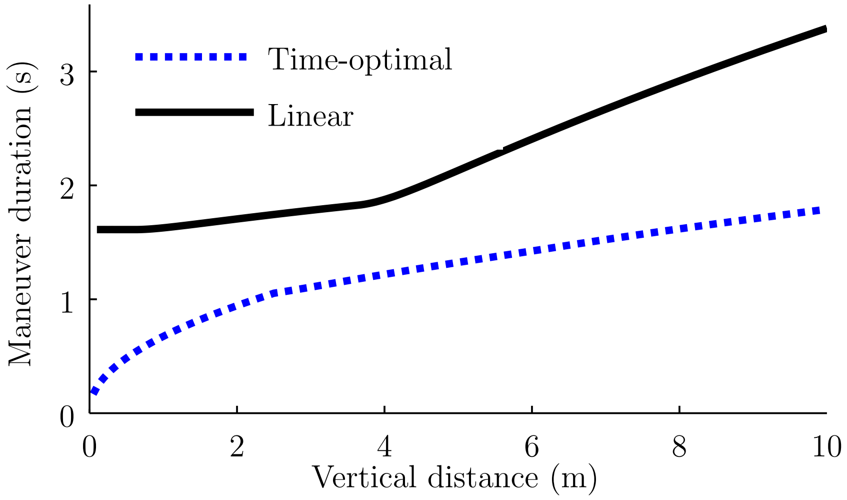

Figure 10. Comparison of maneuver times for purely vertical maneuvers. The maneuver using the linear controller is considered finished when the position error remains within 1% of the desired translation.

The above results show that the linear controller with saturations leaves considerable room for improvement. The the highest performance gains are achievable for very small translations, which take approximately constant time with the linear controller.

7. Experimental Results

Selected numerical results were experimentally validated by applying them on quadrotor vehicles in the ETH Zurich Flying Machine Arena. The vehicles are based on Ascending Technologies 'Humminbird' quadrocopters [8], but have been modified with custom electronics providing additional communications interfaces, sensors with higher dynamic range, and access to low-level control functions [17].

Trajectories were recorded using an infrared motion tracking system. Using retro-reflective markers mounted to the vehicle, position and attitude were measured at a rate of 200 Hz. The vehicle velocity was obtained through differentiation of non-causally filtered position measurements.

The control input trajectories are transfered to the quadrotor vehicle ahead of the start of the experiment. A hover controller is employed to stabilize the vehicle at the initial state. The maneuver is triggered and the vehicle executes the control input trajectories, using only feedback from the on board gyroscopes to control its rotational rates. The trajectory is sampled and executed by the on-board microcontroller at 800 Hz.

Considering that the numerical results were obtained using a first-principles model that, as discussed in Section 2.1, neglects a number of known effects, executing the numerically computed input trajectories directly is not sufficient to achieve a maneuver that is comparable to the simulation results. In order to adapt the input trajectories to modeling inaccuracies, a model-based policy gradient learning algorithm was applied. This algorithm was presented in [17] and [16], and minimizes the final state error over multiple iterations of the maneuver. The results shown here were obtained after convergence of the learning algorithm.

A video of the experiments presented herein is available on the first author's website, and as an electronic appendix to this article.

Figure 11 shows the state trajectories of two maneuvers after convergence of the policy gradient learning algorithm. The upper graph shows the state trajectories of a purely vertical translation of 5 m, the numerical results of which were shown in Figure 4. The lower graph shows a translation of 5 m in both coordinates, for which the numerical results were shown in Section 5.3 (Fig. 7).

For both experiments, the total duration of the maneuver was longer than the calculated duration (approximately 17% for the vertical translation, and approximately 20% for the translation in both coordinates). Multiple reasons for this can be seen in the state trajectories:

The pitch angle trajectories θ show inaccuracies, which can be explained by unmodeled rotational accelerations and propeller dynamics. This is particularly obvious around the switching times of the rotational rate input (e.g. around $t = 0.9\,\text{s}$ in the upper graph, and around $t = 1.4\,\text{s}$ in the lower graph. This highlights the limitations of the dynamical model introduced in Section 2: The shape of the trajectory is similar, but with significant differences when significant changes in the pitch angle occur.

The velocity trajectories show a loss of acceleration at high speeds. This is very clearly visible in the purely vertical displacement (upper plot), where the rate of increase of vertical velocity $\dot{z}$ reduces as the velocity increases ($t = 0.2\,\text{s}$ to $0.9\,\text{s}$). Well-known aerodynamic effects on propellers [13] provide a plausible explanation for this behavior: For a given propeller shaft power, the thrust produced by the propeller decreases significantly as the flight speed increases.

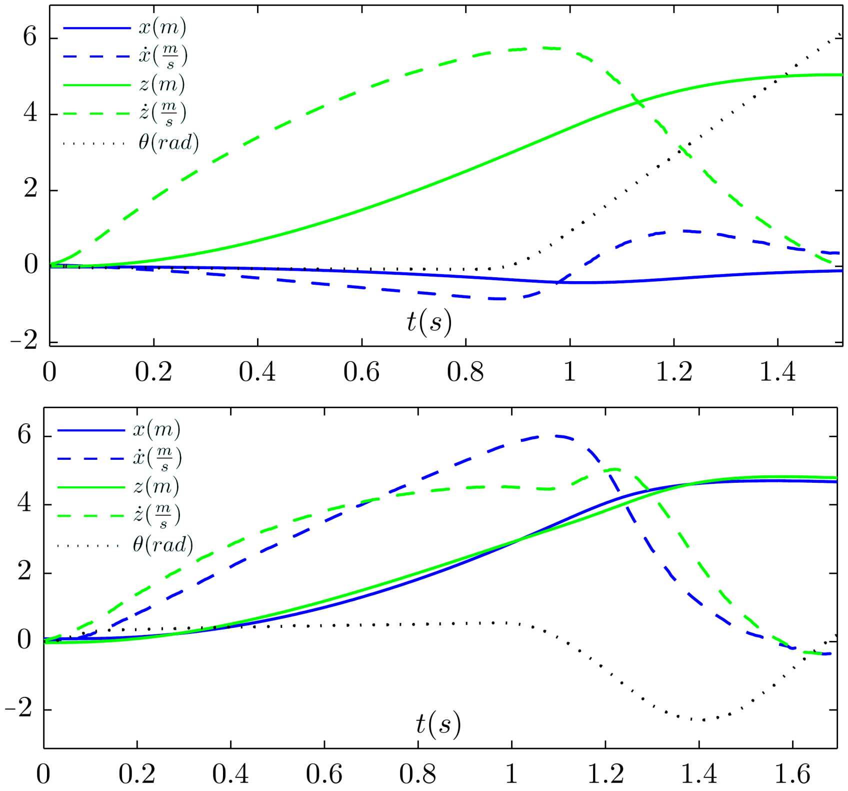

Figure 11. Measured state trajectories for two example maneuvers: Final state $x_T = 0$ and $z_T = 5\,\mathrm{m}$ (top), $x_T = 5\,\mathrm{m}$ and $z_T = 5\,\mathrm{m}$ (bottom). The numerical results for these maneuvers are shown in Figures 4 and 7, respectively. It can be seen that the measured trajectories are similar to the numerical ones, with unmodeled dynamics apparent at high speeds and around switching times.

Inspecting the velocity trajectories more closely, it also becomes apparent that sudden changes of acceleration, such as at the beginning of both maneuvers, are not achieved in the experiment. This behavior can be explained by examining the underlying propeller dynamics: For a sudden thrust change, the propeller speed must be increased or decreased instantaneously. The true propeller speed change dynamics are dictated by the available current and by the motor controllers.

The differences between the trajectories are further highlighted in Figure 12, where experimental and simulative results are superimposed. While the direct comparison highlights the longer duration of the maneuver, it is also clearly visible that the general shape of the simulation results is matched well by the experiments. With the ability to transfer maneuvers from simulation to the experimental platform, it is possible to perform comparative studies not only in simulation, but also in reality. For accurate benchmarks, care should be taken to compare results from similar sources (for example, obtained using the same simulation, as shown in Section 6). If, when assessing performance, numerical results for time-optimal maneuvers are to be compared to experimental results, it may be important to account for differences between experimental and simulative results.

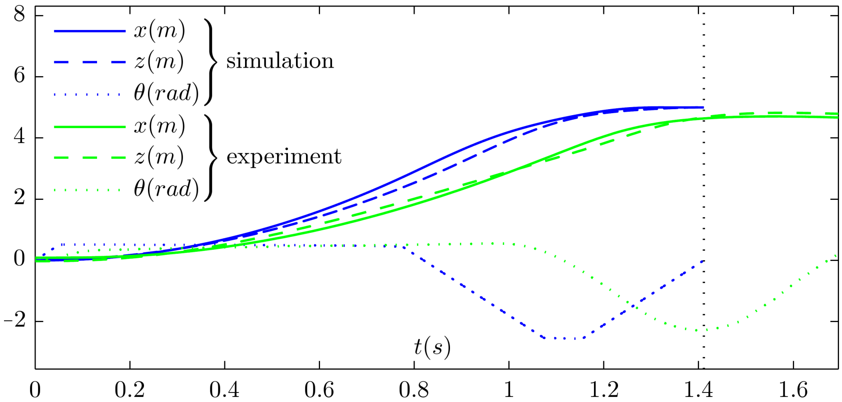

Figure 12. Overlay of trajectories from simulation and experiment: Final state $x_T = 5\,\mathrm{m}$ and $z_T = 5\,\mathrm{m}$. The direct comparison highlights the longer duration of the experimental trajectory, with peak velocities about 16% and 40% lower in x and z, respectively.

8. Conclusion and Future Work

In this paper, a benchmarking methodology for quadrocopters was presented. Using a two-dimensional first-principles model, the algorithm presented herein computes maneuvers that satisfy Pontryagin's minimum principle with respect to time-optimality. Using a non-dimensionalized model, the quadrotor vehicle is characterized by two parameters.

Resulting maneuvers for selected initial and final states were illustrated, highlighting the structure of time-optimal maneuvers. The use of this method to quantify performance gains through changes of physical quadrotor parameters was demonstrated, and the benchmarking of a control algorithm was demonstrated by benchmarking a linear controller as it might be used for relatively simple control tasks.

We expect that this method will enable performance benchmarking of quadrotor controllers and quadrotor design decisions. Furthermore, the insights gained into the structure of time-optimal quadrotor maneuvers may be useful in the development of more advanced control strategies.

To confirm the validity of computed maneuvers, and to demonstrate the transferability to real quadrocopters, the maneuvers were successfully demonstrated in the ETH Zurich Flying Machine Arena testbed. A possible extension would be the experimental benchmarking of control strategies. While the optimality conditions apply only to the first-principles model, the experimental results may still serve as a useful reference point that experimental results of other controllers can be compared to.

In tasks that require specific point-to-point motions, the trajectories computed with this algorithm could be applied, as demonstrated by the experimental results. While the computation of time-optimal trajectories is not fast enough to be performed on-line, a precomputed set of trajectories could be stored and applied for specific motions.

An interesting extension of this work would be the computation of time-optimal maneuvers in three dimensions. While the coordinate system can be appropriately transformed to compute all maneuvers starting and ending at standstill using the two-dimensional model, additional insight may be gained through the ability to compute arbitrary maneuvers.

A further possible extension of the work herein could be the development of algorithms that permit significantly faster computation of time-optimal maneuvers, allowing such maneuvers to be computed at timescales that make them useful in on-line planning scenarios.

A. Algorithm for Calculation of Time-Optimal Maneuvers

This appendix discusses the numerical algorithm presented in Section 4 in more detail, with a focus on how the individual steps were implemented. This implementation (in MATLAB) of the algorithm is available for free use on the first author's website, and is submitted along with this article.

This appendix follows the outline of Section 4, first introducing maneuvers containing no singular arcs in Sections A.1 - A.3, and then showing modifications for bang-singular maneuvers in Section A.4.

Figure 13 shows a flowchart diagram of the algorithm for bang-bang maneuvers, and in the following, the three steps are introduced in detail.

A.1 Switching Time Optimization

Due to the assumption that the optimal solution is a bang-bang maneuver, the control trajectory $\mathbf{u}$ can be efficiently parameterized by the initial control vector $\mathbf{u}(t=0)$ and the switching times of the two control inputs, denoted by the sets

$$\begin{aligned} \{T_{u_R}\} &= T_{u_R}^i \qquad &&\text{for } i = 1, 2, \ldots, N_R, \\ \{T_{u_T}\} &= T_{u_T}^j \qquad &&\text{for } j = 1, 2, \ldots, N_T. \end{aligned} \tag{43}$$

N_R and N_T are the number of switches of the rotational control input and the thrust input, respectively. The principle of STO is to choose N_R and N_T, and to then improve an initial choice of the switching times $\{T_{u_R}\}_{ini}$ and $\{T_{u_T}\}_{ini}$, until a control trajectory is found that guides the quadrotor from $\mathbf{x}_0$ to $\mathbf{x}_T$ with an acceptable accuracy. The final state error is measured using the scalar final state residual function

$$P_{res}(\{T_{u_R}\}, \{T_{u_T}\}, T) = (\mathbf{x}(T) - \mathbf{x}_T)^T W (\mathbf{x}(T) - \mathbf{x}_T), \tag{44}$$

where the matrix $W = \text{diag}(w_1, w_2, w_3, w_4, w_5)$ contains the weights of the different state errors. The final state $\mathbf{x}(T)$ resulting from the chosen switching times can be obtained by numerically integrating the system dynamics $f(\mathbf{x}, \mathbf{u})$ over the interval $[0, T]$, where $\mathbf{u}$ is defined by the initial control inputs $\mathbf{u}(t=0)$ and the switching times $\{T_{u_R}\}$ and $\{T_{u_T}\}$.

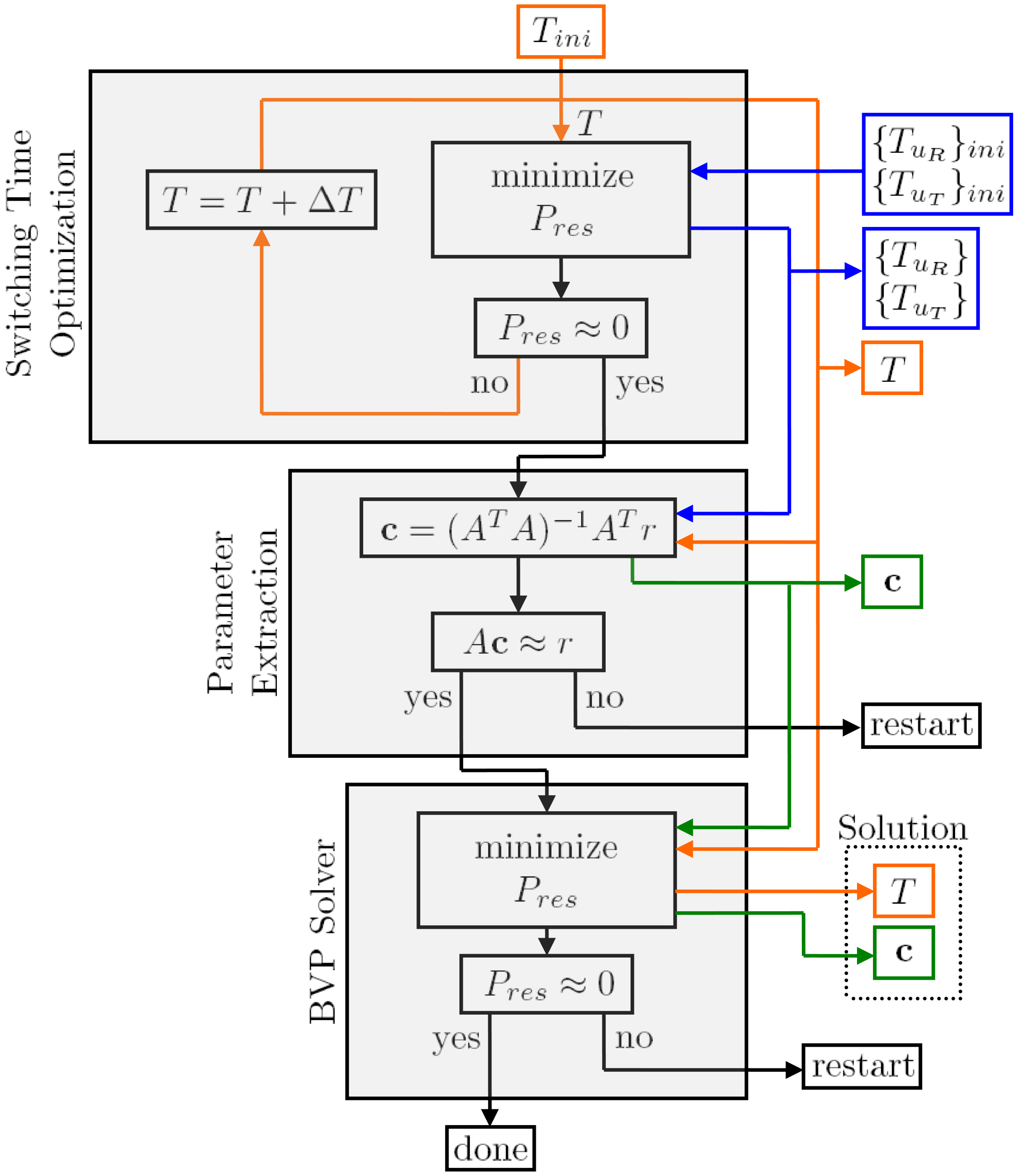

Figure 13. Flowchart diagram of the algorithm that computes bang-bang maneuvers satisfying the minimum principle. The three steps are presented in detail in Sections A.1-A.3. In this graph, $\approx$ is used to denote that the equation must be solved to acceptable accuracy.

The maneuver duration T is not known a priori and we seek the minimum T for which $P_{res} = 0$ can be obtained. The problem can be written as

$$\begin{aligned} &\text{find} && \{T_{u_R}\}, \{T_{u_T}\}, T \\ &\text{subject to} && P_{res}(\{T_{u_R}\}, \{T_{u_T}\}, T) = 0, \\ & && T \leq \{T\}_{ach}, \end{aligned} \tag{45}$$

where $\{T\}_{ach}$ is the set of all T for which $P_{res} = 0$ is achievable, implying that the maneuver to be found is the one with the shortest possible duration.

The solution of (45) is computed by a two-step algorithm: For an initially small, fixed maneuver duration T, the state residual P_{res} is minimized by varying the switching times $\{T_{u_R}\}$ and $\{T_{u_T}\}$ using a simplex search method (this choice was based on the observation that derivative-free optimization algorithms have shown to perform significantly better in this optimization). After the minimization, T is increased using the secant method

$$T_{i+1} = T_i + \frac{T_i - T_{i-1}}{(P_{res,i-1}/P_{res,i}) - 1}, \tag{46}$$

or by a constant value if convergence of the secant method is not assumed, see [6]. These two steps are repeated until $P_{res} = 0$ is achieved. Since the initial value of T is chosen to be too small to complete the maneuver, and since T is successively increased, the algorithm delivers a value close to the smallest T for which $P_{res} = 0$ is achievable.

The choice of the number of switches is based on the user's intuition and experience from the computation of other maneuvers. If the number is chosen too high, the algorithm can converge to the correct result by producing dispensable switching times, as discussed below. The initial guess for the duration of the maneuver T must be chosen to be too short to compute the maneuver, and can be obtained from a guess based on the vehicle's translational acceleration capabilities, or on similar maneuvers.

A.2 Parameter Extraction

After having found a bang-bang trajectory that brings the quadrotor from the initial state $\mathbf{x}_0$ to the desired final state $\mathbf{x}_T$, it is necessary to verify that it is a solution to BVP (32). Therefore, the constant vector $\mathbf{c} = (c_1, c_2, c_3, c_4)$ must be determined, based on the trajectories resulting from the STO.

Dispensable Switching Times If the number of switches N_R and N_T was chosen too high, then the STO may converge to a solution containing dispensable switching times, which in fact do not represent switches. Therefore, before the constant vector $\mathbf{c}$ is computed, all switches at $t = 0$ and $t = T$ are removed, and the initial control vector $\mathbf{u}(0)$ is adjusted accordingly. Furthermore, two switches of the same control input, which occur at the same time, are dispensable as well and must, consequently, also be removed.

Conditions on the Trajectory of Φ_R The switching function Φ_R must be zero whenever the control input u_R switches. From the STO, the set of switching times $\{T_{u_R}\}$ is given, and for each element of this set, Φ_R must vanish. This leads to the conditions

$$\Phi_R(T^i_{u_R}) = 0 \qquad \text{for } i = 1, 2, \ldots, N_R. \tag{47}$$

As shown in Section 3, only the derivative $\dot{\Phi}_R$ of the switching function is known a priori.

However, once the state trajectories are known from the STO, the condition $H \equiv 0$ (which must hold if the maneuver is time-optimal) can be used to compute Φ_R. Recalling the Hamiltonian (14) and using the definition $\Phi_R = p_5$ yields

$$\Phi_R = \frac{1 + p_1\dot{x} + p_2 u_T \sin\theta + p_3\dot{z} + p_4(u_T\cos\theta - 1)}{-u_R}. \tag{48}$$

As shown in (17), the first four costates p_i are all linear in $\mathbf{c}$. The above equation can therefore be written as a linear function of $\mathbf{c}$:

$$\begin{aligned}\Phi_R = \frac{1}{u_R}&\left(-1 + c_1(-\dot{x} + t u_T \sin\theta) + c_2(-u_T\sin\theta)\right.\\ &\left.+ c_3(-\dot{z} + t(u_T\cos\theta - 1)) + c_4(-u_T\cos\theta + 1)\right).\end{aligned} \tag{49}$$

Given the linear form of Φ_R, Equation (47) states N_R linear conditions on the constant vector $\mathbf{c}$.

The derivative $\dot{\Phi}_R$ is given by Equation (31). For a trajectory that satisfies the minimum principle, the integral of $\dot{\Phi}_R$ must coincide with the trajectory of Φ_R given by (49). Hence, for an arbitrary interval $[t_1, t_2] \in [0, T]$,

$$\begin{aligned}\Phi_R(t_2) - \Phi_R(t_1) &= \textstyle\int_{t_1}^{t_2} \dot{\Phi}_R dt\\ &= \textstyle\int_{t_1}^{t_2} (-p_2 u_T\cos\theta + p_4 u_T \sin\theta) dt\end{aligned} \tag{50}$$

must hold, where the left side of the equation is computed using $H \equiv 0$, i.e. by (49). The costates p_2 and p_4 are linear functions of $\mathbf{c}$, and the above equation can be written as

$$\begin{aligned}\Phi_R(t_2) - \Phi_R(t_1) = c_1 &\int_{t_1}^{t_2} t u_T \cos\theta dt - c_2 \int_{t_1}^{t_2} u_T\cos\theta dt\\ -c_3 &\int_{t_1}^{t_2} t u_T \sin\theta dt + c_4 \int_{t_1}^{t_2} u_T \sin\theta dt.\end{aligned} \tag{51}$$

To set up conditions on $\mathbf{c}$ based on (51), the maneuver interval $[0, T]$ is divided into $N_R + 1$ subintervals that are separated by the switching times $\{T_{u_R}\}$, i.e.

$$[0, T] = \bigcup\Big\{[0, T_{u_R}^1], [T_{u_R}^1, T_{u_R}^2], \ldots, [T_{u_R}^{N_R}, T]\Big\}. \tag{52}$$

This choice is beneficial with respect to the computational effort, because the switching function Φ_R must vanish at the switching times; the left side of (51) can be set to zero for all intervals, except for the first and the last one. The $N_R + 1$ intervals describe $N_R + 1$ additional linear conditions on the constant vector $\mathbf{c}$.

Conditions on the Trajectory of Φ_T Since the thrust switching function Φ_T is known explicitly, the conditions resulting from $\{T_{u_T}\}$ are straightforward. From the fact that Φ_T must vanish at each switch of u_T, the condition

$$\Phi_T(T^i_{u_T}) = 0 \qquad \text{for } i = 1, 2, \ldots, N_T \tag{53}$$

must be satisfied, where the set $\{T_{u_T}\}$ is given by the STO. The thrust switching function (26) is a linear function of the costates p_2 and p_4, and again linear in $\mathbf{c}$:

$$\Phi_T = -c_1 t \sin\theta + c_2 \sin\theta - c_3 t \cos\theta + c_4 \cos\theta. \tag{54}$$

This linear form of the thrust switching function Φ_T allows one to define N_T additional linear conditions on the elements of the constant vector $\mathbf{c}$, based on the conditions from (53).

Condition Matrix Equation For the minimum principle to be satisfied, a constant vector $\mathbf{c}$ that fulfills all the linear conditions to an acceptable accuracy must exist. The conditions on $\mathbf{c}$ derived above are therefore combined into a matrix equation, which we denote as

$$A\mathbf{c} = r. \tag{55}$$

The matrix A is of size $(N_c \times 4)$ and the vector r has the length N_c, where N_c is the total number of linear conditions:

$$N_c = 2N_R + N_T + 1 \tag{56}$$

For all maneuvers considered here, the system of equations (55) is overdetermined, permitting no exact solution. Therefore, the least squares solution of (55) is computed ([1]), which is given by

$$\mathbf{c}^* = (A^T A)^{-1} A^T r. \tag{57}$$

To verify that a solution to the overdetermined system of equations exists, $\mathbf{c}^*$ is substituted back into (55). If the error vector exceeds the expected numerical discrepancies[3], then the solution is considered to be invalid. In the context of the optimal control problem, this implies that there exists no constant vector $\mathbf{c}$ for which the minimum principle is fulfilled, and consequently the trajectories $\mathbf{x}$ and $\mathbf{u}$ resulting from the STO do not satisfy the minimum principle. A possible reason is that the chosen number of switches N_R and N_T and the initial values $\{T_{u_R}\}_{ini}$ and $\{T_{u_T}\}_{ini}$ did not cause the STO to converge to the desired maneuver. This may be corrected by varying these parameters. Another reason for the lack of a solution could be that the time-optimal maneuver for the given boundary conditions contains singular arcs, a case that will be discussed in Section 4.4.

---

[3] Numerical discrepancies are to be expected from both the accuracy to which the STO optimization was solved, and numerical integration errors. The tolerance to which the system of equations must be satisfied is defined by the user based on values seen in other maneuvers.

If the condition matrix equation is satisfied to an acceptable accuracy, then a valid parameter vector $\mathbf{c}$ has been found and the parameter extraction step is complete.

A.3 BVP Solver

To verify that BVP (32) is fulfilled and to minimize numerical errors, a last step is performed where the BVP is solved numerically: The state residual P_{res} is minimized by varying the constant vector $\mathbf{c}$ and the maneuver duration T. The problem can be written as

$$\begin{aligned} \text{minimize} \quad & P_{res}(\mathbf{c}, T) \\ \text{subject to} \quad & \dot{\mathbf{x}}_{\mathbf{a}} = f_a(t, \mathbf{x}_{\mathbf{a}}), \\ & \mathbf{x}_{\mathbf{a}}(0) = (\mathbf{x}_0, \Phi_R(0)). \end{aligned} \tag{58}$$

The constants $\mathbf{c}$ resulting from the parameter extraction and the maneuver duration T obtained by the STO are used as initial values. The optimization over the constants $\mathbf{c}$ and the terminal time T is carried out using a simplex algorithm. As these initial values are close to the exact solution, the BVP solver converges quickly, provided that the solution resulting from the STO is indeed a solution to the minimum principle. The initial value of the switching function $\Phi_R(0)$ can be obtained by the condition $H \equiv 0$, i.e. by Equation (49), evaluated at $t = 0$. If P_{res} is sufficiently small after the minimization, the maneuver satisfies the boundary conditions of the final state being reached, and the algorithm has terminated successfully.

A.4 Modified Algorithm for Bang-Singular Maneuvers

The algorithm described above is able to solve BVP (32), provided that the resulting maneuver does not contain singular arcs. In the general case, however, the time-optimal maneuver is bang-singular, and the algorithm needs to be modified to take possible singular arcs into account.

Within a singular arc, the trajectory of u_R is given by Equation (23) and depends on the constants $\mathbf{c}$. Due to this dependency, computing the constants $\mathbf{c}$ after the STO is no longer sufficient, since they determine the singular input and have an impact on the maneuver trajectory. The parameter extraction is therefore embedded into the STO, and the resulting algorithm consists of two successive steps:

1. Applying STO, a maneuver that brings the quadrotor to the desired final state is found, and in parallel, a constant vector $\mathbf{c}$ that fulfills the condition matrix equation resulting from the parameter extraction is computed.

2. Having a reasonable initial guess of the switching times, of the maneuver duration T, and of the constant vector $\mathbf{c}$, a BVP solver that computes a solution to BVP (32) is applied.

Figure 14 shows a flowchart diagram of the algorithm to find bang-singular solutions.

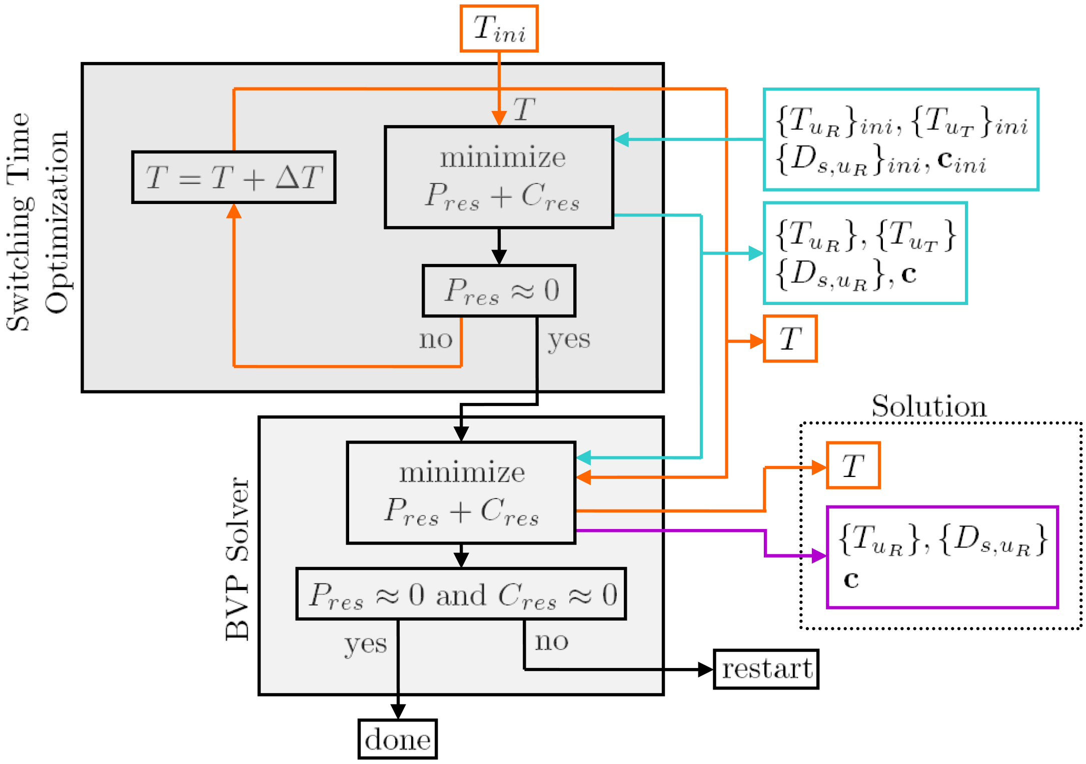
T_{ini}
Switching Time Optimization
T
$T = T + \Delta T$
minimize $P_{res} + C_{res}$
$P_{res} \approx 0$
no
yes
$\{T_{u_R}\}_{ini}, \{T_{u_T}\}_{ini}$ $\{D_{s,u_R}\}_{ini}, \mathbf{c}_{ini}$
$\{T_{u_R}\}, \{T_{u_T}\}$ $\{D_{s,u_R}\}, \mathbf{c}$
T
Solution
T
$\{T_{u_R}\}, \{D_{s,u_R}\}$ $\mathbf{c}$
BVP Solver
minimize $P_{res} + C_{res}$
$P_{res} \approx 0$ and $C_{res} \approx 0$
yes
no
restart
done

Figure 14. Flowchart diagram of the algorithm to compute bang-singular maneuvers that satisfy the minimum principle. The symbol $\approx$ is used to denote that the equation must be solved to acceptable accuracy.

Switching Time Optimization with Embedded Parameter Extraction For bang-singular maneuvers, u_R may stay within a singular arc for a particular duration each time it switches. We introduce a new set of parameters that describes the durations of the singular arcs, and denote the duration within the singular arc at the switching time $T^i_{u_R}$ as D^i_{s,u_R}. At the time $T^i_{u_R}$ the control input u_R enters the singular arc, and at time $T^i_{u_R} + D^i_{s,u_R}$ the singular arc is left and u_R switches to -1 or $+1$ [4]. A bang-singular maneuver is characterized by the sets

$$\begin{aligned} \{T_{u_R}\} &= T^i_{u_R} && \text{for } i = 1, 2, \ldots, N_R, \\ \{D_{s,u_R}\} &= D^i_{s,u_R} && \text{for } i = 1, 2, \ldots, N_R, \\ \{T_{u_T}\} &= T^j_{u_T} && \text{for } j = 1, 2, \ldots, N_T. \end{aligned} \tag{59}$$

Within a singular arc, u_R is given by Equation (23) and its trajectory depends on the

---

[4] It is necessary to additionally specify whether u_R switches to -1 or $+1$ at the end of the singular arc. We employ the convention that u_R switches to the opposing value of the one before the singular arc. A singular arc where u_R returns to the same value after the singular arc can be modeled by an additional switch at the end of the singular arc, with the corresponding duration of the additional singular arc being zero.

constants $\mathbf{c}$. The final state residual P_{res} is therefore not only a function of the maneuver duration T and of the sets of the switching times, but also of the constant vector $\mathbf{c}$. Accordingly, the state residual may be written as

$$\begin{aligned} & P_{res}(\{T_{u_R}\}, \{T_{u_T}\}, \{D_{s,u_R}\}, \mathbf{c}, T) \\ & \qquad\qquad = (\mathbf{x}(T) - \mathbf{x}_T)^T W (\mathbf{x}(T) - \mathbf{x}_T). \end{aligned} \tag{60}$$

The new parameter set $\{D_{s,u_R}\}$ and the constant vector $\mathbf{c}$ are additional optimization variables during the STO.

If the solution is to satisfy the minimum principle, the optimization variables overconstrain the problem: For the solution to satisfy the optimality conditions, the control inputs must be the optimal control inputs, as specified by Equations (24) and (30). These optimal inputs could be found using $\mathbf{c}$ to compute the switching functions. This is avoided, however, because the separate optimization of the switching times and $\mathbf{c}$ has shown to be more robust.

Because only constants $\mathbf{c}$ that satisfy the condition matrix equation $A\mathbf{c} = r$ from the parameter extraction are a valid choice, we define the condition residual to be

$$C_{res}(\{T_{u_R}\}, \{T_{u_T}\}, \{D_{s,u_R}\}, \mathbf{c}, T) = (A\mathbf{c} - r)^T W_c (A\mathbf{c} - r), \tag{61}$$

where W_c is a diagonal matrix containing the weights of the different linear conditions. It is important to note that the matrix A and the vector r are functions of the switching times $\{T_{u_R}\}$ and $\{T_{u_T}\}$, of the singular arc durations $\{D_{s,u_R}\}$, of the maneuver duration T, and of the constants $\mathbf{c}$. For a maneuver that satisfies the minimum principle, the condition residual C_{res} must vanish. Consequently, the STO problem for bang-singular maneuvers can be written as

$$\begin{aligned} \text{find} \quad & \{T_{u_R}\}, \{T_{u_T}\}, \{D_{s,u_R}\}, \mathbf{c}, T \\ \text{subject to} \quad & P_{res}(\{T_{u_R}\}, \{T_{u_T}\}, \{D_{s,u_R}\}, \mathbf{c}, T) = 0, \\ & C_{res}(\{T_{u_R}\}, \{T_{u_T}\}, \{D_{s,u_R}\}, \mathbf{c}, T) = 0, \\ & T \leq \{T\}_{ach}, \end{aligned} \tag{62}$$

where $\{T\}_{ach}$ denotes the set of all T for which $P_{res} = 0$ and $C_{res} = 0$ is achievable.

For bang-singular maneuvers, the sum of the state and the condition residual $P_{res} + C_{res}$ is minimized during the STO. For the computation of C_{res}, the matrix A and the vector r are required: The parameter extraction is no longer an isolated step, but needs to be performed for each evaluation of C_{res} within the STO minimization. The parameter extraction is not used to compute the constants $\mathbf{c}$ (which are optimization variables), but to compute A and r.

Additional Linear Conditions for Bang-Singular Maneuvers For the parameter extraction of bang-singular maneuvers, which is needed to obtain A and r, there exist additional linear conditions that take the requirements on the switching functions within singular arcs into account.

Additional Conditions on the Trajectory of Φ_R*:* Considering bang-singular maneuvers, the rotational switching function Φ_R must not only have a zero-crossing at each $T^i_{u_R}$, but it must also stay at zero for the duration of the corresponding singular arc D^i_{s,u_R}. An additional set of constraints is introduced, requiring that Φ_R is zero at the beginning and at the end of the singular arcs:

$$\begin{aligned} &\Phi_R(T^i_{u_R}) = 0 && \text{for } i = 1, 2, \ldots, N_R, \\ &\Phi_R(T^i_{u_R} + D^i_{s,u_R}) = 0 && \text{for } i = 1, 2, \ldots, N_R. \end{aligned} \tag{63}$$

Because these conditions do not imply that Φ_R is zero during the entire singular arc, it is necessary to verify the trajectory of Φ_R after the computation. If a switch $T^i_{u_R}$ has no singular arc, i.e. if $D^i_{s,u_R} = 0$, then the corresponding two conditions in (63) are identical. From this it follows that one additional condition results for each singular arc. We denote the number of singular arcs as N_s, hence (63) describes $N_R + N_s$ conditions. This means that N_s additional conditions have been identified, compared to the bang-bang case. As derived in Section A.2, these conditions are linear with respect to $\mathbf{c}$.

As the derivative of the rotational switching function $\dot{\Phi}_R$ is known explicitly, we demand that the integration value of $\dot{\Phi}_R$ between two switches of u_R is zero for bang-bang maneuvers. For bang-singular maneuvers, we pose similar conditions, but extra time intervals over the singular arcs are created. An integration value of zero does not imply that Φ_R stays at zero during the whole singular arc, but constant drifts of Φ_R are penalized. Hence, the intervals over which $\dot{\Phi}_R$ is integrated are

$$\begin{aligned} [0, T] \;=\; \bigcup \Big\{ &[0, T^1_{u_R}], [T^1_{u_R}, T^1_{s,u_R}], [T^1_{s,u_R}, T^2_{u_R}], \ldots \\ &\ldots, [T^{N_R-1}_{s,u_R}, T^{N_R}_{u_R}], [T^{N_R}_{u_R}, T^{N_R}_{s,u_R}], [T^{N_R}_{s,u_R}, T] \Big\}, \end{aligned} \tag{64}$$

where $T^i_{s,u_R} = T^i_{u_R} + D^i_{s,u_R}$ is used for a more compact notation. Analogously to the bang-bang case, a linear condition for each of these intervals can be constructed using Equation (51). If a switch has no singular arc, then $D^i_{s,u_R} = 0$ and the corresponding interval vanishes. Hence, for bang-singular maneuvers, $N_R + N_s + 1$ linear conditions on the constant vector $\mathbf{c}$ result. Compared to a bang-bang maneuver, N_s additional conditions are introduced.

Assuming that the thrust input u_T does not switch at the edges of the singular intervals, $\dot{\Phi}_R$ is continuous over the border of the singular arcs, as can be seen from (21). Consequently, the switching function Φ_R enters and leaves a singular arc tangentially. We therefore impose the conditions that the derivative $\dot{\Phi}_R$ is zero at the edges of every singular arc. For each singular arc, i.e. for each $D^i_{s,u_R} > 0$, two additional conditions result:

$$\begin{aligned} &\dot{\Phi}_R(T^i_{u_R}) = 0 && \text{for } i = 1, 2, \ldots, N_s, \\ &\dot{\Phi}_R(T^i_{u_R} + D^i_{s,u_R}) = 0 && \text{for } i = 1, 2, \ldots, N_s. \end{aligned} \tag{65}$$

The derivative of the rotational switching function is given by

$$\dot{\Phi}_R = (c_1 t - c_2) u_T \cos\theta + (c_4 - c_3 t) u_T \sin\theta, \tag{66}$$

which has been derived in Section 3. This is a linear function of the constants $\mathbf{c}$, and yields $2N_s$ additional conditions.

General Condition Matrix Equation: In total, $4N_s$ additional conditions have been identified. It follows that in the case of a bang-singular maneuver, the condition matrix equation

$$A\mathbf{c} = r \tag{67}$$

has N_c rows, with a total number of conditions of

$$N_c = 2N_R + N_T + 4N_s + 1. \tag{68}$$

The condition matrix equation is overdetermined as soon as the maneuver has at least one singular arc.

BVP Solver for Bang-Singular Maneuvers Similar to the algorithm for bang-bang maneuvers, the final step is the reduction of errors through the application of a BVP solver. If the maneuver contains singular arcs, Φ_R stays at zero for a nontrivial interval of time. Since the system is integrated numerically, Φ_R is near zero during the singular arcs, but does not vanish completely due to numerical inaccuracies. As Φ_R enters and leaves the singular arcs tangentially, defining a threshold value below which Φ_R is considered to be zero is not a straightforward task. For this reason, the rotational control trajectory u_R is not determined using the optimal control law (i.e. based on its switching function Φ_R), but is based on the sets $\{T_{u_R}\}$ and $\{D_{s,u_R}\}$. Consequently, $\{T_{u_R}\}$ and $\{D_{s,u_R}\}$ are optimizing variables during the BVP minimization, because they impact the control trajectory $\mathbf{u}$. Further, since the switching times of u_R are not determined based on the constants $\mathbf{c}$, the optimal control laws are not implicitly satisfied. One must thus ensure that the condition matrix equation is fulfilled, which is the case if C_{res} vanishes. Thus, as during the switching time optimization, the sum of the state residual P_{res} and the condition residual C_{res} is minimized. The BVP solver problem for bang-singular maneuvers becomes

$$\begin{aligned} \text{minimize} \quad & P_{res} + C_{res} \\ \text{subject to} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \\ & \mathbf{x}(0) = \mathbf{x}_0, \end{aligned} \tag{69}$$

where the control trajectory u_R is computed according to the switching times and singular arc durations, and u_T according to the optimal control law (30). Note that the arguments $(\{T_{u_R}\}, \{D_{s,u_R}\}, \mathbf{c}, T)$ of P_{res} and C_{res} have been omitted for reasons of clarity.

In the BVP Solver step, the N_T linear conditions resulting from the thrust input are trivially satisfied, because u_T is computed based on its switching function Φ_T. Hence, when the matrix condition equation is computed for the evaluation of C_{res} during the BVP minimization, it has only

$$N_{c,u_R} = 2N_R + 4N_s + 1 \tag{70}$$

rows, since the conditions resulting from u_T can be neglected.

For bang-singular maneuvers, the BVP solver is similar to the STO. The only differences are that the thrust input u_T is determined based on its control law (30), and that the maneuver duration T is an optimization variable, too, and not kept constant during the minimization of $P_{res} + C_{res}$.

Because u_R is not determined by its control law, and since a vanishing condition residual C_{res} does not guarantee that the control law holds, it is necessary to verify that the control law (24) is satisfied by inspecting the switching function Φ_R.

If the residuals P_{res} and C_{res} are sufficiently small after the minimization, and if the control law for the rotational input u_R is fulfilled, then the maneuver is a solution to BVP (32), and therefore satisfies the minimum principle with respect to time-optimality.

References

[1] Dennis S. Bernstein. *Matrix Mathematics.* Princeton University Press, 2005.

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I.* Athena Scientific, third edition, 2005.

[3] S. Bouabdallah, A. Noth, and R. Siegwart. PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. In *International Conference on Intelligent Robots and Systems*, 2004.

[4] Y. Bouktir, M. Haddad, and T. Chettibi. Trajectory Planning for a Quadrotor Helicopter. In *Mediterranean Conference on Control and Automation*, 2008.

[5] Ian D. Cowling, Oleg A. Yakimenko, and James F. Whidborne. A Prototype of an Autonomous Controller for a Quadrotor UAV. In *European Control Conference*, 2007.

[6] Germund Dahlquist and Ake Björck. *Numerical Methods.* Dover Publications, 2003.

[7] Hans Peter Geering. *Optimal Control with Engineering Applications.* Springer, 2007.

[8] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[9] Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. In *IFAC World Congress*, 2011.

[10] Gabriel M Hoffmann, Hao Huang, Steven L Waslander, and Claire J Tomlin. Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In *AIAA Guidance, Navigation and Control Conference*, 2007.

[11] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor Helicopter Trajectory Tracking Control. In *Conference on Decision and Control*, 2008.

[12] Jonathan P. How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.

[13] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering. In *International Conference on Robotics and Automation*, 2009.

[14] Li-Chun Lai, Chi-Ching Yang, and Chia-Ju Wu. Time-Optimal Control of a Hovering Quad-Rotor Helicopter. *Journal of Intelligent and Robotic Systems*, 45(2):115–135, June 2006.

[15] Urszula Ledzewicz, Helmut Maure, and Heinz Schattler. Bang-Bang and Singular Controls in a Mathematical Model for Combined Anti-Angiogenic and Chemotherapy Treatments. In *Conference on Decision and Control*, December 2009.

[16] Sergei Lupashin and Raffaello D'Andrea. Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters. In *IFAC World Congress*, 2011.

[17] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips. In *International Conference on Robotics and Automation*, 2010.

[18] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *International Symposium on Experimental Robotics*, 2010.

[19] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[20] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot. In *Australasian Conference on Robotics and Automation*, 2006.

[21] Oliver Purwin and Raffaello D'Andrea. Performing and Extending Aggressive Maneuvers Using Iterative Learning Control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[22] Emilio Roxin. The Existence of Optimal Controls. *The Michigan Mathematical Journal*, 9(2):109–119, 1962.

[23] Angela Schoellig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasiblity of Motion Primitives for Choreographed Quadrocopter Flight. In *American Control Conference*, 2011.

[24] M Zandvliet, O Bosgra, J Jansen, P Vandenhof, and J Kraaijevanger. Bang-Bang Control and Singular Arcs in Reservoir Flooding. *Journal of Petroleum Science and Engineering*, 58(1-2):186–200, August 2007.

Paper II

Real-Time Trajectory Generation for Quadrocopters

Markus Hehn and Raffaello D'Andrea

Abstract

This paper presents a trajectory generation algorithm that efficiently computes high-performance flight trajectories capable of moving a quadrocopter from a large class of initial states to a given target point that will be reached at rest. The approach consists of planning separate trajectories in each of the three translational degrees of freedom, and ensuring feasibility by deriving decoupled constraints for each degree of freedom through approximations that preserve feasibility. The presented algorithm can compute a feasible trajectory within tens of microseconds on a laptop computer; remaining computation time can be used to iteratively improve the trajectory. By replanning the trajectory at a high rate, the trajectory generator can be used as an implicit feedback law similar to model predictive control. The solutions generated by the algorithm are analyzed by comparing them to time-optimal motions, and experimental results validate the approach.

Under review for publication in *IEEE Transactions on Robotics.*

1. Introduction

Quadrocopters are popular aerial robotic platforms for applications where the ability to hover and move freely in three-dimensional space is important [39]. Trajectory generation remains a problem, however, as flight paths that are feasible under the complex dynamic and input constraints of the vehicles must be computed.

1.1 Goal & Motivation

In well-known, controlled, static environments, quadrocopters can be flown using pre-planned flight paths and feedback control to track these flight paths (see, for example, [20] and references therein). Many of the potential applications for aerial robots (such as inspection tasks, disaster coordination, and journalism), however, do not offer such environments: the task may change dynamically (e.g., a target point might be continuously updated based on incoming information); there may be significant disturbances in the actual flight path (e.g., a large wind gust might push the vehicle too far off course for the feedback control to recover efficiently); and knowledge of the environment may be inaccurate (e.g., the existence, position, size, and shape of obstacles may only become available mid-flight). When dynamic changes such as these are encountered, the original pre-planned flight path may become suboptimal or even infeasible and therefore impossible to execute.

If a system is to operate in dynamic environments or execute dynamically changing tasks, it must be able to quickly update the planned flight trajectory according to new information as it becomes available, and it must be able to do this while the vehicle is in motion. Such trajectory generation methods are often referred to as 'real-time' [31], 'online' [26], or 'reactive' [25] for their ability to accommodate changing constraints by re-planning almost instantaneously. Executing the initial control inputs of the continuously updated trajectory also forms an implicit feedback law that can be used to control the vehicle in a fashion similar to model predictive control [10]. The use of a trajectory generation algorithm in real-time settings results in several additional requirements when compared to the generation of pre-planned flight trajectories:

1) Large range of initial conditions: Pre-planning allows boundary conditions to be carefully designed, e.g. by limiting the planning to trajectories that start and end with the vehicle at rest. If trajectories are re-planned dynamically, it is necessary to account for the non-rest initial state of the vehicle even if a disturbance has caused it to significantly deviate from its planned flight path.

2) Computational complexity: Updating the planned trajectory at a high frequency and with little delay helps to improve reaction time, and it follows that the computation time of the trajectory generation algorithm should be as short as possible. With typical quadrotor position control loops running at rates on the order of 50 Hz to 100 Hz [28,30], computation times of a few milliseconds are desirable. In many scenarios, the use of higher-level path planning algorithms involve evaluating large numbers of potential trajectories (for example in rapidly-exploring random tree or probabilistic road map algo-

rithms, see e.g. [24]), also making short computation times desirable.

1.2 Related Work

The problem of quadrocopter trajectory generation has received significant attention in recent years, and a number of algorithms have been presented. Broadly speaking, a possible categorization of the algorithms is as follows:

A first group of algorithms can be considered as primarily geometric. The trajectory generation process consists of first generating a path in space from a class of path primitives, and thereafter parameterizing the generated path in time such that the dynamic constraints of the quadrocopter are enforced. Examples of such algorithms have been presented using path primitives such as lines [19], polynomials [8], or splines [5].

A second group of algorithms is based on the design of trajectories that minimize a derivative of the position trajectory (or combinations thereof). Because these derivatives can be related to the control input constraints of the quadrocopter through its differential flatness property, the feasibility of the trajectory depends on these derivatives. Examples of such methods include minimum snap trajectory generation [29] and the minimization of a weighted sum of derivatives [38]. More real-time-focused implementations are based on model predictive control methods with learnt linear dynamics [4] or linear dynamics based on a decoupling of the system [32]. An algorithm that provides particularly low computational complexity has been presented for the case where it is sufficient to consider constraints and verify the feasibility of the trajectory *a posteriori* [33].

A third group consists of algorithms that numerically solve an optimal control problem that directly considers the nonlinear dynamics of the quadrocopter. Optimal solutions are then found through the application of well-established optimal control methods such as Pontryagin's minimum principle [17] or numerical optimal control [40].

1.3 Contributions

This paper presents and analyzes a trajectory generation algorithm that allows the fast computation of high-performance flight trajectories to move the quadrocopter from a large class of initial states to a given target location at rest. The specific performance criterion considered herein is the duration of the flight maneuver until the vehicle reaches the target.

The algorithm is designed to be computationally efficient in order to provide a provably feasible trajectory quickly enough for its use in real-time settings. The design is further targeted at flight in relatively small spaces, where peak velocities remain sufficiently small that aerodynamic effects (such as drag) do not become dominant. It is also assumed that an underlying body rate control loop is available to accurately track commanded rotational rates and therefore allow the rotational rates to be modeled as control inputs.

In order to efficiently generate trajectories under these assumptions, a decoupling approach is used to reformulate the quadrocopter trajectory problem in the computation of time-optimal trajectories for acceleration- and jerk-limited triple integrators. The fact that time-optimal trajectories for input-affine systems are bang-singular is well known from optimal control theory [18], [6]. In the context of trajectory generation for robotic applications, the bang-singular structure is attractive because these trajectories often provide a relatively simple parameterization and the possibility of computing the corresponding parameters efficiently. Similar approaches of bang-singular trajectory generation have been demonstrated for robotic arms with many joints [26] and for generic trajectories where the derivatives of the position are constrained [12]. A similar decoupling approach with time-optimal bang-singular trajectories has been used for omnidirectional ground robots in the RoboCup competition [36].

Furthermore, the algorithm provides means to trade off computational complexity and the performance of generated trajectories: after a first trajectory has been computed, remaining computation time can be used to iteratively increase the performance (i.e., reduce the duration) of the trajectory. The computational effort of this iterative optimization can be determined *a priori*, but it is also possible to adapt to changing availability of computational resources: the iteration process can be aborted at any time, and the best trajectory generated up until that time can be used as a solution while still maintaining feasibility.

Preliminary results of the decoupling approach were presented in a previous conference paper [16]. This paper extends these results by: 1) introducing an iterative method for optimally choosing the decoupling parameters, 2) a rigorous way to account for a large class of initial conditions without violating the dynamic constraints of the vehicle, and 3) an evaluation of the computational requirements and performance of the algorithm.

1.4 Outline

The remainder of this paper is organized as follows: Section 2 introduces the model of the quadrocopter dynamics that is used throughout the paper. The trajectory generation problem is then formally presented in Section 3. Section 4 derives the feasibility conditions from the dynamic model and the decoupling used to simplify the trajectory generation problem while guaranteeing feasibility. Section 5 introduces the basic proposed trajectory generation algorithm that satisfies the trajectory generation problem. Section 6 presents the iterative optimization of the decoupling parameters to improve performance when more computation time is available. The use of the trajectory generation algorithm as an implicit feedback law is discussed in Section 7. Section 8 presents results on the computational performance of the method and characterizes the properties of planned trajectories in comparison to time-optimal trajectories. An experimental validation of the method is presented in Section 9, and conclusions as well as an outlook are given in Section 10.

2. Dynamic Model

The quadrocopter is described by six degrees of freedom: the translational position

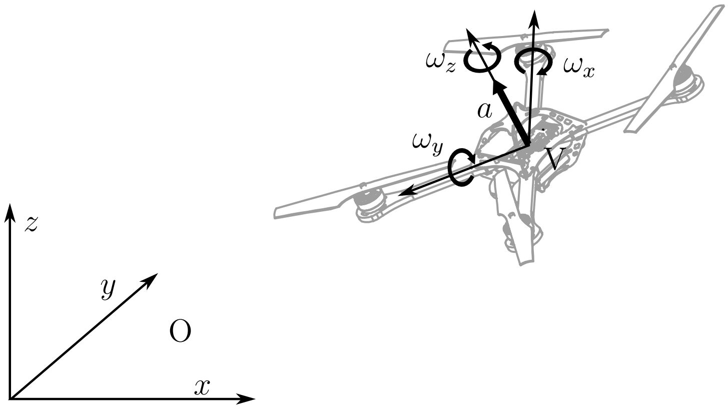

Figure 1. The inertial coordinate system O, the body-fixed coordinate system V, and the control inputs of the quadrocopter. The rotational rates ω_x, ω_y, and ω_z are tracked by an on-board controller, using gyroscope feedback. The collective mass-normalized thrust a acts along the third body axis of the vehicle.

$(x,\ y,\ z)$ is measured in the inertial coordinate system O as shown in Figure 1. The vehicle attitude V is defined by the rotation matrix ${}^{\mathrm{O}}_{\mathrm{V}}R$. The rotation matrix is defined such that, when multiplying it with a vector v in the vehicle coordinate system V, the same vector, described in the inertial coordinate system O, is obtained:

$$ {}^{\mathrm{O}}v = {}^{\mathrm{O}}_{\mathrm{V}}R\,{}^{\mathrm{V}}v. \tag{1}$$

2.1 Control Inputs

The control inputs of the quadrocopter are the rotational rates about the vehicle body axes, ω_x, ω_y, and ω_z, and the mass-normalized collective thrust, a, as shown in Figure 1.

It is assumed that high-bandwidth controllers on the vehicle track the rotational rates (this is usually achieved using feedback from gyroscopes). Typical quadrocopters have very low rotational inertia, and can produce high torques due to the outward mounting of the propellers [23]. This results in very high achievable rotational accelerations $\dot{\omega}_x$ and $\dot{\omega}_y$; the rotational rate control loops can therefore achieve very fast response times to changes in the commanded rotational rates[5]. In the following, it is therefore assumed that the vehicle body rates are directly controllable. Rotational accelerations $\dot{\omega}_z$ are created by causing a drag difference between propellers rotating in opposite directions, and achievable values are typically significantly lower. However, it will be shown that ω_z does not greatly influence the translational dynamics of the vehicle in this trajectory generation problem.

The four control inputs $(\omega_x, \omega_y, \omega_z, a)$ are subject to saturation. The magnitude of the vehicle body rates are limited (such limitations can be caused, for example, by the range of the gyroscopes used for feedback, or performance limitations of the body rate

---

[5] Experiments by the authors with vehicles of approximately 0.5 kg have shown rotational accelerations on the order of $200\,\mathrm{rad\,s^{-2}}$ and body rate tracking time constants on the order of 20 ms.

tracking controllers):

$$|\omega_x| \leq \omega_{xy,\mathrm{max}} \tag{2}$$
$$|\omega_y| \leq \omega_{xy,\mathrm{max}} \tag{3}$$
$$|\omega_z| \leq \omega_{z,\mathrm{max}} \,. \tag{4}$$

The collective thrust is limited by a minimum and a maximum thrust

$$a_{\mathrm{min}} \leq a \leq a_{\mathrm{max}} \tag{5}$$

where $a_{\mathrm{min}} > 0$. This limitation is motivated by the fact that typical quadrocopters have fixed-pitch propellers, the direction of rotation of which cannot be reversed during flight.

2.2 Equations of Motion

The translational acceleration of the vehicle is dictated by the attitude of the vehicle and the total thrust produced by the four propellers a. The translational acceleration in the inertial frame is

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} \,. \tag{6}$$

The change of vehicle attitude is related to the rotational control inputs through the kinematic relationship [22]

$${}^{\mathrm{O}}_{\mathrm{V}}\dot{R} = {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \,. \tag{7}$$

Note that the above equations of motion neglect well-known aerodynamic effects that act on quadrocopters, such as rotor damping [34] and drag-like effects [1]. While such effects can be significant at high flight speeds, the algorithm presented herein is mainly intended for navigating relatively small spaces where flight speeds are limited. Omitting these effects eliminates the need to quantify them over a large range of flight regimes and also greatly simplifies the problem. This motivates their omission in this paper. These simplifying assumptions can be considered similar to model reduction approaches for model predictive control (see e.g. [14]) in that the model accuracy is traded off for computational performance.

3. Problem Statement

The trajectory generation problem considered herein can be described as follows: Let the quadrocopter have a given initial state (consisting of initial vehicle position x_0, y_0, z_0, velocity $\dot{x}_0, \dot{y}_0, \dot{z}_0$, and attitude ${}^{\mathrm{O}}_{\mathrm{V}}R_0$). Generate a trajectory that satisfies the initial state constraints and drives the vehicle to a target point (taken to be, without loss of generality, the origin), with the vehicle reaching the target point at rest. The generated trajectory must be feasible with respect to the quadrotor dynamics (6)-(7) and the control input constraints (2)-(5). It should also reach the target point as quickly as possible, and its generation should be computationally inexpensive.

4. Feasibility Conditions and Decoupling

This section describes how the trajectory generation problem statement from the previous section is converted into three separate, more tractable trajectory generation problems.

The derivation of the control inputs for an arbitrary vehicle trajectory $(x(t),\ y(t),\ z(t))$ is presented first (Section 4.1). Using this derivation, the general conditions under which trajectories are feasible are then obtained (Section 4.2). From these general trajectory constraints, a set of decoupled trajectory constraints are then generated through approximations that preserve feasibility. These decoupled constraints differ from the general trajectory constraints in that the motion constraints of one degree of freedom (e.g., $x(t)$) do not depend on the other two degrees of freedom (e.g., $y(t)$ and $z(t)$).

4.1 Control Inputs for a Given Trajectory

Let $(x(t),\ y(t),\ z(t))$ denote a candidate vehicle trajectory. For notational convenience, the time dependency will hereafter be omitted unless specific times are considered. Taking the second derivative of the trajectory and combining it with the translational equation of motion (6), the vector f is defined to represent the total mass-normalized forces required by the quadrocopter to follow the trajectory:

$$f := \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathrm{g} \end{bmatrix} = {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} . \tag{8}$$

Using the two-norm (denoted by $\|\cdot\|$), the thrust a required to follow the trajectory can be calculated by using the property that a pure rotation matrix does not change the two-norm of a vector [2]:

$$\|f\| = \left\| {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} \right\| = a\,. \tag{9}$$

The direction of thrust is the unit vector in the direction of f

$$\bar{f} := \frac{f}{\|f\|} \tag{10}$$

and can be seen to define the third column of the rotation matrix by substituting $\bar{f}$ back into Equation (8):

$${}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \bar{f}\,. \tag{11}$$

Taking the derivative of the above equation and combining it with Equation (7) gives two of the vehicle body rates as functions of the current attitude and $\dot{\bar{f}}$:

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = {}^{\mathrm{V}}_{\mathrm{O}}R\dot{\bar{f}}\ . \tag{12}$$

Equation (9) and the first two rows of Equation (12) provide three equations for four unknown control inputs. To fully constrain the control input trajectories for the given position trajectory, a fourth constraint must be additionally specified, which can be taken to be a user-defined constraint on ω_z ($\omega_z = 0$, for example).

4.2 Coupled Feasibility Conditions

Feasibility constraints for trajectories can now be calculated from the initial state and control input constraints:

Collective Thrust The collective thrust calculated from Equation (9) must lie between the minimum and maximum thrust defined by Equation (5), i.e.

$$\|f\| = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + \mathrm{g})^2} \geq a_{\min} \tag{13}$$

$$\|f\| \leq a_{\max}\,. \tag{14}$$

Rotational Rates The actual body rate control inputs ω_x and ω_y can only be computed if a constraint on ω_z is specified in addition to the trajectory. However, they can be bounded using the unit norm property of the rotation matrix [2] with Equation (12):

$$\omega_{x,y} \leq \left\| \dot{\bar{f}} \right\| \tag{15}$$

from which it follows that a trajectory is guaranteed to be feasible if

$$\left\| \dot{\bar{f}} \right\| \leq \omega_{xy,\max} \ . \tag{16}$$

Note that the above constraint is independent of the specification of ω_z, and therefore holds for a translational trajectory irrespective of the rotational motion defined through ω_z.

Initial State The trajectory must satisfy the constraints arising from the initial state of the vehicle. Specifically, the position and velocity must coincide with the initial values

$$x(t=0) = x_0, \;\; y(t=0) = y_0, \;\; z(t=0) = z_0 \tag{17}$$

$$\dot{x}(t=0) = \dot{x}_0, \;\; \dot{y}(t=0) = \dot{y}_0, \;\; \dot{z}(t=0) = \dot{z}_0 \tag{18}$$

and the acceleration at the start of the trajectory must be such that Equation (11) is satisfied with the given attitude, that is

$${}^{\mathrm{O}}_{\mathrm{V}}R_0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \bar{f}(t=0) \, . \tag{19}$$

4.3 Decoupling of the Feasibility Conditions

To simplify the three-dimensional planning problem, it is desirable to perform the planning separately for each of the coordinates x, y, and z. This requires the definition of independent motion constraints for each degree of freedom. In terms of trajectory feasibility, the three axes are coupled in the acceleration constraints (13)-(14), in the mixed acceleration and jerk constraint (16), and in the initial attitude constraint (19). By conservatively approximating these constraints, it is possible to reformulate the trajectory generation problem such that the three degrees of freedom are entirely decoupled, implying that the overall trajectory will be guaranteed to be feasible if the three degrees of freedom satisfy their respective independent constraints.

Minimum Thrust Constraint In order to satisfy the minimum thrust constraint (13), the vertical acceleration is constrained to be

$$\ddot{z} \geq \ddot{z}_{\text{min}} \tag{20}$$

where the minimum permissible vertical acceleration $\ddot{z}_{\text{min}}$ must satisfy

$$\ddot{z}_{\text{min}} \geq a_{\text{min}} - \mathrm{g} \ . \tag{21}$$

It is then straightforward to verify that Equation (13) holds independently of the trajectories in x and y because the thrust is no smaller than a_{min}. While, at this point, it may seem beneficial to choose the smallest possible value of $\ddot{z}_{\text{min}}$, the tradeoffs involved in the choice of this parameter will become obvious in the following section.

Rotational Rate Constraint The rotational rate control inputs nonlinearly couple the jerk and acceleration of the three degrees of freedom according to Equation (12), and their limitation (2)-(3) therefore couples the permissible values of the trajectory jerks and accelerations.

The trajectory jerk $\dot{f}$ is related to the actual control inputs through (12) by explicitly calculating $\dot{\bar{f}}$:

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = {}^{\text{V}}_{\text{O}}R \left(\frac{\dot{f}}{\|f\|} - \frac{f f^T \dot{f}}{\|f\|^3} \right) \ . \tag{22}$$

Using the triangle inequality and the Cauchy-Schwarz inequality [2], the above expression is at most

$$\omega_{x,y} \leq \left\| \frac{\dot{f}}{\|f\|} - \frac{f f^T \dot{f}}{\|f\|^3} \right\| \tag{23}$$

$$\omega_{x,y} \leq \frac{\left\|\dot{f}\right\|}{\|f\|} + \frac{\left\|f f^T \dot{f}\right\|}{\|f\|^3} \tag{24}$$

$$\omega_{x,y} \leq \frac{\left\|\dot{f}\right\|}{\|f\|} + \frac{\left\|f\right\|^2 \left\|\dot{f}\right\|}{\|f\|^3} \tag{25}$$

$$\omega_{x,y} \leq 2\frac{\left\|\dot{f}\right\|}{\|f\|}. \tag{26}$$

Because the minimum vertical acceleration constraint (20) provides a lower bound for

$\|f\|$, feasibility with respect to the body rate input constraints (2)-(3) is guaranteed if

$$\left\|\dot{f}\right\| \leq \frac{(\ddot{z}_{\min} + \mathrm{g})}{2} \omega_{xy,\max} \tag{27}$$

holds. For the decoupled system, this can be ensured by limiting the permissible per-axis jerk of a planned trajectory by

$$|\dddot{x}| \leq \dddot{x}_{\max}, \ |\dddot{y}| \leq \dddot{y}_{\max}, \ |\dddot{z}| \leq \dddot{z}_{\max} \tag{28}$$

and choosing the limiting values such that Equation (27) is satisfied:

$$\dddot{x}^2_{\max} + \dddot{y}^2_{\max} + \dddot{z}^2_{\max} \leq \frac{(\ddot{z}_{\min} + \mathrm{g})^2}{4} \omega^2_{xy,\max} \, . \tag{29}$$

It can now be seen that the lower bound of the vertical acceleration ($\ddot{z}_{\min}$) can be used as a design parameter in order to trade off higher allowable jerk magnitudes against achievable negative accelerations in the vertical (Equations (20) and (29)).

For the purpose of this paper, the jerk limit is chosen to be identical for the three degrees of freedom, i.e.

$$\dddot{w}_{\max} := \dddot{x}_{\max} = \dddot{y}_{\max} = \dddot{z}_{\max} = \frac{(\ddot{z}_{\min} + \mathrm{g})}{\sqrt{12}} \omega_{xy,\max} \, . \tag{30}$$

Maximum Thrust Constraint The maximum allowable thrust (14) imposes a constraint on the two-norm of the accelerations in the three degrees of freedom. This constraint is satisfied if

$$|\ddot{x}| \leq \ddot{x}_{\max}, \ |\ddot{y}| \leq \ddot{y}_{\max}, \ \ddot{z} \leq \ddot{z}_{\max} \tag{31}$$

where $\ddot{z}$ is already lower-bounded by Equation (20). The acceleration bounds of the three degrees of freedom are parameterized as functions of the maximum thrust constraint (5) using the maximum acceleration tradeoff parameters $\alpha_x, \alpha_z \in (0 \ 1)$:

$$\alpha_z := \frac{\ddot{z}_{\max}}{(a_{\max} - \mathrm{g})} \tag{32}$$

$$\alpha_x := \frac{\ddot{x}_{\max}}{\sqrt{a^2_{\max} - (\ddot{z}_{\max} + \mathrm{g})^2}} \, . \tag{33}$$

The two parameters specify the maximum accelerations $\ddot{x}_{\max}$ and $\ddot{z}_{\max}$. The permissible acceleration $\ddot{y}_{\max}$ is then chosen to be as large as possible while fulfilling constraint (14):

$$\ddot{y}_{\max} = \sqrt{a_{\max}^2 - \ddot{x}_{\max}^2 - (\ddot{z}_{\max} + \mathrm{g})^2}\,. \tag{34}$$

If the constraints (31) defined through Equations (32)-(34) with $\alpha_x, \alpha_z \in (0\ 1)$ are satisfied, then the maximum thrust constraint (14) is satisfied by design.

Initial Acceleration To decouple the initial acceleration constraint (19), it is replaced by individual initial acceleration constraints for the three degrees of freedom. The initial accelerations $\ddot{x}_0, \ddot{y}_0, \ddot{z}_0$ are defined through the initial attitude ${}^{\mathrm{O}}_{\mathrm{V}}R_0$ using Equation (6):

$$\begin{bmatrix} \ddot{x}_0 \\ \ddot{y}_0 \\ \ddot{z}_0 \end{bmatrix} := {}^{\mathrm{O}}_{\mathrm{V}}R_0 \begin{bmatrix} 0 \\ 0 \\ a_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} \tag{35}$$

with the initial thrust control input a_0 remaining a free degree of freedom. For applications where the trajectory generation algorithm is applied recursively (e.g. when it is used as a feedback law), the last value of the thrust a is typically available and can readily be used to compute the initial acceleration; in other usage scenarios, the initial value a_0 may be considered a design variable.

It is assumed that the initial state $\ddot{z}_0$ defined by Equation (35) satisfies the minimum acceleration constraint (20), i.e. the initial acceleration must satisfy $\ddot{z}_0 \geq \ddot{z}_{\min}$ for the chosen value a_0. This condition limits the range of permissible initial attitudes as can be seen in Equation (35).

Because the allowable acceleration magnitudes $(\ddot{x}_{\max}, \ddot{y}_{\max}, \ddot{z}_{\max})$ can be traded off freely through Equations (32) and (33), the initial accelerations $(\ddot{x}_0, \ddot{y}_0, \ddot{z}_0)$ may not satisfy the constraints (31). For nonzero initial accelerations, the acceleration constraints (31) are therefore replaced by the time-varying constraints

$$|\ddot{x}| \leq \overline{\ddot{x}}(t), \;\; |\ddot{y}| \leq \overline{\ddot{y}}(t), \;\; \ddot{z} \leq \overline{\ddot{z}}(t). \tag{36}$$

These time-varying constraints are constructed as follows, here explained for the example of the first degree of freedom: The maximum allowable acceleration magnitude $\overline{\ddot{x}}(t)$ starts at the magnitude of the initial acceleration (e.g. $|\ddot{x}_0|$). It then has constant slope (e.g. c_x) until it reaches the design value (e.g. $\ddot{x}_{\max}$):

$$\overline{\ddot{x}}(t) = \begin{cases} |\ddot{x}_0| + c_x t & \text{for } 0 \leq t < (\ddot{x}_{\max} - |\ddot{x}_0|)/c_x \\ \ddot{x}_{\max} & \text{for } t \geq (\ddot{x}_{\max} - |\ddot{x}_0|)/c_x\,. \end{cases} \tag{37}$$

It now remains to choose the slopes c_x, c_y, c_z such that the maximum thrust constraint (14) remains satisfied for all times given the initial acceleration $(\ddot{x}_0, \ddot{y}_0, \ddot{z}_0)$. This is done by differentiating between degrees of freedom where the constraints are increasing (i.e., $c_x > 0$) and those where they are decreasing ($c_x < 0$).

For degrees of freedom where the initial acceleration exceeds the allowable acceleration (e.g. $|\ddot{x}_0| > \ddot{x}_{\max}$), the slope is chosen to be the maximum jerk $c_x = -\dddot{w}_{\max}$ as given in Equation (30). This corresponds to the fastest possible transition into the designed maximum magnitude that is permissible under the jerk constraint (28).

Degrees of freedom where the initial acceleration is smaller than the allowable acceleration (e.g. $|\ddot{x}_0| < \ddot{x}_{\max}$) cannot increase their acceleration at the maximum permissible jerk value without violating the maximum thrust constraint (14). Intuitively speaking, they must 'wait' for other degrees of freedom to reduce their respective acceleration before using the full acceleration. Given that the constraints are always reduced in magnitude at the maximum rate, the longest duration for any degree of freedom to reduce to zero acceleration is

$$\Delta T_0 := \max\left(\frac{|\ddot{x}_0|}{\dddot{w}_{\max}}, \frac{|\ddot{y}_0|}{\dddot{w}_{\max}}, \frac{|\ddot{z}_0|}{\dddot{w}_{\max}}\right) . \tag{38}$$

The slope for acceleration bounds exceeding the initial acceleration magnitude are then chosen to reach their allowable acceleration after ΔT_0, given here for the example of the x degree of freedom:

$$c_x = \frac{\ddot{x}_{\max} - |\ddot{x}_0|}{\Delta T_0} . \tag{39}$$

The proof that this choice of slopes c_x, c_y, c_z satisfies the maximum thrust constraint (14) is omitted here but may be found in Appendix A.

Overview This completes the decoupling of the system. In conclusion, if the following conditions are satisfied:

1. the initial attitude is converted to an initial acceleration according to Equation (35) and satisfies the minimum vertical acceleration constraint (20);
2. the jerk in each axis is constrained in magnitude according to Equation (28); and
3. the acceleration of the individual degrees of freedom is constrained by Equations (20) and (36);

then the three-dimensional overall trajectory will be feasible with respect to the system dynamics (6)-(7), the input constraints (2)-(5), and the initial conditions. It is then possible to design the trajectory of the three translational degrees of freedom independently.

The parameters used in the decoupling of the three degrees of freedom are the maximum acceleration tradeoff parameters α_x and α_z, the minimum vertical acceleration $\ddot{z}_{\min}$, and the initial thrust a_0.

5. Decoupled Trajectory Planning

The basic trajectory generation algorithm will be introduced in this chapter. The decoupled feasibility constraints (Section 4.3) allow the planning to be carried out independently for each degree of freedom while guaranteeing feasibility. This makes the trajectory generation problem significantly more tractable and allows it to be solved in a computationally efficient way.

5.1 Time-Optimal Trajectories of the Decoupled System

Each degree of freedom is represented by a triple integrator under acceleration and jerk constraints (the only difference is that the acceleration constraints for x and y are symmetric, while they may be asymmetric for z). As the planning is identical for all coordinates, it is presented here for a general degree of freedom w. In order to achieve fast motion, we plan time-optimal trajectories for each axis.

Problem Statement Let $s = (s_1,\ s_2,\ s_3) = (w,\ \dot{w},\ \ddot{w})$ be the state. The time-optimal planning problem can then be stated as: find the planning input $u = \dddot{w}$ minimizing the final time $t_{f,w}$:

$$u^* = \arg\min_{u} t_{f,w} \tag{40}$$

subject to the system dynamics

$$\dot{s}_1 = s_2 \tag{41}$$
$$\dot{s}_2 = s_3 \tag{42}$$
$$\dot{s}_3 = u, \tag{43}$$

the initial and final conditions

$$s_1(t = 0) = w_0 \tag{44}$$
$$s_2(t = 0) = \dot{w}_0 \tag{45}$$
$$s_3(t = 0) = \ddot{w}_0 \tag{46}$$
$$s(t = t_{f,w}) = 0 \tag{47}$$

and the state and input constraints

$$\underline{\ddot{w}} \leq s_3 \leq \overline{\ddot{w}} \tag{48}$$
$$|u| \leq \dddot{w}_{\max} \ . \tag{49}$$

In this formulation, the initial vehicle attitude (35) is enforced by the initial condition (46), the jerk constraint (28) by the input constraint (49), and the acceleration constraints (20) and (36) by the state constraint (48). Because the translational dynamics (6) contain no velocity-dependent aerodynamic effects, no velocity constraints are considered.

Necessary Optimality Conditions Using Pontryagin's minimum principle (see, for example, [3]), necessary conditions for optimal input trajectories will now be derived. The methodology used to handle the state constraint (48) is the *direct adjoining approach* [15], in which the augmented Hamiltonian function is defined by

$$\begin{aligned} H(s,u,\lambda,\eta) = {} & \lambda_1 s_2 + \lambda_2 s_3 + \lambda_3 u \\ & + \eta_1 \left(s_3 - \underline{\ddot{w}}\right) + \eta_2 \left(\overline{\ddot{w}} - s_3\right) \end{aligned} \tag{50}$$

where λ are the adjoint variables and η are state constraint multipliers that fulfill

$$\eta \geq 0 \tag{51}$$

$$\eta_1 = 0 \text{ if } s_3 > \underline{\ddot{w}} \tag{52}$$

$$\eta_2 = 0 \text{ if } s_3 < \overline{\ddot{w}} \ . \tag{53}$$

The adjoint variables must fulfill

$$\dot{\lambda} = -\nabla_s H(s,u,\lambda,\eta) \tag{54}$$

which results in

$$\dot{\lambda}_1 = 0 \tag{55}$$

$$\dot{\lambda}_2 = \lambda_1 \tag{56}$$

$$\dot{\lambda}_3 = \lambda_2 + \eta_1 - \eta_2 \ . \tag{57}$$

The optimal control u^* is the control input that minimizes the Hamiltonian function:

$$\begin{aligned} u^* &= \arg\min H(s,u,\lambda,\eta) && (58) \\ &= \arg\min \lambda_3 u \ . && (59) \end{aligned}$$

For this problem structure, it can be shown that the adjoint variables λ are continuous [15]. Furthermore, $\lambda_3 = 0$ must hold when a state constraint is active. The optimal control input u^* consists of *interior arcs* and *boundary arcs*. An interior arc is characterized by $\eta_1 = \eta_2 = 0$ (i.e., the state constraint (48) is not active), $\lambda_3 \neq 0$ and therefore

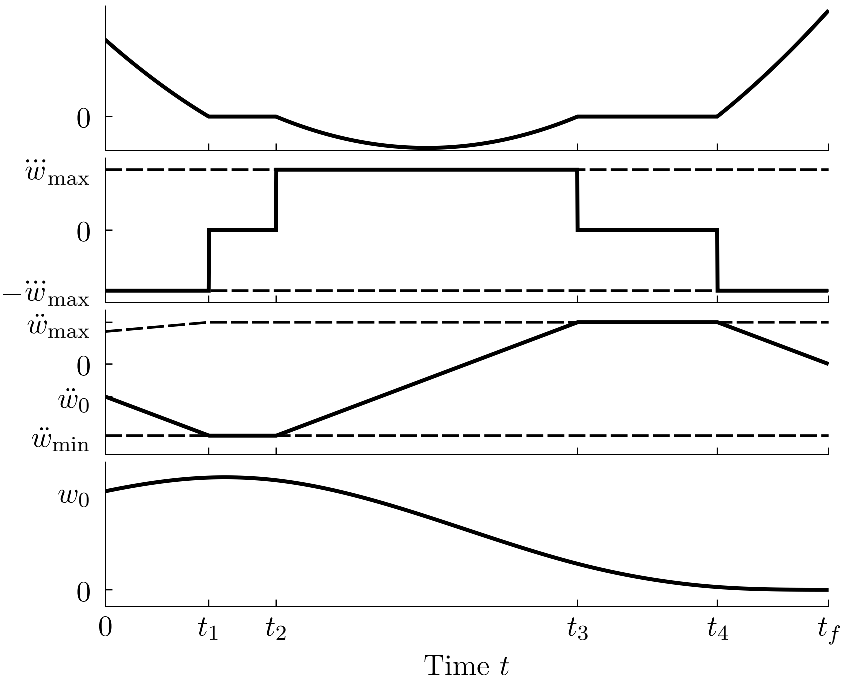

Figure 2. Example of a solution to the one-dimensional planning problem introduced in Section 5.1. From top to bottom, the plots show the trajectories of: 1) the third adjoint variable λ_3; 2) the optimal control input u^* (solid) along with the maximum allowable input magnitude $\dddot{w}_{\max}$ (dashed); 3) the acceleration trajectory $\ddot{w}^*$ (solid) along with the upper and lower acceleration bounds $\underline{\ddot{w}}$, $\overline{\ddot{w}}$ (dashed); and 4) the position trajectory w^*. Note that this sketch shows asymmetrical acceleration bounds $\underline{\ddot{w}}$, $\overline{\ddot{w}}$, as used for the vertical degree of freedom.

$u^* = \pm\dddot{w}_{\max}$. On boundary arcs, the state constraint is active (i.e., $s_3 = \underline{\ddot{w}}$ or $s_3 = \overline{\ddot{w}}$) and it follows that $\lambda_3 = 0$ and u^* is chosen such that the state constraint remains active.

Structure of Time-Optimal Trajectories Using the properties of interior arcs and boundary arcs, it is straightforward to characterize the solution of the costate dynamics (57) further: During an interior arc, λ_3 is a second order polynomial since $\dddot{\lambda}_3 = \dot{\lambda}_1 = 0$. Since λ_3 is continuous, a boundary arc can only begin at a point where $\lambda_3 = 0$ holds, and the solution of λ_3 is then given by the constraint that $\lambda_3 = 0$ must hold for the duration of the boundary arc.

It can be verified from the above constraints that the trajectory consists of at most five sections:

- $[0\; t_1]$ is an interior arc, with $u^* = \pm\dddot{w}_{\max}$.
- $[t_1\; t_2]$ is a boundary arc, with u^* such that $\ddot{w} = \overline{\ddot{w}}$ or $\ddot{w} = \underline{\ddot{w}}$.
- $[t_2\; t_3]$ is an interior arc, with $u^* = \mp\dddot{w}_{\max}$.
- $[t_3\; t_4]$ is a boundary arc, with u^* such that $\ddot{w} = \overline{\ddot{w}}$ or $\ddot{w} = \underline{\ddot{w}}$.
- $[t_4\; t_f]$ is an interior arc, with $u^* = \pm\dddot{w}_{\max}$.

Furthermore, each boundary arc induces one additional constraint, in that one or the other of the following conditions must hold:

- If the duration of the boundary arc is nonzero ($t_2 - t_1 > 0$ or $t_4 - t_3 > 0$, respectively), then s_3 must coincide with the upper or lower acceleration constraint at the start point of the boundary arc (t_1 or t_3).
- If the above condition does not hold, the corresponding boundary arc must vanish ($t_2 = t_1$ or $t_4 = t_3$, respectively).

In the following, the solution to the optimal control problem (40)-(49) will be denoted by $w^*(t)$. An example of the solution of the optimal control problem is illustrated in Figure 2. This example depicts the case where all five sections are of non-zero duration.

5.2 Computation of Solutions

The control trajectory $u^*(t)$ (and therefore $w^*(t)$) is fully specified by the five times t_1, t_2, t_3, t_4, and t_f and the initial control input. The trajectory is constrained by the three terminal state conditions (47) and the two boundary arc constraints. Analytically integrating the equations of motion (41)–(43) is straightforward and results in three equations of first, second, and third order in the switch times $t_1 \ldots t_f$, respectively. The two boundary arc constraints yield two additional first order equations in the switch times.

The solution is computed by applying a bisection algorithm [7] to the final time t_f. For a given value of t_f, it is straightforward to compute the remaining times $t_1 \ldots t_4$ and thereby the resulting final position $w(t_f)$, which is a strict monotonic function of t_f for a given initial control input [6].

5.3 Overview

The basic trajectory generation algorithm has herewith been described. The total three-dimensional maneuver is given by

$$x(t) = \begin{cases} x^*(t) & \text{for } 0 \leq t \leq t_{f,x} \\ 0 & \text{for } t > t_{f,x} \end{cases} \tag{60}$$

$$y(t) = \begin{cases} y^*(t) & \text{for } 0 \leq t \leq t_{f,y} \\ 0 & \text{for } t > t_{f,y} \end{cases} \tag{61}$$

$$z(t) = \begin{cases} z^*(t) & \text{for } 0 \leq t \leq t_{f,z} \\ 0 & \text{for } t > t_{f,z}\,. \end{cases} \tag{62}$$

The maneuver satisfies the initial conditions, the quadrocopter dynamics, and the input constraints. It ends at rest at the origin at time

$$t_f := \max(t_{f,x}, t_{f,y}, t_{f,z})\,. \tag{63}$$

It is therefore a valid solution to the trajectory generation problem stated in Section 3.

6. Choice of Design Parameters

The algorithm presented in the previous chapter computes a feasible trajectory that satisfies the trajectory planning problem. However, the decoupling approach involved the choice of the design parameters $\alpha_x, \alpha_z, \ddot{z}_{\min}, a_0$ in order to allow the separate planning in each degree of freedom. This chapter presents and discusses the properties and influence of the individual parameters, and proposes possible iterative improvement schemes that permit the computation of better trajectories[6] through their variation.

For any choice of the decoupling parameters satisfying the conditions outlined in Section 4.3, the solution of the decoupled trajectory planning (60)-(62) is a feasible solution to the trajectory generation problem. It is therefore possible to apply the iterative improvement schemes to only a subset of parameters, or to not apply them at all.

A consequence of this property is that, as long as the basic trajectory generation algorithm from the previous chapter has been executed at least once, a feasible solution is available at any time, and further executions then improve on previous ones. This implies that computational constraints can always be abided by aborting computation when the permissible computation duration is reached. If the computation time is not sufficient for the iteration schemes to converge to a user-defined level of accuracy, then the available solution differs from the converged solution in that the maneuver duration may be longer, but is still guaranteed to be feasible with respect to the dynamics and constraints of the problem.

6.1 Acceleration Tradeoff Parameters

The acceleration tradeoff parameters α_x and α_z control how much of the vehicle thrust capability is allocated to each of the three degrees of freedom according to the respective acceleration bounds (48).

To plan a three-dimensional trajectory, the time-optimal trajectory for each of the three decoupled systems is computed. The execution times of the three degrees of freedom will generally depend on the choice of the respective acceleration bounds (48); the maneuver durations $t_{f,x}, t_{f,y}, t_{f,z}$ are likely to differ from each other. To improve overall performance, the allowable per-axis acceleration may be varied such that, if one degree

---

[6] Because the objective of the trajectory generation is the minimization of the maneuver duration, trajectories are considered better if they are of shorter duration. The descriptions “shorter” and “better” are therefore used interchangeably.

of freedom has a longer execution time than the others, it is allocated more acceleration in order to reduce the execution time.

An important property of the decoupling parameters α_x and α_z is that the duration of the time-optimal maneuver of each single degree of freedom depends on them monotonically. This can be shown as follows:

The acceleration boundaries $(\ddot{x}_{\max}, \ddot{y}_{\max}, \ddot{z}_{\max})$ are strictly monotonic with respect to the control effort tradeoff parameters (32)-(33). By construction, the time-varying acceleration constraints (37) monotonically depend on the respective acceleration boundary $\ddot{w}_{\max}$. That is, if $\ddot{w}_{\max,1} \geq \ddot{w}_{\max,2}$ then

$$\overline{\ddot{w}}_1(t) \geq \overline{\ddot{w}}_2(t) \text{ for all } t \geq 0\,. \tag{64}$$

Because the computed trajectories are optimal with respect to the maneuver duration, an increase of the constraints cannot increase the terminal time of the maneuver [3] and the maneuver duration is therefore monotonic with respect to the decoupling parameters α_x and α_z.

From this property, it follows that a maneuver synchronizing the maneuver durations of the three degrees of freedom minimizes t_f. This can be shown as follows: Assume that the three maneuver durations are synchronized. Because the maneuver durations of the three degrees of freedom are monotonic with respect to the control effort tradeoff parameters, no change of the parameters can reduce all three maneuver durations (and thereby the total maneuver duration t_f as given by Equation (63)).

The monotonicity of the maneuver durations with respect to the optimization parameters allows the use of a large number of optimization algorithms to find their optimal value. A straightforward implementation consists of two loops: an inner loop that synchronizes the two horizontal degrees of freedom (i.e., $t_{f,x}$ and $t_{f,y}$) by varying α_x, and an outer loop that synchronizes the vertical motion (i.e., $t_{f,z}$) to the two horizontal motions by varying α_z. Each of these loops can be implemented in a simple manner by a bisection algorithm, which assures linear convergence. Specifically, this implies that the complexity of the two nested bisections finding the optimal values of α_x and α_z to within a tolerance ϵ_{xz} is [7]

$$\mathcal{O}\left(\left(\log_2\left(\epsilon_{xz}\right)\right)^2\right)\,. \tag{65}$$

In addition, the computation of the inner loop can be aborted early if it is found that the maneuver duration of both horizontal degrees of freedom is either shorter or longer than the duration of the vertical degree of freedom. Due to the maneuver duration monotonicity with respect to α_x, it is not possible to increase or decrease the maneuver duration of both horizontal degrees of freedom, and synchronization with the vertical degree of freedom is therefore not possible. By using this property, it is not necessary to perform the bisection of α_x to full accuracy for each bisection step in α_z.

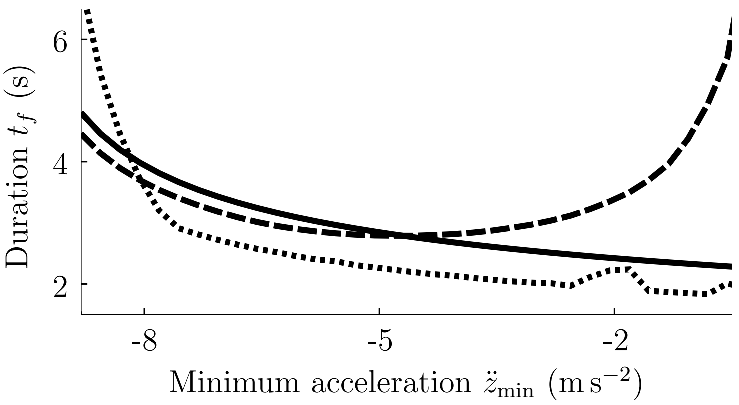

Figure 3. Influence of the design parameter $\ddot{z}_{\min}$ on the maneuver duration t_f with optimized tradeoff parameters α_x, α_z for three different maneuvers (solid, dashed, and dotted). The optimization presented in Section 6.2 finds the minimum of this function through an exhaustive search.

The two nested bisection algorithms provide a straightforward way to compute the optimal values of the two control tradeoff parameters α_x and α_z. Furthermore, the maneuver duration monotonicity and linear convergence of the bisection method allow the *a priori* determination of hard constraints on the number of required executions. However, more sophisticated and multivariate optimization methods (see e.g. [11]) could provide higher computational efficiency in the finding of the optimal values.

6.2 Minimum Vertical Acceleration

The minimum vertical acceleration $\ddot{z}_{\min}$ directly affects the acceleration constraint (48) for the decoupled motion planning problem of the vertical degree of freedom. Through Equation (30), it also influences the allowable control input (49) for all three degrees of freedom. Figure 3 shows the maneuver duration (after optimization of α_x and α_z to a tolerance of $\epsilon_{xz} = 10^{-3}$ as described in the previous section) over varying values of $\ddot{z}_{\min}$ for three different maneuvers. It can be seen that the function may have multiple local minima. This complicates the application of standard optimization methods.

In order to find the global minimum maneuver time during the on-line optimization of the minimum vertical acceleration, it is necessary to use an optimization algorithm that is not sensitive to local minima. A straightforward way to achieve this is to grid the search space at a user-defined grid size, and evaluate the maneuver duration at each point. While such a naive approach would be computationally prohibitive for multivariate optimizations, it can be performed to a satisfactory tolerance in the case of only one optimization variable.

Like in the optimization of the acceleration tradeoff parameters, the use of more sophisticated optimization algorithms could further reduce the computational complexity. While it may be difficult to guarantee the convergence to the global optimum in all cases, it may be acceptable to provide only local convergence since feasibility remains guaranteed.

6.3 Overview

This chapter presented schemes for finding the optimal values of the design parameters $\alpha_x, \alpha_z, \ddot{z}_{\min}$ such that the maneuver duration is minimized. The application of these schemes is optional since the feasibility of the trajectory generation algorithm does not depend on them. Depending on the computational constraints and performance requirements, one may choose to optimize all design parameters, or to fix some or all of them ahead of time.

The schemes presented herein are focused on simplicity and assured convergence, and optimize each of the design parameters individually in nested loops. Through the use of more advanced optimization algorithms, and by directly considering the multivariate optimization problem, the computation time may be reduced further.

The decoupling process also involved the choice of the initial thrust a_0 in Equation (35). While this can be considered a design parameter and therefore optimized, it is assumed herein that it is chosen based on the thrust applied previously. This results in continuous accelerations and therefore continuous thrust commands (as computed by Equation (9)), a favourable property with respect to the underlying thrust dynamics.

7. Use as a Feedback Law

By solving the trajectory generation problem repeatedly at a high frequency, it is possible to use the trajectory generation algorithm as an implicit feedback law. The control law then consists of updating the initial conditions according to an updated state estimate (typically obtained using new measurements), re-planning the trajectory to the target point, and applying the control inputs of the first control interval (that is, for the duration until the trajectory is re-planned). This method is very similar to model predictive control (see, for example, [10] for an overview) in that an updated optimal trajectory is generated at each time step.

To apply the feedback law, it is necessary to compute the control inputs from the planned trajectory. Whereas the control inputs are usually assumed piecewise constant in model predictive control, the trajectory planned here typically results in continuously varying control inputs (the trajectory jerk is piecewise constant, but the control inputs vary over time). For any given time, the thrust input a can be computed according to Equation (9), and the body rates ω_x, ω_y can be computed by numerically integrating the rotation matrix derivative (7) with the intermediate control inputs computed through Equation (12). In addition, the specification of the body rate ω_z is required, which is not defined by the planned translational trajectory (see Section 4.2). Because the trajectory is feasible for arbitrary choices of ω_z and the control inputs can be computed under consideration of any given trajectory of ω_z, this additional degree of freedom can be chosen independently. Examples of possible choices are to simply set $\omega_z = 0$ in order to reduce control effort, or to choose ω_z such that the vehicle heading remains constant or follows a specified trajectory.

8. Performance Evaluation

The overall trajectory generation algorithm, consisting of the basic decoupled trajectory generator (Section 5) and the iterative improvement schemes (Section 6), has now been described. Generated trajectories satisfy the feasibility constraints imposed by the dynamics of the quadrocopter by construction. This section will characterize the performance of the presented algorithm with respect to the design objectives of achieving high flight performance (short maneuver durations) and low computational complexity (short computation times).

8.1 Comparison to Time-Optimal Trajectories

To determine the performance of maneuvers computed by the presented method and to characterize the influence of the decoupling assumptions (presented in Section 4.3), the generated trajectories are compared to time-optimal trajectories.

A method using Pontryagin's minimum principle to derive optimality conditions when computing time-optimal trajectories for quadrocopters was presented previously in [17]. The method uses a two-dimensional version of the quadrocopter model (2)-(7) to make the problem computationally tractable. While computation times on the order of several hours and required user interaction make the method difficult to use in practical applications, the explicit consideration of the minimum principle optimality conditions makes the resulting trajectories strong candidates for truly time-optimal trajectories. They thus provide useful benchmarks for comparing against other methods (such as the one presented in this paper) in order to analyze relative performance.

Time-optimal reference trajectories were computed for hover-to-hover translations with the input saturations chosen to be $a_{\max} = 20\,\mathrm{m\,s^{-2}}$, $a_{\min} = 1\,\mathrm{m\,s^{-2}}$, and $\omega_{xy,\max} = 10\,\mathrm{rad\,s^{-1}}$. The same saturations were used for the real-time trajectory generation algorithm. Both iterative performance improvement schemes presented in Section 6 were applied, with the tolerance of the acceleration tradeoff parameter optimization being $\epsilon_{xz} = 10^{-3}$ and a minimum vertical acceleration grid size of $0.25\,\mathrm{m\,s^{-2}}$. While the time-optimal reference trajectories were computed using a two-dimensional model, no such simplifications were assumed in the trajectory generation algorithm. The initial acceleration (see Equation (35)) was set to $a_0 = \mathrm{g}$.

Consider, as a first example, a purely horizontal translation of $10\,\mathrm{m}$. The planned trajectories of both the time-optimal trajectory generator and the real-time trajectory generation algorithm are shown in Figure 4, and the control inputs along the trajectory can be seen in Figure 5. The duration of the time-optimal maneuver is $1.56\,\mathrm{s}$, whereas the

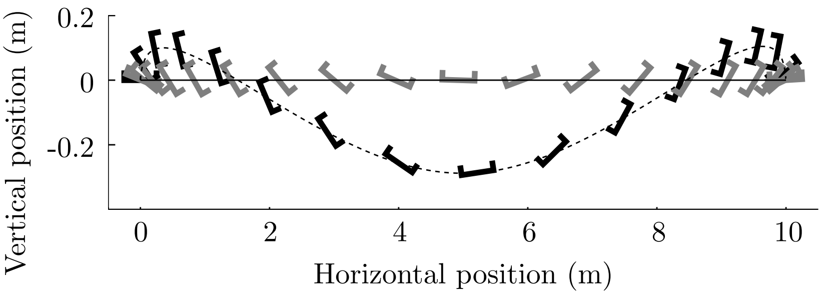

Figure 4. Comparison of a horizontal translation maneuver of 10 m generated using the algorithm presented herein (solid line, gray snapshots; total maneuver duration 2.01 s) and a time-optimal translation [17] (dashed line, black snapshots; total duration 1.56 s). Snapshots are shown at an interval of 0.1 s. The decoupling presented in Section 4.3 results in the vehicle remaining at a constant altitude throughout the maneuver. Note that the vertical axis is not to scale.

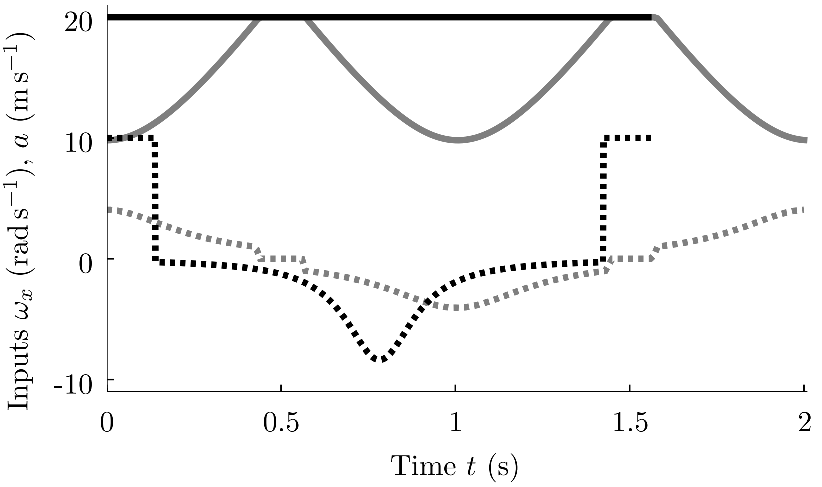

Figure 5. Control inputs corresponding to the comparison of the purely horizontal maneuver shown in Figure 4. The solid lines represent the thrust input a, and the dotted lines the rotational rate input ω_x. Black lines correspond to the time-optimal translation [17], and gray lines are the inputs of the trajectory generated by the algorithm presented herein. Note that the decoupling yields non-maximal thrust inputs when the vehicle rotates (beginning, center, and end of the maneuver), and the conservatism introduced in the decoupled jerk feasibility constraints leads to a less efficient use of the rotational rate control input.

trajectory generated with the algorithm presented herein takes 2.01 s, which amounts to an increase of 29 %. Considering the shape of the generated trajectories, two significant factors contributing to the increased duration can be identified:

1) The decoupling assumption, along with identical initial and final altitudes and the maneuver starting at rest, leads to the generated trajectory being entirely at constant altitude. In comparison, the time-optimal maneuver exploits any available thrust during attitude changes to accelerate upwards, thereby allowing the vehicle to tilt further and increase its horizontal acceleration. This effect is further enabled because the time-

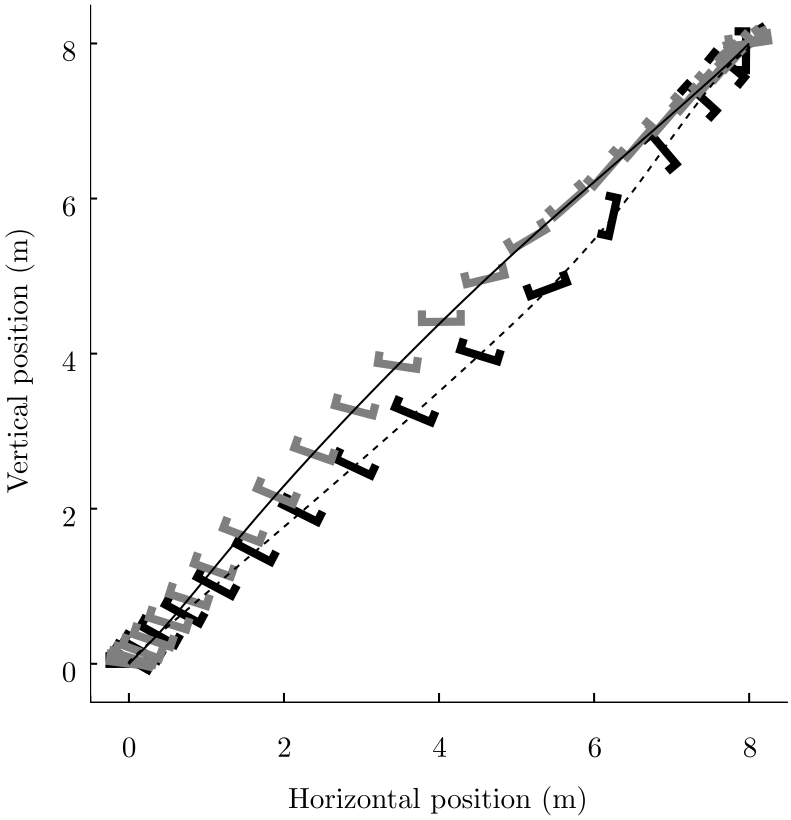

Figure 6. Comparison of a diagonal translation maneuver of 8 m in the horizontal and 8 m in the vertical. The solid line and gray snapshots show the trajectory generated using the algorithm presented herein (total maneuver duration 2.78 s). The dashed line and black snapshots represent a time-optimal translation [17] (total duration 1.76 s). Snapshots are shown at an interval of 0.1 s. Note that the time-optimal maneuver involves the vehicle rotating to a pitch angle of approx. 140° in the final deceleration phase, whereas the minimum vertical acceleration constraint of the trajectory generation algorithm presented herein limits the pitch angle to approximately 50°.

optimal maneuver is planned using the true control inputs (a, ω_x), enabling instantaneous changes in thrust. The decoupling presented herein results in the trajectory being planned in jerk, meaning that the thrust (as computed by Equation (9)) is always continuous throughout the maneuver. Although this lack of discontinuities in the thrust control input results in lower performance, it may be desirable due to the underlying actuator dynamics of the quadrocopter.

2) A further noticeable effect is highlighted in the control inputs corresponding to the two maneuvers (Figure 5): while the time-optimal maneuver uses the maximum rotational rate input for significant durations at the beginning and end of the maneuver, the motion generated using the algorithm presented herein results in lower rotational rate control inputs. This is mostly caused by the conservative approximations used to decouple the rotational rate feasibility conditions (30).

As a second example, consider a translation of 8 m in the horizontal and 8 m in the vertical. The time-optimal trajectory and the generated trajectory are shown in Figure 6.

In this case, the duration of the generated trajectory is 58 % longer than the time-optimal trajectory (2.78 s vs. 1.76 s). Two contributing factors can be identified:

1) The arguments about the use of jerk as a control input also apply in this case, leading for example to a slower initial acceleration than the time-optimal motion provides.

2) A significant difference can be seen towards the end of the motion: during the final deceleration phase, the generated trajectory results in the vehicle pitching to an angle of approximately 50°, applying the optimized minimal vertical acceleration $\ddot{z}_{\text{min}}$ and the minimal horizontal acceleration $-\ddot{x}_{\text{max}}$. The time-optimal trajectory results in the vehicle being pitched to an upside-down attitude at an angle of approximately 140°, applying full thrust and thereby achieving significantly lower vertical accelerations. This difference is caused by the decoupling of the three degrees of freedom, wherein the minimum vertical acceleration constraint (20) was introduced. The value of the constraint was then optimized to trade off the permissible jerk against allowable vertical deceleration (Section 6.2).

8.2 Computational Performance

In the following, computation times will be given for a laptop computer with an Intel Core i7-2620M processor running at 2.7 GHz. While a laptop computer would typically be used as an off-board computation platform, there are multi-rotor computation platforms that provide comparable computation capabilities (e.g., the Ascending Technologies Mastermind onboard PC). As will be obvious from the following results, it is also straightforward to deploy the algorithm on platforms with less available computation power.

Minimal Computational Requirements The minimal implementation of the trajectory generation algorithm consists of the solution of the decoupled planning problem (presented in Section 5) for each of the three degrees of freedom. The decoupling parameters $\alpha_x, \alpha_z, \ddot{z}_{\text{min}}$ are chosen *a priori*. This minimal implementation results in low computational requirements, at the price of reduced flight performance.

The computation time was evaluated through the computation of several million trajectories from randomized initial conditions. The average computation time of a three-dimensional flight trajectory was 24.6 µs. This short computation time is particularly suited for implementations on low-power platforms: based on typical feedback rates on the order of 50 Hz to 100 Hz [28, 30], a current-generation microcontroller (as used in open-source flight control units [28] and costing less than 5US$) would suffice to use the trajectory generation algorithm as an implicit feedback law[7].

Iterative Improvement Performance After the execution of the minimal implementation of the algorithm, remaining computation time can be used to iteratively im-

---

[7] Consider, as an example, the STM32F3 microcontroller by STMicroelectronics, which is based on the ARM Cortex M4F architecture. At 72 MHz, it achieves a CoreMark [9] score of approximately 245 compared to the laptop computer's CoreMark score of 13986. Using the CoreMark scores as a rough performance measure, re-planning trajectories at 100 Hz using the minimal implementation would therefore require about 15% of the available computation time on the microcontroller.

prove the trajectory performance as presented in Section 6. For example, it can be shown that the optimization of the acceleration tradeoff parameters α_x and α_z to a tolerance of 1 % requires at most 171 calls of the one-dimensional decoupled trajectory generator, therefore requiring approximately 1.4 ms of computation time. For a tolerance of 10 % in the tradeoff parameters, a maximum of 78 calls suffice (resulting in approximately 0.6 ms computation time). This optimization can then be repeated for varying values of the minimum vertical acceleration $\ddot{z}_{\min}$, where the number of evaluated points may be chosen based on computational constraints.

9. Experimental Validation

To verify the trajectory generation algorithm experimentally, it was implemented in the Flying Machine Arena, an indoor aerial vehicle development platform at ETH Zurich [27]. The vehicles used for the experiments are custom-built quadrocopters that are based on Ascending Technologies 'Hummingbird' vehicles [13]. The electronics mounted on board each vehicle provide inertial measurements and implement body rate feedback loops. The control inputs ($\omega_x, \omega_y, \omega_z, a$ as defined in Section 2.1) are communicated to the vehicle from a desktop computer through a low-latency wireless communication channel at a rate of 50 Hz. A commercial motion capture system provides position and attitude information, which is filtered by a Luenberger observer.

For the experiments presented herein, the trajectory generation algorithm is run recursively and used as a feedback law as presented in Section 7. Every 20 ms, the updated vehicle state (computed by the Luenberger observer) is used used as an initial condition to solve for the trajectory to the target point. The control inputs are then computed and sent to the vehicle.

9.1 Constant Altitude Experiment

As a first test case, trajectories are planned with non-changing altitude target points, shown in Figure 7. The experiment starts with the vehicle at approximately $(x = -2.6\,\mathrm{m}, y = 1.5\,\mathrm{m})$, with the target point being $(x = 3\,\mathrm{m}, y = -4\,\mathrm{m})$, and the vehicle moving away from the target point at a speed of approximately $3\,\mathrm{m\,s^{-1}}$. When the vehicle is within 1 m of the target point, the target point is switched to $(x = 3\,\mathrm{m}, y = 2\,\mathrm{m})$. During the experiment, the vehicle reached a maximum speed of $7\,\mathrm{m\,s^{-1}}$.

While the first target point does not induce an overshoot of the planned trajectory in the positive x-direction, the switch in target point induces an overshoot of up to 40 cm. This overshoot is caused because the acceleration tradeoff parameter α_x (see Section 6.1) is adjusted by the optimization when the trajectory to the new target point is planned, and changes from a value of $\alpha_x = 0.69$ to $\alpha_x = 0.07$ when the target point switches. The trajectory generation's planning objective is to minimize the maneuver duration; almost all available acceleration is allocated to the second degree of freedom after the target point changes because the largest motion is required in that direction.

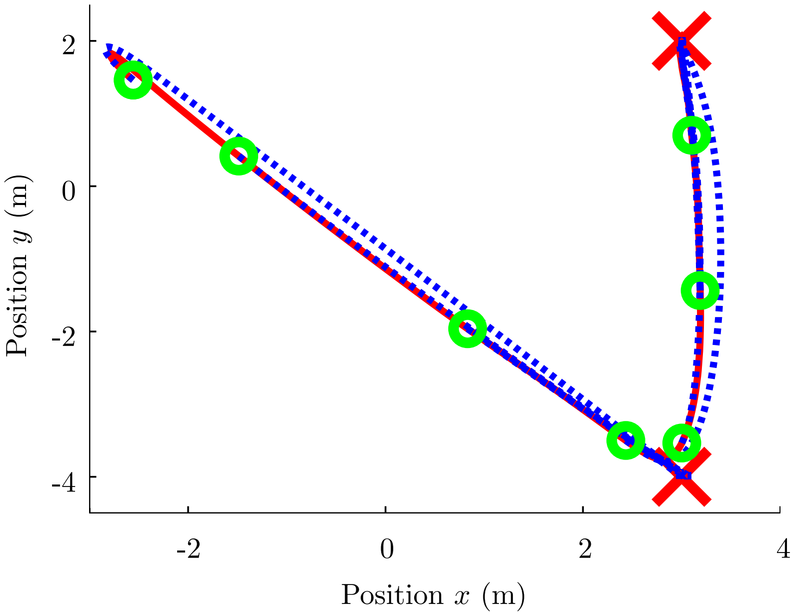

Figure 7. Selection of replanned trajectories during experiment at constant altitude. The trajectory is re-planned every 20 ms, and used as an implicit feedback law by applying the initial control inputs. The figure shows the flight path (solid red) and a small selection of planned trajectories (dotted blue, starting at green circles and ending at red crosses). The vehicle starts at the top-left green circle, moving towards the top left, and the target point is set to the bottom right. As the vehicle approaches the bottom right target point, the target is switched to the top right. Because the trajectory is re-planned at every time step, the new target point is accounted for almost instantaneously.

Note that the implicit control law does not track a previously planned trajectory, but a new trajectory to the target point is computed at every controller update. This replanning behaviour can be observed particularly well when significant deviations from a previously planned trajectory occur, for example during the initial deceleration phase and when the vehicle changes direction after the target point is changed. While the initially planned trajectory contains a significant amount of overshoot, the actual overshoot during execution of the maneuver is smaller. Potential explanations for this behaviour are the unmodeled effects of drag (which can cause more deceleration) [1] and translational lift (which can cause increased propeller efficiency) [21]; both of these are known to have significant influence at the maneuvering speeds encountered in this experiment. Because the higher-than-planned deceleration allows for a more direct flight path, the trajectories planned later on contain almost no overshoot.

9.2 Three-Dimensional Motion

In a second experiment, the set points are varied in all three dimensions. As can be seen in Figure 8, the vehicle starts at approximately ($x = -2\,\mathrm{m}, y = 1\,\mathrm{m}, z = 6\,\mathrm{m}$), with the target point being ($x = 3\,\mathrm{m}, y = -2.5\,\mathrm{m}, z = 1.5\,\mathrm{m}$). Like in the previous experiment,

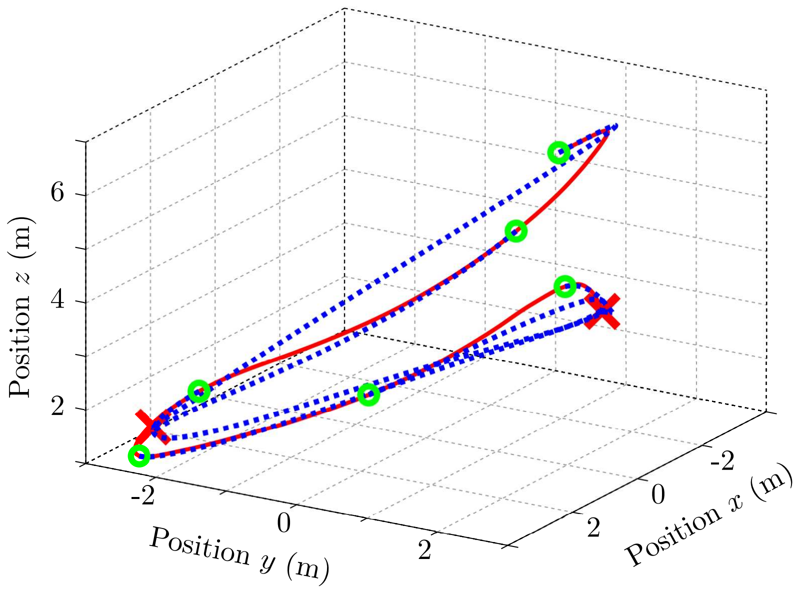

Figure 8. Three-dimensional motion experiment. The vehicle starts at the top-right (green circle) with the target point being at the bottom left (left red cross). As the vehicle approaches the target point, the target is switched to the center-right target point (right red cross). The solid red line shows the flown trajectory, and the dotted blue lines (starting at the green circles) show some of the planned trajectories.

the target point is changed when the vehicle is within 1 m of the target point, and the new target point is then ($x = 0\,\mathrm{m}, y = 2.5\,\mathrm{m}, z = 4\,\mathrm{m}$). The vehicle starts at a speed of $3.6\,\mathrm{m\,s^{-1}}$, and reaches a peak speed of $7.2\,\mathrm{m\,s^{-1}}$.

Figure 9 shows the flight trajectory over time along with the same planned trajectories that are seen in Figure 8. Note that, particularly during high-speed flight phases, the vehicle lags behind the planned trajectories. This behaviour is again consistent with aerodynamic effects [35], and is excarberated (in comparison to the constant altitude experiment) by the vertical speed of the vehicle ranging from $-3.8\,\mathrm{m\,s^{-1}}$ to $2.8\,\mathrm{m\,s^{-1}}$. The propeller inflow velocity thus changes considerably.

9.3 Other Uses

This trajectory generation algorithm has also been used extensively as a building block in higher-level tasks. An example of this is its use in the Flight Assembled Architecture project [41], in which a fleet of four quadrocopters assembled a 6 m tall tower out of 1500 foam bricks (see Figure 10) in front of a live audience. To achieve this task, the minimal implementation of the trajectory generation algorithm (that is, using fixed decoupling parameters $\alpha_x, \alpha_z, \ddot{z}_{\min}$) was used to plan the motions of quadrocopters between battery charging stations, the pick-up points of the foam bricks, and their respective placement point. The ability to plan trajectories from non-rest conditions was used in conjunction with way points in order to guide vehicles around obstacles without stopping. Furthermore, the planned trajectories were used as an input to a space reservation system

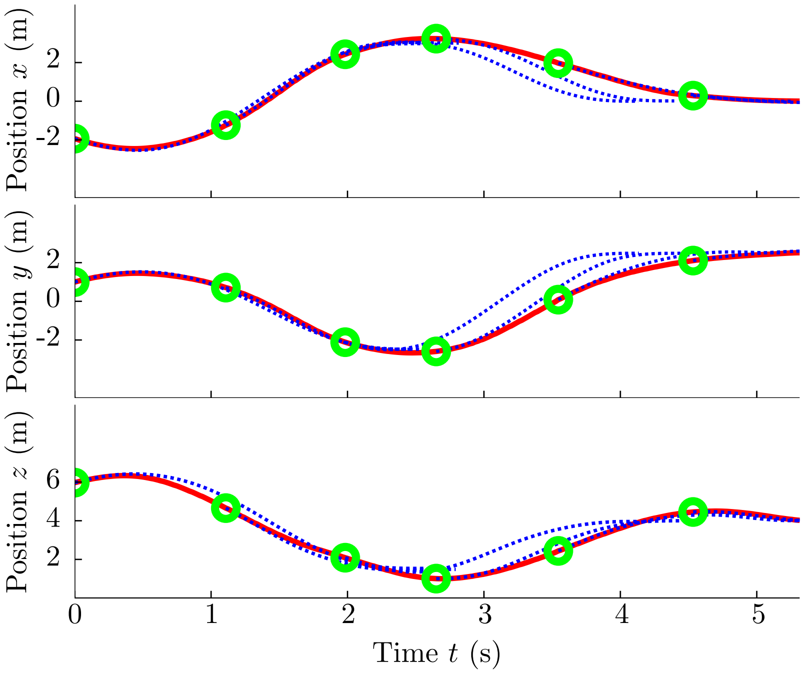

Figure 9. Flown (solid red) and a selection of planned (dashed blue, starting at green circles) trajectories over time for the experiment shown in Figure 8. The set point switches from $(x = 3\,\mathrm{m}, y = -2.5\,\mathrm{m}, z = 1.5\,\mathrm{m})$ to $(x = 0\,\mathrm{m}, y = 2.5\,\mathrm{m}, z = 4\,\mathrm{m})$ at $t = 1.94\,\mathrm{s}$.

(inspired by [37]) in order to ensure that no collisions occur. Over the duration of the project, tens of thousands of trajectories were generated.

10. Conclusion

This paper presented and analyzed a method for generating quadrocopter flight trajectories. The method efficiently computes trajectories that both allow for fast motion and are guaranteed to be feasible.

The decoupling of the trajectory generation problem under conservative approximation of the feasibility constraints allows the problem to be simplified considerably, and the computation of time-optimal trajectories for the decoupled system reduces to determining the switching times of the bang-singular solution trajectory. A particular strength of the method is that it very quickly allows the computation of a provably feasible trajectory (that is, within few tens of microseconds on a laptop computer), but can also iteratively provide higher-quality solutions if more computation time is available. The iterative improvement allows optimal design parameters to be determined and can be carried out until convergence criteria are met, or aborted at any time if no more computation time is available.

Analysis of the computed trajectories showed structural differences when compared to time-optimal trajectories. These differences are mostly caused by the decoupling ap-

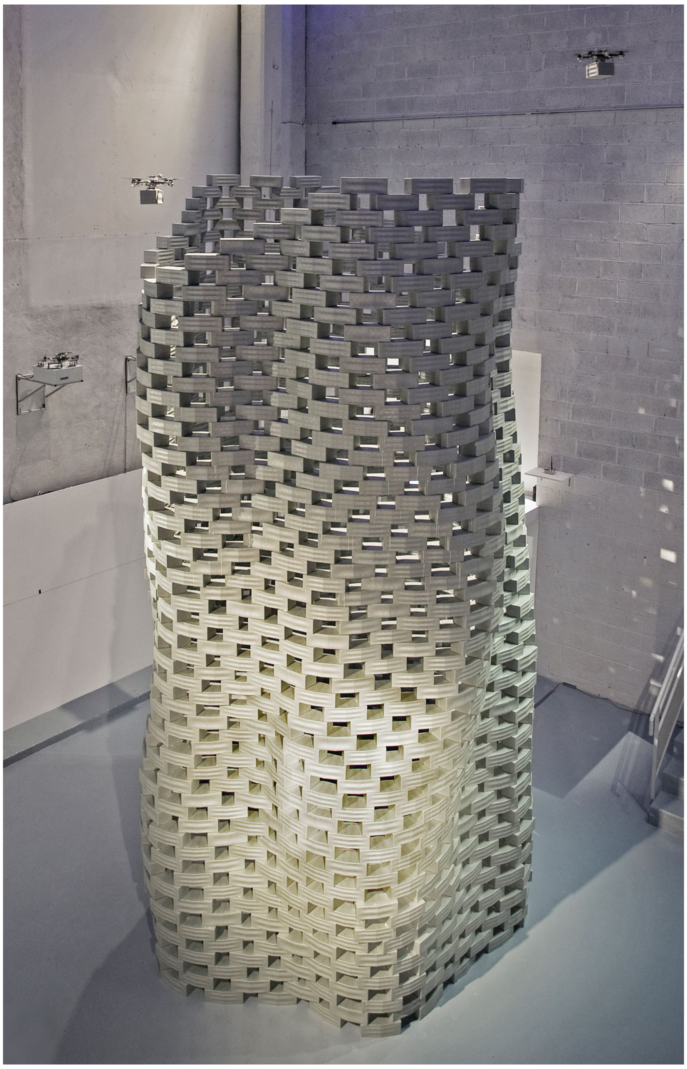

Figure 10. Tower made of foam bricks, assembled by quadrocopters during the Flight Assembled Architecture project [41]. Two of the quadrocopters can be seen above the tower to the left and right; a third is seen resting on a charging station on the wall to the left. The flight paths used by the quadrocopters for this project were generated using the trajectory generation algorithm presented in this paper.

proach. The trajectory generation algorithm presented herein trades off the full use of the dynamic capabilities of quadrocopters for the ability to compute tens of thousands of trajectories every second. This makes it particularly suitable to applications that require the planning of large numbers of trajectory candidates (such as, for example, sampling-based path planners) or fast re-planning due to rapidly changing or *a priori* unknown environments or task objectives.

The approach was experimentally verified for quadrocopters maneuvering at moderately high speeds (up to $7\,\mathrm{m\,s^{-1}}$), and was shown to cope well with disturbances due to unmodeled dynamic effects and changing target points. Its robustness and applicability as a building block in more complex systems has been demonstrated by its use in the assembly of a large structure by quadrocopters, in the process of which tens of thousands of trajectories were generated.

While this paper has demonstrated the validity of the decoupling approach and the possibility to design algorithms of sufficiently low computational efficiency, a number of potential improvements remain to be investigated.

First, the algorithm presented herein is aimed at applications where relatively small spaces are to be navigated. This allows the quadrotor dynamics to be simplified by neglecting aerodynamic effects, and leads to the planned trajectories not being subject to velocity constraints. To broaden the scope of the algorithm, an interesting extension would be to include velocity constraints, which could be used to guarantee that aerodynamic effects remain sufficiently small. Such constraints may also be imposed by the sensing modalities used for the state estimation of the quadrocopter.

Second, in the design of the decoupled feasibility constraints, the per-axis jerk constraint (Equation (30)) was chosen to be symmetric for all three axes. The tradeoff of the three jerk values could also be considered to be design parameters, and could be optimized for each maneuver.

Finally, the iterative improvement schemes used to adapt the decoupling parameters to the specific problem data could be improved, thereby increasing performance. The optimal choice of the decoupling parameters forms an optimization problem that contains strong structure, and should therefore lend itself to the application of many more advanced optimization methods. When the trajectory generator is called repeatedly (for example because it is used as a feedback law), the use of the previous solution to initialize the optimization algorithm could further reduce computational cost. An alternative approach would be the application of machine learning algorithms to predict the optimal choice of the decoupling parameters from the initial conditions. This would allow the move of the computational burden to before the real-time use of the algorithm without compromising performance.

A. Feasibility of Time-Varying Acceleration Constraints

This appendix contains the proof that the time-varying acceleration constraints (36) (used to account for non-zero initial accelerations) satisfy the maximum thrust constraint (14). For the purpose of simplicity, the proof will first be presented for a related set of time-varying constraints (referred to as *constant time bound change*). The findings of this related acceleration change will then be applied to the actual time-varying acceleration constraints (36).

Constant time bound change

Consider the set of time-varying acceleration constraints $\overline{\ddot{x}}_{\mathrm{ct}}, \overline{\ddot{y}}_{\mathrm{ct}}, \overline{\ddot{z}}_{\mathrm{ct}}$, defined according to Equation (37), where the slope values c_x, c_y, c_z are chosen such that all degrees of freedom reach their respective bounds $(\ddot{x}_{\text{max}}, \ddot{y}_{\text{max}}, \ddot{z}_{\text{max}})$ in constant time ΔT_0, i.e.

$$|\ddot{x}_0| + c_x \Delta T_0 = \ddot{x}_{\text{max}} \tag{66}$$

$$|\ddot{y}_0| + c_y \Delta T_0 = \ddot{y}_{\text{max}} \tag{67}$$

$$|\ddot{z}_0| + c_z \Delta T_0 = \ddot{z}_{\text{max}} \tag{68}$$

with ΔT_0 defined in Equation (38). Then the maximum acceleration bound (14) remains satisfied for all $t \in (0 \quad \Delta T)$:

$$\begin{aligned} &\overline{\ddot{x}}_{\text{ct}}^2 + \overline{\ddot{y}}_{\text{ct}}^2 + \left(\overline{\ddot{z}}_{\text{ct}} + \mathrm{g}\right)^2 = \\ &\left(|\ddot{x}_0| + \frac{c_x}{\Delta T} t\right)^2 + \left(|\ddot{y}_0| + \frac{c_y}{\Delta T} t\right)^2 \\ &\qquad + \left(|\ddot{z}_0| + \mathrm{g} + \frac{c_z}{\Delta T} t\right)^2 \leq a_{\text{max}} \,. \end{aligned} \tag{69}$$

Proof: Without loss of generality, let the normalized time be $\tilde{t}$ such that $\Delta\tilde{T} = 1$. Rewriting (69) in terms of $(\ddot{x}_{\text{max}}, \ddot{y}_{\text{max}}, \ddot{z}_{\text{max}})$ using Equations (66)-(68) results in the condition

$$\begin{aligned} &\left(\ddot{x}_{\text{max}} + c_x \left(\tilde{t} - 1\right)\right)^2 + \left(\ddot{y}_{\text{max}} + c_y \left(\tilde{t} - 1\right)\right)^2 + \\ &\qquad \left(\ddot{z}_{\text{max}} + \mathrm{g} + c_z \left(\tilde{t} - 1\right)\right)^2 \leq a_{\text{max}} \end{aligned} \tag{70}$$

which can be rewritten using Equation (34) to be

$$\begin{aligned} &2\left(\ddot{x}_{\text{max}} c_x + \ddot{y}_{\text{max}} c_y + \left(\ddot{z}_{\text{max}} + \mathrm{g}\right) c_z\right) \geq \\ &\qquad \left(1 - \tilde{t}\right)\left(c_x^2 + c_y^2 + c_z^2\right) \,. \end{aligned} \tag{71}$$

Because $0 \leq \left(1 - \tilde{t}\right) \leq 1$ a sufficient condition for the inequality to hold is

$$\left(c_x^2 + c_y^2 + c_z^2\right) \leq 2\left(\ddot{x}_{\text{max}} c_x + \ddot{y}_{\text{max}} c_y + \left(\ddot{z}_{\text{max}} + \mathrm{g}\right) c_z\right) \,. \tag{72}$$

The initial acceleration (35) can be rewritten using Equations (66)-(68) to yield

$$\left(\ddot{x}_{\text{max}} - c_x\right)^2 + \left(\ddot{y}_{\text{max}} - c_y\right)^2 + \left(\ddot{z}_{\text{max}} + \mathrm{g} - c_z\right)^2 = a_0 \tag{73}$$

with $a_0 \leq a_{\text{max}}$ by design. Expanding the above yields the required inequality (72).

Time-varying acceleration constraints

The time-varying acceleration constraints $\overline{\ddot{x}}, \overline{\ddot{y}}, \overline{\ddot{z}}$ presented in Section 4.3 differ from the constant time bound change $\overline{\ddot{x}}_{\text{ct}}, \overline{\ddot{y}}_{\text{ct}}, \overline{\ddot{z}}_{\text{ct}}$ in the choice of c_x, c_y, c_z in the case that the initial acceleration exceeds the allowed acceleration magnitude (e.g. $|\ddot{x}_0| > \ddot{x}_{\max}$). In this case, the actual time-varying bounds decrease at the minimum jerk $-\dddot{w}_{\max}$ until the allowable acceleration (e.g. $\ddot{x}_{\max}$) is reached. To show that the inequality (69) remains satisfied, it suffices to note that, through the construction of c_x, c_y, c_z in Equations (66)-(68),

$$c_x \geq -\dddot{w}_{max}, \;\; c_y \geq -\dddot{w}_{max}, \;\; c_z \geq -\dddot{w}_{max} \tag{74}$$

holds, and therefore

$$\overline{\ddot{x}}_{\mathrm{ct}} \geq \overline{\ddot{x}}, \quad \overline{\ddot{y}}_{\mathrm{ct}} \geq \overline{\ddot{y}}, \quad \overline{\ddot{z}}_{\mathrm{ct}} \geq \overline{\ddot{z}} \tag{75}$$

holds for all times t. The time-varying acceleration constraints must therefore be feasible with respect to the maximum acceleration bound (14) since the constant time bound change is feasible.

References

[1] Moses Bangura and Robert Mahony. Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor. In *Australasian Conference on Robotics and Automation*, 2012.

[2] Dennis S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, third edition, 2005.

[4] Patrick Bouffard, Anil Aswani, and Claire J Tomlin. Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *International Conference on Robotics and Automation*, 2012.

[5] Y. Bouktir, M. Haddad, and T. Chettibi. Trajectory Planning for a Quadrotor Helicopter. In *Mediterranean Conference on Control and Automation*, 2008.

[6] Arthur Earl Bryson and Yu-Chi Ho. *Applied Optimal Control*. Taylor & Francis, 1975.

[7] Richard L Burden and J Douglas Faires. *Numerical Analysis*. Brooks/Cole, ninth edition, 2011.

[8] Ian D. Cowling, Oleg A. Yakimenko, and James F. Whidborne. A Prototype of an Autonomous Controller for a Quadrotor UAV. In *European Control Conference*, 2007.

[9] Shay Gal-on and Markus Levy. Exploring CoreMark - A Benchmark Maximizing Simplicity and Efficacy. *The Embedded Microprocessor Benchmark Consortium (EEMBC)*, 2012.

[10] Carlos E. García, David M. Prett, and Manfred Morari. Model Predictive Control: Theory and Practice - A Survey. *Automatica*, 25(3):335–348, 1989.

[11] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, October 2003.

[12] Corrado Guarino, Lo Bianco, and Fabio Ghilardelli. Third Order System for the Generation of Minimum-Time Trajectories with Asymmetric Bounds on Velocity , Acceleration , and Jerk. In *International Conference on Intelligent Robots and Systems: Workshop on Robot Motion Planning: Online, Reactive, and in Real-time*, 2012.

[13] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[14] Juergen Hahn and Thomas F. Edgar. An Improved Method for Nonlinear Model Reduction Using Balancing of Empirical Gramians. *Computers & Chemical Engineering*, 26(10):1379–1397, 2002.

[15] Richard F. Hartl, Suresh P. Sethi, and Raymond G. Vickson. A Survey of the Maximum Principles for Optimal Control Problems with State Constraints. *SIAM Review*, 37(2):181–218, 1995.

[16] Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. In *IFAC World Congress*, 2011.

[17] Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control. *Autonomous Robots*, 33(1-2):69–88, 2012.

[18] H. Hermes and G. Haynes. On The Nonlinear Control Problem With Control Appearling Linearly. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 1(2):85–108, 1963.

[19] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor Helicopter Trajectory Tracking Control. In *Conference on Decision and Control*, 2008.

[20] Jonathan P. How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.

[21] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering. In *International Conference on Robotics and Automation*, 2009.

[22] Peter C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

[23] Qimi Jiang, Daniel Mellinger, Christine Kappeyne, and Vijay Kumar. Analysis and Synthesis of Multi-Rotor Aerial Vehicles. In *International Design Engineering Technical Conference*, 2011.

[24] Sertac Karaman and Emilio Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[25] Alonzo Kelly and Bryan Nagy. Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[26] Torsten Kroger and Friedrich M. Wahl. Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events. *IEEE Transactions on Robotics*, 26(1):94–111, 2010.

[27] Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A Platform for Aerial Robotics Research and Demonstration: The Flying Machine Arena. *Mechatronics*, 24(1):41–54, 2014.

[28] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Frandorfer, and Marc Pollefeys. PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

[29] Daniel Mellinger and Vijay Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *International Conference on Robotics and Automation*, 2011.

[30] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[31] Mark B. Milam, Kudah Mushambi, and Richard M. Murray. A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems. In *Conference on Decision and Control*, 2000.

[32] Mark W. Mueller and Raffaello D'Andrea. A Model Predictive Controller for Quadrocopter State Interception. In *European Control Conference*, 2013.

[33] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. A Computationally Efficient Algorithm for State-to-State Quadrocopter Trajectory Generation and Feasibility Verification. In *International Conference on Intelligent Robots and Systems*, 2013.

[34] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot. In *Australasian Conference on Robotics and Automation*, 2006.

[35] Caitlin Powers, Daniel Mellinger, Aleksandr Kushleyev, and Bruce Kothmann. Influence of Aerodynamics and Proximity Effects in Quadrotor Flight. In *International Symposium on Experimental Robotics*, 2012.

[36] Oliver Purwin and Raffaello D'Andrea. Trajectory Generation and Control for Four Wheeled Omnidirectional Vehicles. *Robotics and Autonomous Systems*, 54:13–22, 2006.

[37] Oliver Purwin, Raffaello D'Andrea, and Jin-Woo Lee. Theory and Implementation of Path Planning by Negotiation for Decentralized Agents. *Robotics and Autonomous Systems*, 56(5):422–436, 2008.

[38] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial Trajectory Planning for Quadrotor Flight. In *International Conference on Robotics and Automation*, 2013.

[39] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Autonomous Mobile Robots*. The MIT Press, second edition, 2011.

[40] Wannes Van Loock, Goele Pipeleers, and Jan Swevers. Time-Optimal Quadrotor Flight. In *European Control Conference*, 2013.

[41] J Willmann, F Augugliaro, T Cadalbert, Raffaello D'Andrea, Fabio Gramazio, and Matthias Kohler. Aerial Robotic Construction Towards a New Field of Architectural Research. *International Journal of Architectural Computing*, 10:439–460, 2012.

Paper III

Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters

Markus Hehn and Raffaello D'Andrea

Abstract

This paper presents an algorithm that permits the calculation of interception maneuvers for quadrocopters. The translational degrees of freedom of the quadrocopter are decoupled. Pontryagin's minimum principle is used to show that the interception maneuver that minimizes the time to rest after the interception is identical to the time-optimal maneuver that drives the vehicle to the position at which it comes to rest after the interception. This fact is leveraged to apply previously developed, computationally efficient methods for the computation of interception maneuvers. The resulting trajectory generation algorithm is computationally lightweight, permitting its use as an implicit feedback law by replanning the trajectory at each controller update. The validity and performance of the approach is demonstrated experimentally by intercepting balls mid-flight. The real-time trajectory generation permits to take into account changes in the predicted ball flight path at each controller update.

Published in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. The publication includes a multimedia attachment available at `www.hehn.be` and `ieeexplore.ieee.org`.
DOI: 10.1109/IROS.2012.6386093

©2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

1. Introduction

Quadrocopters have been widely adopted as experimental platforms for research in flying robotics (see, for example, the testbeds [15, 20]). Reasons for the popularity of these vehicles include the ability to hover, mechanical simplicity and robustness, and their exceptional maneuverability due to typically high thrust-to-weight ratios combined with the off-center mounting of the propellers.

From a controls perspective, a recent focus has been the planning and following of trajectories that exploit the dynamical capabilities of these vehicles. Results include algorithms that plan trajectories from classes of motion primitives, such as lines [14] or polynomials [4, 7], while others solve an optimal control problem for approximate or full vehicle dynamics (e.g. for minimum snap [19] or minimum time [12]).

In this paper, we consider the problem of using a quadrocopter for the purpose of interception. The general interception problem has been studied in an optimal control context for several decades (see, for example, [5] and references therein). Variations of the problem have also been studied in robotics (e.g. for robotic arms [1] and ground robots [22]) For quadrotor applications, interception problems have been treated in a number of scenarios, including ball juggling, where the interception was more strongly constrained to occur at a specified velocity and attitude [21]. In [3], a ball flight path was intercepted on a given plane by setting the controller reference position to the interception point.

The method we present herein permits the computation of interception maneuvers for quadrocopters in real time. These maneuvers are optimal in that they minimize the time to rest after the interception event, when decoupled dynamics are assumed. This paper shows that the resulting maneuver structure is identical to the time-optimal maneuver that brings the vehicle to the position at which it comes to rest after the interception. This allows us to apply the efficient trajectory planning algorithm that was introduced in [11]. Because the entire trajectory (consisting of the interception and to-rest motion thereafter) is planned, it is easy to verify additional constraints such as, for example, maximum allowable positions.

The trajectory generation algorithm is computationally light weight, permitting us to recompute trajectories in real time at update rates on the order of tens to hundreds of replannings per second. This also permits us to use the trajectory generation as an implicit feedback law by applying the control inputs of the first section of the planned trajectory (similar to model predictive control [8]) at each controller update. Furthermore, trajectories can be planned from arbitrary initial states, and the generated trajectory is guaranteed to be feasible with respect to the dynamic and input constraints of the vehicle.

The remainder of this paper is structured as follows: In Section 2, we introduce the dynamic model of the quadrocopter used in the trajectory generation. In Section 3, we provide a brief overview of the previously presented trajectory generation algorithm for time-optimal maneuvers to a specified position. Section 4 presents the interception

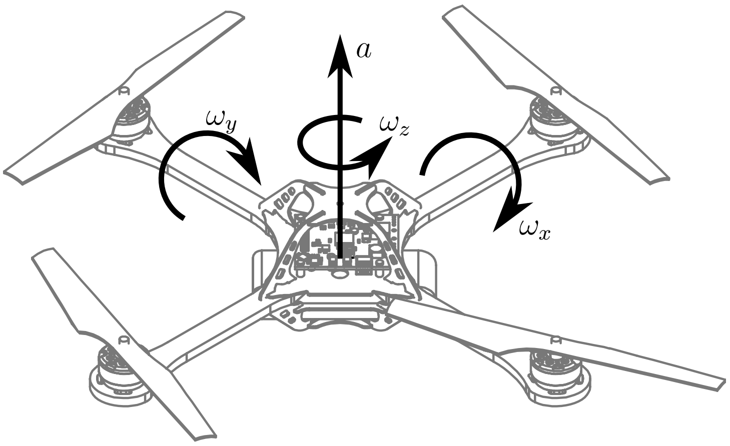

Figure 1. The four control inputs of the quadrotor vehicle. The rotational rates ω_x, ω_y, and ω_z are assumed to follow commands without dynamics or delay. This is motivated by a high-bandwidth on-board controller.

problem and includes the derivation of optimality conditions for interception trajectories. In Section 5, we discuss the implementation of an algorithm computing such interception maneuvers. Section 6 describes the experimental setup we use to intercept balls mid-flight and presents experimental results, while Section 7 draws conclusions, highlighting directions for future research.

2. Vehicle Dynamics

The quadrocopter is described by six degrees of freedom: Its translational position (x, y, z) in the inertial frame O and its attitude V, defined by the rotation matrix ${}^{\mathrm{O}}_{\mathrm{V}}R$.

The four control inputs of the vehicle are the desired rotational rates about the vehicle body axes (ω_x, ω_y, and ω_z), and the mass-normalized collective thrust, a, as shown in Figure 1.

It is assumed that the three rotational rates ω_x, ω_y, ω_z can be changed arbitrarily fast. This is motivated by the large rotational accelerations quadrocopters can achieve due to their ability to produce high torques and their low rotational inertia [17], which allow the rotational rate commands to be tracked with very high bandwidth on board the vehicle.

Analogously to the vehicle body rates, we assume that the thrust can be changed instantaneously. Experimental results have shown that the true thrust dynamics, caused by the dynamics of the motors changing speed, are about as fast as the rotational rate dynamics.

It is further assumed that all control inputs are subject to saturation. The magnitude of the vehicle body rates are limited (such limitations can be caused, for example, by the range of the gyroscopes, or limitations of the body rate tracking controllers). The collective thrust is limited by a minimum and a maximum thrust

$$a_{\min} \leq a \leq a_{\max} \ , \tag{1}$$

where $a_{\min} > 0$. This positive lower bound is motivated by the fact that typical quadrotor vehicles have propellers of fixed-pitch type, and are not able to stop or reverse the propellers' direction of rotation in flight.

2.1 Equations of Motion

The translational acceleration of the vehicle is dictated by its attitude and the collective thrust control input. In the inertial frame, the translational acceleration is

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} , \tag{2}$$

where g denotes the gravitational acceleration.

The change of vehicle attitude is related to the rotational control inputs through [16]

$${}^{\mathrm{O}}_{\mathrm{V}}\dot{R} = {}^{\mathrm{O}}_{\mathrm{V}}R \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} . \tag{3}$$

3. Trajectory Generation Algorithm

In this section, we provide an overview of the method used to generate trajectories from arbitrary initial conditions to a target point. This approach was introduced in [11], and is described here briefly because it will later be used.

The trajectory generation problem is simplified by approximating the quadrotor dynamics with three triple integrators:

$$\begin{bmatrix} \dddot{x} \\ \dddot{y} \\ \dddot{z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} , \tag{4}$$

and planning the trajectory in the jerks $(v_x(t), v_y(t), v_z(t))$. The control inputs along a trajectory $(x(t), y(t), z(t))$ are

$$f(t) := \begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathrm{g} \end{bmatrix} , \tag{5}$$

$$\bar{f}(t) := \frac{f(t)}{\|f(t)\|} \ , \tag{6}$$

$$a(t) = \|f(t)\| \ , \tag{7}$$

$$\begin{bmatrix} \omega_y(t) \\ -\omega_x(t) \\ 0 \end{bmatrix} = {}^{\mathrm{V}}_{\mathrm{O}}R(t)\dot{\bar{f}}(t) \ . \tag{8}$$

For a trajectory to be feasible, the resulting control inputs a, ω_x, and ω_y must lie within their allowable sets. In order to satisfy the thrust constraint (1), constant acceleration bounds for each degree of freedom are introduced. Furthermore, it was shown that it is difficult to choose allowable jerk values such that the rotational rate control inputs (8) do not exceed the vehicle body rate limitations. It is however straightforward to compute the rotational rates along a planned trajectory. If these exceed limitations, it was shown that a feasible trajectory can always be found by reducing the allowable jerk values.

A trajectory to the target position can be computed for each degree of freedom of the approximate dynamics (4), with given acceleration and jerk constraints. Time-optimal trajectories from arbitrary initial conditions to a target position (to be reached at rest) were presented. Using Pontryagin's minimum principle, it follows that the switching function is of parabolic shape with additional intervals at the zero crossings in which it remains zero. The optimal control input v^* is bang-singular, consisting of at most five distinct regions:

- $[0\,t_1)$: $v^* = \pm v_{\max}$,
- $[t_1\,t_2)$: $v^* = 0$,
- $[t_2\,t_3)$: $v^* = \mp v_{\max}$,
- $[t_3\,t_4)$: $v^* = 0$,
- $[t_4\,t_f]$: $v^* = \pm v_{\max}$,

where $v_{\max}$ denotes the maximum allowable jerk. In this solution, the intervals $[t_1\,t_2)$ and $[t_3\,t_4)$ represent singular arcs, in which the control input is determined by the acceleration remaining constant on its boundary. The initial control input and the five times $t_1 \ldots t_f$ fully define the maneuver.

The performance of this trajectory generation algorithm has been demonstrated in a number of experiments [11]. It was shown that it can be used as an implicit feedback law by re-planning a trajectory for each controller update. The computational load caused by this is minor, with computation taking no longer than 0.2 ms on a conventional desktop computer.

We will now adapt the trajectory generation algorithm, not to reach a target position as quickly as possible, but to cross a specified position at a specified time.

4. The Interception Maneuver

While the time-optimal to-rest maneuvers presented above are very useful in many applications, situations may arise in which a specified position must be reached more quickly than it is possible to reach at rest. Using the same coordinate decoupling (4) as before, we will now derive trajectories for a given *interception point* $(\hat{x}, \hat{y}, \hat{z})$, which must be reached at a specified *interception time* $\hat{t}$.

Because the interception constraint is not sufficient to uniquely define the trajectory, we further require the planned trajectories to bring the vehicle to rest as quickly as possible after the interception. This choice provides two advantages:

- The problem statement now includes not only the motion to intercept the position at the right time, but also the motion to bring the vehicle back to rest after the interception. This is desirable because it permits easy verification of constraints such as a maximum desirable displacement.
- The choice of the time to rest as a cost function forces aggressive deceleration after the time of interception, avoiding excessive overshoot.

We will now formally state the trajectory generation problem described above, allowing the optimality conditions to be applied in order to find solutions to it.

4.1 Problem statement

We describe the problem for a single degree of freedom, assuming that we decouple the dynamics according to equation (4). We denote the degree of freedom q, and state the optimal control problem as follows: Let $s = (s_1,\, s_2,\, s_3) = (q,\, \dot{q},\, \ddot{q})$ be the state. The objective is to minimize t_f subject to the system dynamics

$$\dot{s}_1 = s_2 \ , \tag{9}$$

$$\dot{s}_2 = s_3 \ , \tag{10}$$

$$\dot{s}_3 = v \ , \tag{11}$$

and the initial, interception, and final state constraints

$$s(t = 0) = s_0 \ , \tag{12}$$

$$s_1(t = \hat{t}) = \hat{q} \ , \tag{13}$$

$$s_2(t = t_f) = s_3(t = t_f) = 0 \ , \tag{14}$$

where $\hat{t}$ is the interception time and $\hat{q}$ is the interception position. Furthermore, the input and state constraints

$$|v| \le v_{\max} \ , \tag{15}$$

$$\ddot{q}_{\min} \leq s_3 \leq \ddot{q}_{\max} \tag{16}$$

must be satisfied.

We begin by noting that the algorithm presented in Section 3 already solves the above problem if it is possible to reach $\hat{q}$ such that t_f is smaller than $\hat{t}$: Because the computed trajectory reaches $\hat{q}$ at rest, the interception constraint (13) will be satisfied by this time-optimal motion followed by the vehicle remaining at rest until the interception time (i.e. $v^* = 0$ for $t_f < t \leq \hat{t}$). From here on, we will assume that $\hat{t}$ is smaller than t_f, and will now derive the optimality conditions and resulting structure of optimal maneuvers.

4.2 Necessary Optimality Conditions

Analogous to the derivations for the original trajectory generation algorithm, we apply the minimum principle (see, for example, [2]) to derive necessary conditions for optimal input trajectories. The state constraints (16) are handled using a direct adjoining approach [10], in which the Hamiltonian function is augmented by the state constraints. With the cost given to be the final time, the Hamiltonian is then

$$\begin{aligned} H(s, v, \lambda, \eta) = 1 + \lambda_1 s_2 + \lambda_2 s_3 + \lambda_3 v \\ + \eta_1 \left(-\ddot{q}_{\min} + s_3\right) + \eta_2 \left(\ddot{q}_{\max} - s_3\right) \ , \end{aligned} \tag{17}$$

where λ are the adjoint variables and η are state constraint multipliers that fulfill the constraints

$$\eta \ \geq 0 \ , \tag{18}$$

$$\eta_1 = 0 \text{ if } s_3 > \ddot{q}_{\min} \ , \tag{19}$$

$$\eta_2 = 0 \text{ if } s_3 < \ddot{q}_{\max} \ . \tag{20}$$

The adjoint variables evolve over time according to

$$\dot{\lambda} = -\nabla_s H(s, v, \lambda, \eta), \tag{21}$$

from which it follows that

$$\dot{\lambda}_1 = 0 \ , \tag{22}$$

$$\dot{\lambda}_2 = \lambda_1 \ , \tag{23}$$

$$\dot{\lambda}_3 = \lambda_2 + \eta_1 - \eta_2 \ . \tag{24}$$

The optimal control v^* is the control input that minimizes the Hamiltonian function:

$$v^* = \arg\min H(s, v, \lambda, \eta) = \arg\min \lambda_3 v \ . \tag{25}$$

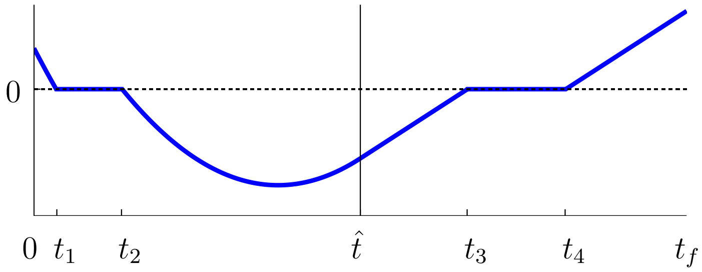

Figure 2. Sketch of the adjoint variable trajectory λ_3 for an example maneuver. At the interception time $\hat{t}$, the shape switches from parabolic to constant slope as λ_1 jumps to zero. The intervals $[t_1 t_2)$ and $[t_3 t_4)$ represent singular arcs, in which one of the acceleration constraints (16) is active.

At the interception time $\hat{t}$, the discontinuity in the first adjoint variable [5] is

$$\lambda_1(\hat{t}^-) = \lambda_1(\hat{t}^+) + \nu \ , \tag{26}$$

where $\hat{t}^-$ and $\hat{t}^+$ signify just before and just after $\hat{t}$, respectively, and ν is a constant Lagrange multiplier. The other two adjoint variables λ_2, λ_3 remain continuous over the interception time. Because the final state $s_1(t_f)$ is free, the costate constraint at the final time is [2]

$$\lambda_1(t_f) = 0 \ . \tag{27}$$

It was shown in [18] that, for problems of this form, the adjoint variables λ are continuous when the acceleration constraint (16) becomes active or stops being active. Furthermore, $\lambda_3 = 0$ must hold over the duration in which a state constraint is active.

The costate dynamics (22)-(24) form a double integrator with respect to λ_3. With λ_1 being zero from $\hat{t}^+$ onwards, and with the intersection time condition (26), it follows that the trajectory of λ_3 has the following shape, which is sketched in Figure 2:

- In the time interval $\left[0 \ \hat{t}^-\right]$, λ_3 has a parabolic shape because $\ddot{\lambda}_3 = \lambda_1$ is constant when no constraints are active.
- In the time interval $\left[\hat{t}^+ \ t_f\right]$, $\ddot{\lambda}_3 = \lambda_1 = 0$, implying that λ_3 has constant slope when no constraints are active.
- Whenever λ_3 has a zero crossing, an acceleration constraint may become active, implying that λ_3 remains zero while the constraint is active.

4.3 Equivalence to time-optimal motions

It follows from the above constraints that the maneuver minimizing the to-rest duration with the interception constraint must have the same structure as the time-optimal trajectories presented in the previous section: Both consist of, at most, three regular arcs (where $\lambda_3 \neq 0$) and two singular arcs (where $\lambda_3 = 0$). Both trajectories are fully specified by the five times t_1, t_2, t_3, t_4, and t_f and the initial control input. For the interception trajectory, an additional constraint arises because the switching function λ_3 is linear after the interception time: The remaining trajectory can only contain two regular arcs and one singular arc.

Now, assume that we have computed an interception maneuver that satisfies the above optimality conditions, implying that it minimizes the to-rest duration after the interception. The maneuver terminates at some position q_f (by definition of the maneuver this position is reached at rest). Then this maneuver is identical to the time-optimal maneuver from the same initial conditions to the position q_f as described in Section 3: Because both the time-optimal maneuver to q_f and the interception maneuver are fully defined by the switching times and the initial control input, and all boundary constraints are satisfied for both maneuvers, this must be the case.

Conversely, assume that we have computed a time-optimal maneuver to the position q_f. Then for all intermediate positions occurring after the time t_2, this is the interception maneuver that is optimal with respect to the above optimality conditions (we must limit the intermediate positions to ones occurring after t_2 due to the constant slope of λ_3 permitting only one zero crossing after $\hat{t}$).

The above equivalence of interception maneuvers and time-optimal maneuvers allows us to leverage the trajectory generation algorithm that we have developed for time-optimal maneuvers, as will be seen in Section 5.

4.4 Existence of solutions

While the optimality conditions describe the structure of interception maneuvers that minimize the to-rest time after interception, it remains to verify that a maneuver satisfying the interception constraint (13) exists. Clearly, this will not always be the case: For small values of $\hat{t}$, the available control effort will not suffice to drive the system to $\hat{q}$ in time if $\hat{q}$ differs significantly from the motion dictated by the initial conditions. This is a fundamental difference to the motion to a given end point discussed in Section 3, where all target points could be reached.

The existence of solutions can be formalized as the position $\hat{q}$ (with arbitrary velocity and acceleration) lying in the reachable set at time $\hat{t}$ for the given initial conditions. Using the fact that the reachable set is convex [6], it can be shown that the positions reachable at time $\hat{t}$ are bounded by the two trajectories for which $t_2 = \hat{t}$ (these are trajectories that apply the maximum or minimum possible control effort for the entire duration up to $\hat{t}$). The upper bound is reached by applying $v^* = v_{\max}$ in the interval $[0\ t_1)$, and the lower bound is reached by applying $v^* = -v_{\max}$.

Note that if no solution to the interception problem is found with this strategy, then there exists no other control input that can reach the interception point at time $\hat{t}$ [13]. This implies that, if we assume the decoupled dynamics (4), the strategy presented above provides an interception trajectory whenever the interception constraint (13) can be satisfied by the system dynamics.

5. Computation and Verification of Interception Maneuvers

In this section, we discuss how the properties of optimal interception trajectories can be used to efficiently compute interception maneuvers.

5.1 Computation of maneuvers

The identical structure of the interception maneuver and the time-optimal maneuver (recapitulated in Section 3) permits us to compute maneuvers satisfying the interception constraint in a fashion that is very similar to the one employed for time-optimal maneuvers (see [11]): There is generally no closed-form solution for the five times $t_1 \dots t_f$ from the constraint equations (12)-(14) and the singular arc constraints (19)-(20). It is however straightforward to find solutions using a one-dimensional bisection algorithm. While the computation of a time-optimal maneuver is carried out by iterating over the position at the end of the maneuver until the final position constraint is satisfied, we now iterate over the position at the interception time $\hat{t}$ in order to satisfy the interception constraint.

5.2 Extremal points of the trajectory

In many interception scenarios, it is important to not only satisfy the interception constraint, but to also come to rest within certain space constraints. We have not included such position constraints in the derivations in Section 4 (the trajectory is already fully defined by the chosen constraints and optimality conditions, and additional constraints would complicate the trajectory structure significantly).

However, verifying such constraints once the trajectory has been planned is straightforward: The solution computed by the interception strategy presented herein results in a position trajectory that is piecewise polynomial of at most order three. This makes it simple to compute extremal points by finding points where the velocity vanishes. Such extremal points can then be compared to space restrictions.

5.3 Control effort distribution between the degrees of freedom

All previous derivations were based on a single degree of freedom. In order to intercept a position in 3D, all three degrees of freedom must be able to reach their respective target position by the interception time. To control each degree of freedom's ability to reach the target point in time, the acceleration constraints (16) can be varied. The three acceleration constraints are linked through condition (1). We parameterize the acceleration constraints as follows:

$$\ddot{z}_{\max} = c_z a_{\max} - \mathrm{g} \ , \tag{28}$$

$$\ddot{z}_{\min} = a_{\min} - \mathrm{g} \ , \tag{29}$$

$$\ddot{x}_{\max} = -\ddot{x}_{\min} = c_x \sqrt{a_{\max}^2 - (\ddot{z}_{\max} + \mathrm{g})^2} \ , \tag{30}$$

$$\ddot{y}_{\max} = -\ddot{y}_{\min} = \sqrt{1 - c_x^2} \sqrt{a_{\max}^2 - (\ddot{z}_{\max} + \mathrm{g})^2} \ , \tag{31}$$

where c_x and c_z are parameters. It is straightforward to verify that for all values of $c_x \in [0 \;\; 1]$, $c_z \in [\mathrm{g}/a_{\max} \;\; 1]$, the acceleration constraint (1) is satisfied.

It then remains to find parameters c_x and c_z such that each degree of freedom is able to reach the interception point. We propose a two-step strategy:

1. Starting from $c_z = 1$, find the lowest value c_z for which the vertical degree of freedom is able to both satisfy the interception constraint and not violate space restrictions, and then

2. vary c_x until both horizontal degrees of freedom reach the interception point and remain within space constraints.

The above method finds a three-dimensional interception trajectory that satisfies the interception constraint and possible space restrictions. It may however be beneficial to additionally specify a performance measure for the choice of c_x and c_z. We intend to investigate this in further research.

The above steps complete the description of an algorithm permitting the computation of a three-dimensional interception maneuver that is feasible with respect to the dynamics and constraints of the quadrocopter.

6. Experimental Results

The interception strategy presented herein has been tested in the Flying Machine Arena, an aerial vehicle development platform at ETH Zurich [17]. To demonstrate its performance, we apply it to the problem of “hitting” a ball mid-flight with a quadrotor vehicle.

6.1 Experimental setup

We use modified Ascending Technologies ‘Hummingbird’ quadrocopters [9]. The vehicles are equipped with custom electronics, allowing greater control of the vehicle’s response to control inputs, sensors providing a higher dynamic range, and extended interfaces [17].

The trajectory generation algorithm is run on an off-board desktop computer at a rate of 70 Hz. For each computed trajectory, a command consisting of the three vehicle body rates and the collective thrust is sent to the vehicle through a low-latency 2.4 GHz radio link. The full state information (required for the initial conditions of the trajectory generation) is obtained from a state observer. The state observer receives precise vehicle position and attitude measurements from an infrared motion capture system at a rate of 200 Hz.

The ball that the vehicle is to hit is also tracked by the motion capture system, and its state is estimated using a Kalman filter combined with a drag coefficient estimator [21].

6.2 Determination of ball interception time

The interception algorithm presented above requires the interception time and position as inputs. To intercept the flight path of the ball, the interception time remains to be chosen. It is fixed as follows:

We consider the ball to be flying as soon as it crosses a threshold height. At this point, its flight path is predicted forward based on a first-principles model of its dynamics. The prediction is evaluated in discrete time steps from the current time up to the time at which the ball will fall below a certain height. The interception feasibility is verified for each of the discrete prediction points, with the additional constraint that the entire maneuver must remain within a specified volume. If the ball is interceptable, a range of possible interception times is found, the mean of which is chosen as the interception time to which the planning will occur from this point on. Figure 3 shows a ball being found, and a number of planned candidate interception trajectories. The interception time chosen by the algorithm lies in the middle of the candidate trajectories.

Note that further investigation is required to find the best way to choose the interception time. The method presented here was seen to work well in experiments, but its properties have not been analyzed thoroughly. We intend to investigate this in future work.

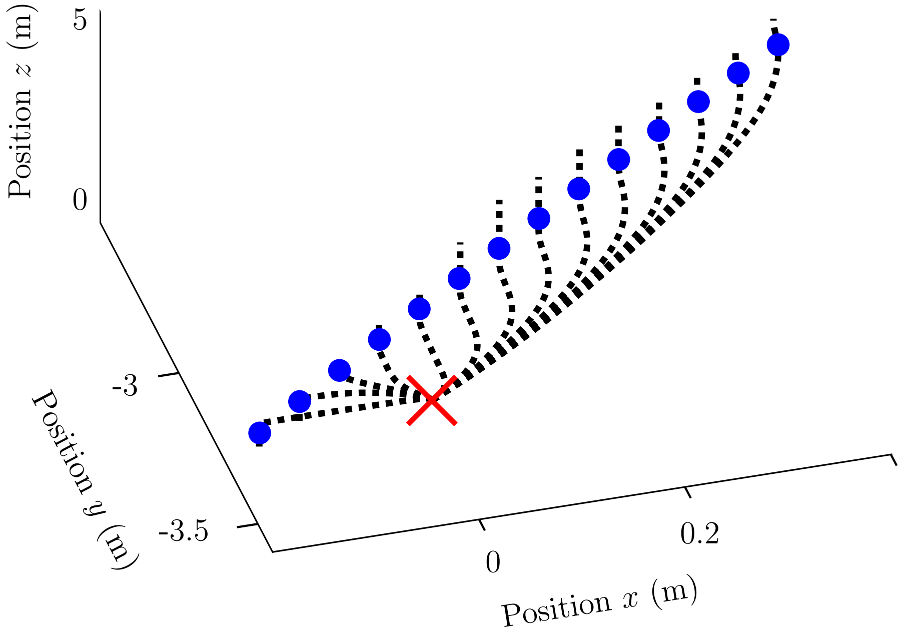

Figure 3. A set of candidate trajectories planned when a ball is first detected. The red cross denotes the initial position of the vehicle. The blue dots are the ball positions predicted at various instances in time. The dotted black lines show the planned flight path for each of the predicted ball positions. The chosen interception time lies in the middle of these points.

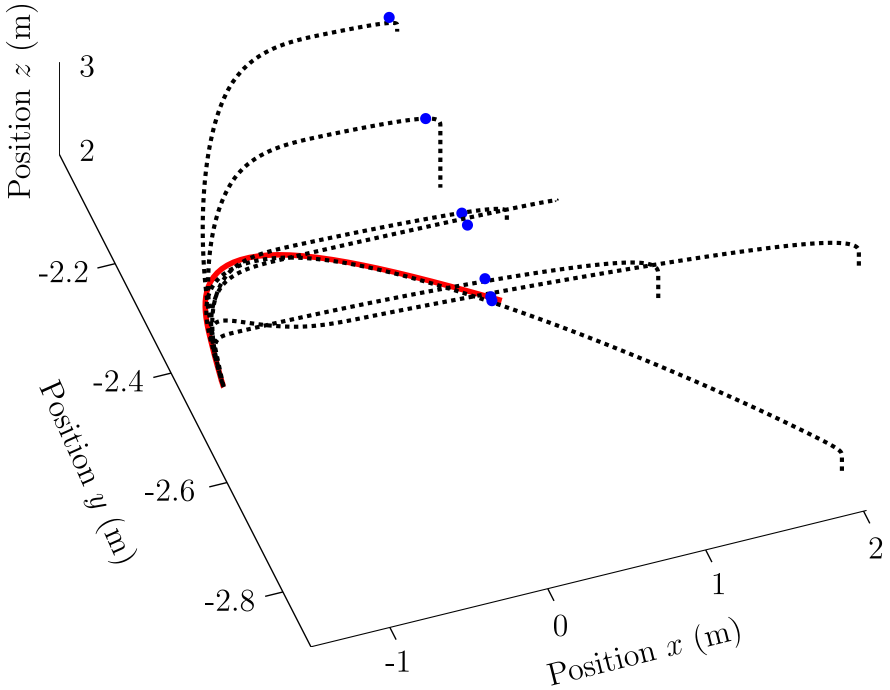

Figure 4. Trajectories planned as the ball state and drag estimates converge. A selection of the planned trajectories is shown here (dotted black lines). The predicted interception location (blue points) changes by about 0.55 m. The updated interception location is accounted for at every re-planning of the trajectory. The solid red line shows the trajectory that was actually flown. The vehicle intercepted the ball at the end of the red line.

6.3 Flight results

Flight tests were carried out with the ball being thrown in varying directions. It was found that the vehicle was able to intercept ping-pong sized balls reliably when a feasible interception trajectory was found. A video showing a number of experiments may be found on the first authors web site.

Figure 4 shows an interception maneuver that highlights the advantage of the real-time capability of the trajectory generation algorithm: At the beginning of the maneuver, the ball state and drag estimates are subject to significant variation. Because the interception trajectory is replanned at every controller update, the moving interception point is naturally considered for each controller update. In this specific example, the interception point changes by about 55 cm over the course of 0.6 s before converging, and the vehicle successfully intercepted the ball.

7. Conclusion & Outlook

This paper introduced an interception trajectory generation algorithm for quadrotor vehicles. It was shown that the problem can be reformulated such that its structure is identical to the trajectory generation for time-optimal trajectories. For this class of trajectory generation problems, an efficient computational method has been developed previously and could be readily adapted. The successful implementation of this interception strategy has been verified experimentally by intercepting balls mid-flight.

In the trajectory design approach presented herein, a number of design parameters remained to be chosen, for example the control effort tradeoff between the three decoupled degrees of freedom. We are planning to investigate ways to systematically choose these such that the best possible performance is achieved.

Furthermore, we intend to investigate strategies for determining the optimal interception time for objects flying along trajectories, where optimality could be defined by criteria such as minimal vehicle velocity at interception time, the shortest time to interception, or other objectives.

An interesting extension of this work would be to include multiple interception constraints. Because the current interception method is computationally very lightweight, we believe that it should be possible to plan such multiple interception trajectories while maintaining real-time capabilities.

References

[1] Berthold Bäuml, Thomas Wimböck, and Gerd Hirzinger. Kinematically Optimal Catching a Flying Ball with a Hand-Arm-System. In *International Conference on Intelligent Robots and Systems*, 2010.

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I.* Athena Scientific, third edition, 2005.

[3] Patrick Bouffard, Anil Aswani, and Claire J Tomlin. Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *International Conference on Robotics and Automation*, 2012.

[4] Y. Bouktir, M. Haddad, and T. Chettibi. Trajectory Planning for a Quadrotor Helicopter. In *Mediterranean Conference on Control and Automation*, 2008.

[5] Arthur Earl Bryson and Yu-Chi Ho. *Applied Optimal Control.* Taylor & Francis, 1975.

[6] S Chang. Minimal Time Control With Multiple Saturation Limits. *IEEE Transactions on Automatic Control*, 8(1):35–42, January 1963.

[7] Ian D. Cowling, Oleg A. Yakimenko, and James F. Whidborne. A Prototype of an Autonomous Controller for a Quadrotor UAV. In *European Control Conference*, 2007.

[8] Carlos E. García, David M. Prett, and Manfred Morari. Model Predictive Control: Theory and Practice - A Survey. *Automatica*, 25(3):335–348, 1989.

[9] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[10] Richard F. Hartl, Suresh P. Sethi, and Raymond G. Vickson. A Survey of the Maximum Principles for Optimal Control Problems with State Constraints. *SIAM Review*, 37(2):181–218, 1995.

[11] Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. In *IFAC World Congress*, 2011.

[12] Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control. *Autonomous Robots*, 33(1-2):69–88, 2012.

[13] Henry Hermes and Joseph P Lasalle. Functional Analysis and Time Optimal Control. *Mathematics in Science and Engineering*, 56, 1969.

[14] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor Helicopter Trajectory Tracking Control. In *Conference on Decision and Control*, 2008.

[15] Jonathan P. How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.

[16] Peter C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

[17] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips. In *International Conference on Robotics and Automation*, 2010.

[18] H. Maurer. On Optimal Control Problems with Bounded State Variables and Control Appearing Linearly. *SIAM Journal Control and Optimization*, 15(3):345–362, 1977.

[19] D. Mellinger, N. Michael, and V. Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[20] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[21] Mark Muller, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Ball Juggling. In *International Conference on Intelligent Robots and Systems*, 2011.

[22] Frieder Stolzenburg, Oliver Obst, and Jan Murray. Qualitative Velocity and Ball Interception. *Lecture Notes in Computer Science - KI2002: Advances in Artificial Intelligence*, 2479:95–99, 2002.

Part B

ITERATIVE LEARNING OF PERIODIC MOTIONS

Paper IV

A Frequency Domain Iterative Learning Algorithm for High-Performance, Periodic Quadrocopter Maneuvers

Markus Hehn and Raffaello D'Andrea

Abstract

Quadrocopters offer an attractive platform for aerial robotic applications due to, amongst others, their hovering capability and large dynamic potential. Their high-speed flight dynamics are complex, however, and the modeling thereof has proven difficult. Feedback control algorithms typically rely on simplified models, with feedback corrections compensating for unmodeled effects. This can lead to significant tracking errors during high-performance flight, and repeated execution typically leads to a large part of the tracking errors being repeated. This paper introduces an iterative learning scheme that non-causally compensates repeatable trajectory tracking errors during the repeated execution of periodic flight maneuvers. An underlying feedback control loop is leveraged by using its set point as a learning input, increasing repeatability and simplifying the dynamics considered in the learning algorithm. The learning is carried out in the frequency domain, and is based on a Fourier series decomposition of the input and output signals. The resulting algorithm requires little computational power and memory, and its convergence properties under process and measurement noise are shown. Furthermore, a time scaling method allows the transfer of learnt maneuvers to different execution speeds through a prediction of the disturbance change. This allows the initial learning to occur at reduced speeds, and thereby extends the applicability of the algorithm for high-performance maneuvers. The presented methods are validated in experiments, with a quadrocopter flying a figure-eight maneuver at high speed. The experimental results highlight the effectiveness of the approach, with the tracking errors after learning being similar in magnitude to the repeatability of the system. The applicability to more complex tasks is demonstrated experimentally by learning a maneuver wherein an inverted pendulum is balanced on the quadrocopter, while a trajectory is tracked.

Under Review for publication in *Mechatronics*.

1. Introduction

Aerial robots serve as platforms for robotic applications that provide numerous benefits, including the ability to move freely in three-dimensional space, and the significantly increased ability to overcome obstacles due to not being limited to motion on the ground. For relatively small platforms that require hovering capabilities, multi-rotor vehicles such as quadrocopters are often the vehicle of choice [53]. Compared to other such platforms, quadrocopters profit from high mechanical robustness due to a minimal number of moving parts [45], safety due to comparatively small rotor size, and high thrust-to-weight ratios allowing high-performance maneuvers as well as the transport of large payloads.

While the use of quadrocopters as robotic platforms was largely confined to research institutions in the past, a growing number of industrial applications are now in the process of being developed and deployed. Examples include aerial imaging for photogrammetry, motion picture production, and journalism [19], environmental monitoring and inspection tasks of hard-to-reach objects such as pipelines, dams, and power lines [54], the creation of ad-hoc antenna networks or arrays [56], as well as disaster coordination [1].

The capability of quadrocopters to perform highly dynamic, complex, and precise motions has been demonstrated repeatedly in recent years (see, for example, [34, 37, 39, 48]). In order to execute such high-performance motions, the commonly used approach consists of using a first-principles model of the quadrotor dynamics to design the nominal maneuver, and a model-based feedback control law to ensure tracking of the nominal trajectory.

Such traditional feedback controllers however have important limitations in high-performance quadrotor applications. While the first-principles models used to design the controllers capture the near-hover behaviour of quadrocopters well, secondary effects become increasingly important when maneuvering speed increases. Examples of such effects are the complex drag and lift behaviour of rotary wings under unsteady inflow conditions [11], the aerodynamic effects of a vehicle moving through the turbulent wake of its propellers [6], and external influences such as wind or ground and wall effects when operating in proximity to the environment [46]. Such effects are not typically accounted for in the maneuver and controller synthesis stage in order to make the design process tractable. The execution then heavily relies on the feedback controller to compensate for potentially significant effects not captured by the nominal dynamics.

In order to improve the tracking performance of quadrocopters under feedback control, a number of researchers have proposed learning schemes. Examples of such schemes include those based on reinforcement learning techniques [8,60] and neural networks [16,17], which are designed to automatically find well-performing control policies, and adaptive control methods [4,40,42] that adapt parameters that are based on modeled disturbances such as payloads, center of mass shifts and external disturbances.

When the motion that we are concerned with is to be executed repeatedly, a further opportunity to improve tracking performance may arise: Many of the disturbances that degrade tracking performance will be similar each time the vehicle performs the motion.

These disturbances can then be compensated for non-causally using data from past executions. Control strategies that exploit available data from past executions in order to improve tracking performance were first proposed in the late 1970s and early 1980s [5,27] for applications in motion control and power supply control. Since then, active research in this field, covering numerous applications and problem formulations (see e.g. [12,15,59], and references therein), has shown it to be a powerful approach for high-performance reference tracking. In extensions to these learning methods, several authors have shown the application of learning control methods to systems with underlying feedback control loops (e.g. [9,14]). In such scenarios, the powerful capability of learning control to non-causally compensate repeatable disturbances is combined with real-time feedback control to correct for non-repetitive noise.

While the application of learning algorithms, and specifically non-causal strategies, to stationary systems is well-established, its use for flying vehicles is less mature and has been actively researched during recent years. Several high-performance maneuvers for multi-rotor vehicles have been demonstrated with the use of learning algorithms. Broadly speaking, the learning approaches used can be categorized in two groups:

The first group is characterized by its ability to learn motions that are parameterized. The motion is thus described by a (finite) set of design parameters, chosen by the user. After the execution of the motion, these parameters are adapted to compensate for disturbances. The direction and magnitude of the correction may be model-based, or based on the user's intuition. A discussion on the importance of choosing 'good' design parameters may be found in [31], where a learning algorithm for this kind of parameterized motions is demonstrated for multiple flips and fast translations with quadrocopters. A further demonstration of this class of learning algorithms is provided in [36]. The ability to shape the tracking performance strongly depends on the number of parameters that are optimized; in the above examples, the objective is to minimize the error at specific time instants ('key frames'), and a relatively small number of parameters is sufficient to do so. This makes the methods computationally lightweight.

The second group of learning approaches considers more generic motions that need not be specified by parameters. The system dynamics are considered in discrete time, and the correction consists of correction values (typically control inputs or set points) for each discrete time step. After execution of the motion, a numerical optimization over the correction values is performed in order to minimize a metric related to the tracking error. In this optimization, a model of the system dynamics provides the mapping from corrections to the tracking error. This approach is commonly known as a form of iterative learning control [12], and its application to high performance quadrocopter flight has been demonstrated [38,43,47,51].

The delimitation between the two groups is not strict. Indeed, the second group of learning approaches could be seen as using a very large number of values to parameterize the correction.

The algorithm presented in this paper can be characterized to be a form of repetitive control [59] in that it is a technique for non-causally compensating repeated tracking errors in the execution of periodic motions. Algorithms of this form have previously shown good performance when applied to related problems where aerodynamic disturbances are considered, in particular the rejection of periodic wind disturbances on wind turbines [7, 24].

Similar to the second group of learning algorithms, we do not assume a parameterized motion. However, we reduce the dimensionality of the corrections that we intend to learn by assuming that they are periodic. This allows us to parameterize the corrections as the coefficients of a truncated Fourier series. The order of the Fourier series provides a means to trade off computational complexity and the ability to compensate for temporally local or high-frequency disturbances. Furthermore, the approach can be considered to be conceptually similar to the one presented by [31], which presents an adaptation strategy to correct for state errors at discrete points in time of parameterized motion primitives. However, we consider periodic errors (instead of errors at specific points in time), and do not require parameterization of the maneuver.

The contribution of this paper to the field of quadrocopter control lies in the application of methods from the fields of repetitive control and iterative learning control to quadrocopters. A general framework for arbitrary periodic motions is presented. We demonstrate how a feedback controller can be leveraged to shape the closed-loop dynamics of the quadrotor system, and show that a linear time-invariant approximation of the closed-loop dynamics suffices to guide the learning process. Using statistical properties of the disturbance, measurement noise, and the influence of nonlinearities, we derive the optimal inter-execution learning update step size. The validity of the approach and its performance is investigated through experiments in the ETH Flying Machine Arena, both with a quadrocopter under normal position control and with a task-specific controller stabilizing a more complex system consisting of a quadrocopter balancing an inverted pendulum.

Furthermore, this paper introduces a novel method that extends the applicability of the repetitive control approach when the reference trajectory is too fast to be learnt directly, for example because the initial execution fails entirely. The core idea here lies in providing an improved initial guess of the disturbances degrading tracking performance. This typically enables learning of the trajectory because the errors are sufficiently small for the first-principles model to provide reliable information on how to compensate. To find the improved initial guess, we introduce a time scaling method that allows initial learning to occur at reduced maneuvering speeds and the transfer of learnt corrections from the reduced-speed execution to full speed. This method may also be applied to other complex dynamic systems where it is necessary to limit initial tracking errors in order to avoid the system failing. The time-scaling method provides an interesting alternative to methods that rely on aborting trials when the errors grow too large [51] in that the tracking error is always learnt over the entirety of the maneuver, and over more general methods to extend the motion [47] due to its computational simplicity.

Preliminary results of this method were presented at international conferences [22,23], and this paper extends these results through the computation of the optimal learning rate

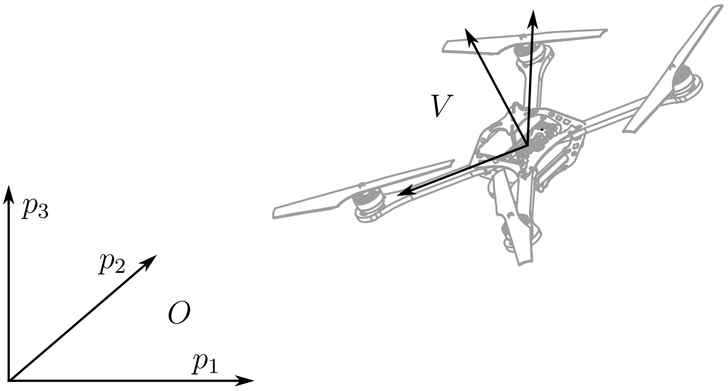

Figure 1. The inertial coordinate system O and the vehicle coordinate system V, used to describe the dynamics of the quadrocopter.

for given statistical properties of the process, a novel way to predict disturbances when transferring knowledge between different execution speeds, as well as providing an in-depth experimental analysis of the method.

The remainder of this paper is structured as follows: We introduce the model of the quadrocopter and the used feedback law in Section 2. The learning algorithm is then presented and analyzed in Section 3. Section 4 presents experimental results highlighting the performance of the algorithm. Section 5 discusses advantages and restrictions of the learning algorithm, and Section 6 provides a conclusion.

2. Quadrocopter Dynamics and Closed-Loop Control

This section first introduces the first-principles model of a quadrocopter, along with a discussion of the accuracy of the model. Furthermore, we introduce the input-output linearizing feedback controller used to control the vehicle. The combination of vehicle and feedback controller form the closed-loop dynamics that the iterative learning algorithm is then applied to. For ease of notation, vectors are expressed as n-tuples $(x_1, x_2, ...)$ where convenient, with dimension and stacking clear from context.

2.1 Quadrocopter Dynamics

The quadrocopter is modeled as a rigid body with six degrees of freedom: its position (p_1, p_2, p_3) in the inertial coordinate system O; and its attitude, represented by the rotation matrix ${}^{O}_{V}R$ between the inertial coordinate system O and the body-fixed coordinate system V, as shown in Figure 1.

The quadrotor vehicle incorporates four actuators, consisting of motors with fixed-pitch propellers. Each motor produces a thrust force and a drag torque, and the resulting rotational dynamics in the body-fixed coordinate frame V are [49]

$$I\dot{\omega} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ \zeta(F_1 - F_2 + F_3 - F_4) \end{bmatrix} - \omega \times I\omega \tag{1}$$

where $\omega = (\omega_x, \omega_y, \omega_z)$ is the rotational rate of the vehicle, I is its rotational inertia, L the arm length of the vehicle, ζ the drag-to-thrust ratio of the propeller, and F_1 to F_4 are the individual thrust forces of each propeller. The total mass-normalized thrust produced by the four propellers is

$$a = \frac{1}{m}\left(F_1 + F_2 + F_3 + F_4\right) \tag{2}$$

where m denotes the mass of the vehicle. The rotational kinematics are given by the first-order differential equation of the rotation matrix [26]

$${}^{O}_{V}\dot{R} = {}^{O}_{V}R \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3}$$

The translational dynamics of the vehicle, expressed in the inertial coordinate system O, are

$$\begin{bmatrix} \ddot{p}_1 \\ \ddot{p}_2 \\ \ddot{p}_3 \end{bmatrix} = {}^{O}_{V}R \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} \tag{4}$$

where g denotes gravitational acceleration.

This first-principles model of the quadrocopter rotational and translational dynamics is commonly used to design and analyze algorithms such as feedback control laws and path planners (see, for example, [33] and references therein), and captures near-hover dynamics well. When higher maneuvering speeds are reached, however, a multitude of additional – mainly aerodynamic – effects become more significant. A number of these effects have been identified and incorporated into more refined models; for example, they include induced translational and rotational rotor drag [11], blade flapping [44], and dominant propellers tip vortices during vertical descent flight [6]. Furthermore, the model neglects potential interactions of the vehicle with the environment. For example, it is well known that flight dynamics are influenced significantly by ground and wall effects [46], fast maneuvers can cause the vehicle to fly through its own wake, and external disturbances such as wind effects [2] can cause large disturbances.

While the modeling of such effects has provided valuable insight, their incorporation into control strategies has generally been slow due to the highly complex nonlinear models making controller design significantly more difficult, and the added difficulty of identifying the parameters of such models. Instead, most control laws treat such effects as disturbances, relying on feedback control to account for them implicitly. In this paper, we will follow a similar approach in that we do not model the effects, but we will compensate for them non-causally during the repeated execution of periodic motions.

2.2 Feedback Control

Within this paper, we assume that an existing feedback control law is used to stabilize the quadrocopter and track set points. The feedback control law was described and analyzed in more detail in [50], and is taken as a given in this paper. It consists of cascaded feedback linearizing control loops for position, attitude, and rotational rates as follows:

Position Control For all three degrees of freedom, a feedback control law determines the desired acceleration $\hat{\ddot{p}}_i$ from the position and velocity errors such that the loop is shaped to the dynamics of a second-order system with time constant τ_i and damping ratio ζ_i:

$$\hat{\ddot{p}}_i = \frac{1}{\tau_i^2}\left(\hat{p}_i - p_i\right) - 2\frac{\zeta_i}{\tau_i}\dot{p}_i \quad \text{for } i = \{1, 2, 3\} \tag{5}$$

where $\hat{p}_i$ is the commanded position.

With R_{xy} denoting the x-th element of the y-th column of ${}^{O}_{V}R$, the thrust is computed to enforce the desired vertical acceleration according to (4):

$$a = \frac{1}{R_{33}}\left(\hat{\ddot{p}}_3 + \mathrm{g}\right) \ . \tag{6}$$

Reduced Attitude Control The desired rotation matrix entries $\hat{R}_{13}$ and $\hat{R}_{23}$ for the given desired accelerations are computed from (4), (5) and (6) to be

$$\hat{R}_{13} = \frac{\hat{\ddot{p}}_1}{a} \text{ and } \hat{R}_{23} = \frac{\hat{\ddot{p}}_2}{a} \ . \tag{7}$$

The attitude control loop is shaped such that the rotation matrix entries R_{13} and R_{23} react in the manner of a first-order system with time constant τ_{rp} by computing the desired derivative of the rotation matrix elements:

$$\hat{\dot{R}}_{i3} = \frac{1}{\tau_{rp}}\left(\hat{R}_{i3} - R_{i3}\right) \text{ for } i = \{1, 2\} \ . \tag{8}$$

Inverting the rotational kinematics (3), this is converted to the commanded rotational rates about the first two body axes of the vehicle:

$$\begin{bmatrix} \hat{\omega}_x \\ \hat{\omega}_y \end{bmatrix} = \frac{1}{R_{33}} \begin{bmatrix} R_{21} & -R_{11} \\ R_{22} & -R_{12} \end{bmatrix} \begin{bmatrix} \hat{\dot{R}}_{13} \\ \hat{\dot{R}}_{23} \end{bmatrix} . \tag{9}$$

The commanded rotational rate about the third body axis, $\hat{\omega}_z$, can be determined separately as it does not influence the translational dynamics of the vehicle. We employ a proportional controller on the Euler angle describing the vehicle heading.

Body Rate Control Using the desired body rates $\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$ as commands, the body rate controller is designed to follow the commands in the fashion of three decoupled first-order systems. In order to achieve this, the rotational dynamics (1) are inverted:

$$\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ \zeta(F_1 - F_2 + F_3 - F_4) \end{bmatrix} = I \begin{bmatrix} (\hat{\omega}_x - \omega_x)/\tau_{pq} \\ (\hat{\omega}_y - \omega_y/\tau_{pq} \\ (\hat{\omega}_z - \omega_z)/\tau_r \end{bmatrix} + \omega \times I\omega \tag{10}$$

with the time constants τ_{pq} for the first two axes, and τ_r for the third axis. The above equation, in combination with the total thrust constraint resulting from combining equations (2) and (6), define the four individual propeller forces F_1 to F_4, and thereby complete the feedback control design.

Trajectory Feed-Forward The presented controller can readily be augmented to apply feed-forward velocities, accelerations, rotation rates, and rotational accelerations for known input trajectories by extending Equations (5), (8), (9), and (10) with the corresponding feed-forward terms derived from the derivatives of the nominal position trajectory $\hat{p}(t)$. A discussion of the effects and performance benefits thereof can be found in [38], where it is shown that feed-forward commands can improve tracking performance, though large systematic errors remain. This can be explained by the fact that these feed-forward terms are model-based, and the unmodeled effects discussed in Section 2.1 cause significant model mismatch that is not accounted for.

While many applications profit from additional feed-forward terms, we found them unnecessary in conjunction with the learning method presented herein. This is because they do not improve the repeatability of the flight performance, and the learning algorithm compensates for systematic tracking errors almost entirely. Indeed, the learning algorithm can be considered to be providing the necessary feed-forward signal to make the system track the reference trajectory accurately, and thereby also captures conventional feed-forward terms.

2.3 Approximate Closed-Loop System Dynamics

The learning algorithm that will be introduced in Section 3 relies on an approximation of the system dynamics in the form of a linear time-invariant (LTI) system.

The feedback control design is based on cascaded control loops that are designed using a loop shaping approach. We assume time scale separation between the control loops (i.e., $\tau_{rp} \ll \tau_{12}$ and $\tau_{pq} \ll \tau_{rp}$) and approximate the closed-loop dynamics to depend only on the position control loops. The nominal dynamics of the closed-loop system from a position set point $\hat{p}$ to the vehicle position p can then be approximated by three decoupled LTI second-order systems:

$$\ddot{p}_i \approx \frac{1}{\tau_i^2}\left(\hat{p}_i - p_i\right) - 2\frac{\zeta_i}{\tau_i}\dot{p}_i \text{ for } i = \{1, 2, 3\} \tag{11}$$

with time constant τ_i and damping ratio ζ_i. We will use these nominal closed-loop dynamics in the iteration-domain learning algorithm.

More accurate characterizations of the closed-loop dynamics could be used in the learning algorithm, e.g. by including the underlying control loops such as the attitude control (7)-(9). However, our experiments showed that the low-order model was sufficient to guide the iterative learning process as long as very high frequencies are not considered.

3. Learning Algorithm

This section introduces the learning algorithm that is applied to the quadrocopter system in order to compensate for systematic disturbances that deteriorate flight performance during the execution of periodic motions. The fundamental idea is to use data from past executions in order to characterize the tracking errors, and to then compensate for them in a non-causal manner during following executions. For this compensation, we leverage the prior knowledge of the dominant dynamics of the quadrocopter under feedback control, and combine this knowledge with measurement data from experiments in order to determine appropriate compensations to apply during the next execution. Compared to the use of pure feedback control, this scheme can improve the tracking performance because repeated disturbances are compensated for non-causally, whereas pure feedback control is limited to causal corrections.

The basic system structure used in the learning algorithm is depicted in Figure 2. The closed-loop dynamics of the quadrocopter under feedback control remain unmodified by the learning algorithm. We use the approach of adapting the set point of the controller, also called a serial architecture [12] or indirect learning-type control [58, 59]. The controller set point is augmented by adding a correction input $u(t)$ to it. We assume that we can derive a linear time-invariant (LTI) approximative model of the closed-loop dynamics between the position tracking error and the correction input (as done in Section 2.3 for the presented feedback control law). In comparison to modifying the control inputs of the quadrocopter directly, the serial architecture offers the advantage that the dynamics from a change in the set point to a change in the tracking error output are those of the closed loop system, which are designed to be approximately LTI. Due to the periodic

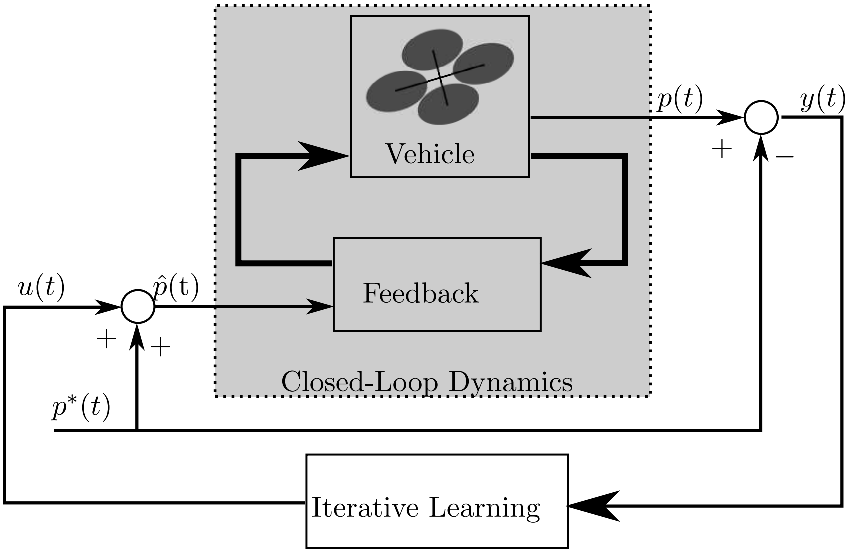

Figure 2. The learning architecture. The iteration-domain feedback law uses the tracking error as input, and its output is a set point shift to the real-time feedback law.

nature of the motions addressed in this paper, we use Fourier series [25] to characterize correction input and tracking error output signals of the system.

The learning algorithm builds upon this LTI model when interpreting the measurement data of an execution in order to determine the appropriate correction input signal for the next execution, as detailed in Section 3.2. Using the statistical properties of the relevant disturbance signals, it is possible to derive the optimal correction input signal in a least-squares sense (Section 3.3). Safer learning of high-performance maneuvers can be achieved by reducing the execution speed during the initial learning phase, and using the disturbance characteristics identified at lower speeds to provide an improved disturbance prediction at high speeds (Section 3.4).

3.1 System Model

This section introduces the quantities used by the learning algorithm. Because the maneuvers considered herein are of periodic nature, time signals are parameterized as Fourier series with fundamental frequency $\Omega_0 = 2\pi/T$, with T being the period of the maneuver. Furthermore, the notation $(\cdot)^i$ is used to denote quantities of iteration i of the learning algorithm. Unless stated otherwise, we assume that time signals are periodic with period T.

The error output measurement $y(t)$ is the signal capturing the tracking errors which should be eliminated by the learning algorithm. The dimension of $y(t)$ is not defined by the problem, and may be chosen by the user as the set of error outputs that should be minimized. For the specific implementation considered in this paper, it consists of the three-dimensional deviation of the vehicle position from its reference trajectory $p^*(t)$, as seen in Figure 2. We assume that the measured output of an experiment can be decomposed as follows, where the components will be explained in the following:

$$y^i(t) = d^i(t) + h^i(t) + n^i(t)\,. \tag{12}$$

The first component $d^i(t)$ represents the systematic tracking error that should be compensated. It is not known a priori, but we will later assume that a statistical description of $d^0(t)$ is available for the derivation of the optimal learning step size in Section 3.3. Specifically, we will then assume that its mean is zero, and that it is stationary with a known autocorrelation function. The evolution of the tracking error over several iterations is modeled as

$$d^{i+1}(t) = d^i(t) + \gamma^i(t) \tag{13}$$

where $\gamma^i(t)$ models slight changes to the disturbance from iteration to iteration as a trial-uncorrelated sequence of zero-mean stationary noise, where we will again assume a known autocorrelation function in Section 3.3.

The second component, $h^i(t)$, is the response of the closed-loop dynamic system to the correction input $u^i(t)$, characterized in the frequency domain by

$$H^i(\omega) = G(\omega)U^i(\omega) \tag{14}$$

with $G(\omega)$ being the transfer function [13] of the LTI approximation of the closed-loop system, and $H^i(\omega)$ and $U^i(\omega)$ being the frequency domain representations of $h^i(t)$ and $u^i(t)$, respectively. The final component of the tracking error measurement, $n^i(t)$, represents trial-uncorrelated, non-periodic noise. It is assumed to vary for each execution of the maneuver, and captures effects such as non-repeatable disturbances to the vehicle (for example wind gusts), but also measurement noise. The non-repeatable disturbances are assumed to be zero mean; a non-zero mean would represent a repeatable disturbance and would therefore be accounted for by $d^i(t)$. Furthermore, we assume stationarity, and for the derivation of the optimal learning step size a known autocorrelation function.

The goal of the learning algorithm is to choose the correction input $u^i(t)$ such that the output $y^i(t)$ is minimized. The desired output is assumed to be periodic with period T, and the algorithm will use measurements of previous executions $y^{i-1}(t), y^{i-2}(t), \ldots$ in order to determine $u^i(t)$. Intuitively, this task implies finding the correction input $u^i(t)$ such that $h^i(t) + d^i(t)$, is minimized, meaning that the unwanted repeated disturbance is canceled out in the output (12) as well as possible.

Note that the dimensions of $u(t)$ and $y(t)$ are not necessarily given by the system model, but can be chosen by the user to indicate which signals are relevant to the tracking task at hand, and which quantities can be modified. We will, however, assume that the dimension of $u(t)$ is no less than that of $y(t)$, implying that we have at least as many compensation inputs available as we have error quantities. If this is not the case, full tracking can typically not be achieved, although it can be shown that the tracking error can still be reduced [23]. In the specific implementation of the algorithm considered in this paper, $u(t)$ is a three-dimensional additive correction to the position control reference point and $y(t)$ is the three-dimensional deviation of the vehicle from the reference trajectory, as seen in Figure 2.

3.2 Learning Update Law

Due to the periodic nature of the maneuvers considered herein, we will leverage Fourier series decompositions of correction input and error output signals. We parameterize the correction input by a truncated Fourier series of order N and fundamental frequency Ω_0:

$$u^i(t) = r_0^i + \sum_{k=1}^{N} r_k^i \cos(k\Omega_0 t) + \sum_{k=1}^{N} s_k^i \sin(k\Omega_0 t) \tag{15}$$

where $\Omega_0 = 2\pi/T$ is the fundamental frequency of the series. The frequency domain representation of $u(t)$ is then

$$U^i(0) = r_0^i \tag{16}$$

$$U^i(k\Omega_0) = r_k^i - js_k^i \quad \text{for } k = 1, 2, \ldots, N\,. \tag{17}$$

The system response $h^i(t)$ to this Fourier series $u^i(t)$ is also a Fourier series of order N [25], which is defined in the frequency domain through the relationship

$$H(k\Omega_0) = G(k\Omega_0)U(k\Omega_0) \quad \text{for } k = 0, 1, \ldots, N\,. \tag{18}$$

We will now invert this relationship in order to use it as a learning feedback law that compensates a given disturbance. Assuming that $G(k\Omega_0)$ is full rank, let $G^+(k\Omega_0)$ denote the Moore-Penrose pseudoinverse [10] of $G(k\Omega_0)$ (in the special case that G is square, $G^+(k\Omega_0) = G^{-1}(k\Omega_0)$ holds).

Assume that we have executed the trajectory for iteration i, and measured the tracking error $y^i(t)$. Let $Y^i(k\Omega_0)$ denote the Fourier series that coincides with $y^i(t)$ for $t \in [0\ \ T]$. The iteration feedback law is then given by the correction Fourier series

$$U^{i+1}(k\Omega_0) = U^i(k\Omega_0) - \mu_k^i G^+(k\Omega_0)\, Y^i(k\Omega_0) \tag{19}$$

for $k = 0 \ldots N$, with the step size μ_k^i controlling the adaptation rate. The time signal $u^{i+1}(t)$ is then constructed from $U^{i+1}(k\Omega_0)$ and applied to the system in the next iteration.

3.3 Learning Step Size

The remaining degree of freedom in the learning update law (19) is the step size μ_k^i. Depending on the application scenario and, more specifically, the availability of the statistical properties of the disturbance signals $d^0(t)$, $n^i(t)$ and $\gamma^i(t)$, the step size μ_k^i may be considered to be a tuning parameter that is directly set by the user, or its optimal value may be computed from the available statistical properties.

Noise Influence on Fourier Coefficients In practice, the Fourier series coefficients of $y^i(t)$ are estimated from a set of discrete-time observations of the continous-time signal, which are used to compute a discrete Fourier Transform [41]. The influence of noise such as $n^i(t)$ on the coefficients of a discrete Fourier Transform has been studied in the past [52]: It was shown that, for a sufficiently large number of samples of $y^i(t)$, the noise on the Fourier coefficients will be additive, approximately Gaussian regardless of the distribution of the noise, and zero mean. A more rigorous description of the statistical properties of the coefficients can be found in [52], and the variance of the individual coefficients can be computed from the correlation function of $n^i(t)$. The specific equations are not repeated here, but it is sufficient for our purposes to state that the variance of the coefficients is known, and that they are approximately uncorrelated for a sufficiently large number of discrete-time observations.

Using $N^i(k\Omega_0)$ to denote the additive zero-mean coefficient noise caused by $n^i(t)$, the tracking error Fourier series coefficients at iteration $i+1$ can be written as

$$Y^{i+1}(k\Omega_0) = D^{i+1}(k\Omega_0) + H^{i+1}(k\Omega_0) + N^{i+1}(k\Omega_0)\,. \tag{20}$$

Dropping the argument $k\Omega_0$ for notational convenience (all quantities in the following are parameterized by $k\Omega_0$), we expand the above using Equations (13), (18), and (19):

$$\begin{aligned} Y^{i+1} &= D^{i+1} + GU^{i+1} + N^{i+1} && (21)\\ &= D^i + \Gamma^i + G\left(U^i - \mu_k^i G^+ Y^i\right) + N^{i+1} && (22) \end{aligned}$$

where Γ^i, analogously to N^{i+1}, is zero-mean coefficient noise caused by the disturbance change $\gamma^i(t)$ as defined in Equation (13).

Stability We first consider μ_k^i a tuning parameter, and derive stability conditions for constant values $\mu_k^i = \bar{\mu}_k$. We rewrite Equation (22) using Equations (18), (20) and the fact that G^+ is the right inverse of G:

$$\begin{aligned} Y^{i+1} &= D^i + \Gamma^i + GU^i - \bar{\mu}_k Y^i + N^{i+1} && (23)\\ &= \Gamma^i + (1 - \bar{\mu}_k)\, Y^i - N^i + N^{i+1}\,. && (24) \end{aligned}$$

The expected value of the Fourier coefficients is then

$$\mathrm{E}\left[Y^{i+1}\right] = \left(1 - \bar{\mu}_k\right) \mathrm{E}\left[Y^{i}\right] \tag{25}$$

since the noise terms are zero mean as discussed in Section 3.3. Note that

$$\mathrm{E}\left[Y^{0}\right] = 0 \tag{26}$$

since it is assumed that D^0 is zero mean and the expected value of the Fourier coefficients Y^i is therefore zero for all i. If the algorithm is incorrectly initialized or D^0 is not zero mean, the tracking error still converges to zero in expectation for step sizes $0 < \bar{\gamma}_k \leq 1$. Furthermore, the variance of the Fourier coefficients can be computed by rewriting Equation (23) using (18) and (20):

$$Y^{i+1} = \left(1 - \mu_k^i\right)\left(D^i + H^i\right) + \Gamma^i - \mu_k^i N^i + N^{i+1}\,. \tag{27}$$

Taking the variance and using the fact that $(D^i + H^i)$, Γ^i, N^i and N^{i+1} are independent, it follows that

$$\begin{aligned}
\mathrm{Var}\big[Y^{i+1}\big] &= \left(1 - \bar{\mu}_k\right)^2 \mathrm{Var}\big[D^i + H^i\big] + \mathrm{Var}\big[\Gamma^i\big] \\
&\quad + \bar{\mu}_k^2 \mathrm{Var}\big[N^i\big] + \mathrm{Var}\big[N^{i+1}\big] && (28)\\
&= \left(1 - \bar{\mu}_k\right)^2 \left(\mathrm{Var}\big[D^i + H^i\big] + \mathrm{Var}\big[N^i\big]\right) \\
&\quad + \mathrm{Var}\big[\Gamma^i\big] + \left(2\bar{\mu}_k - 1\right) \mathrm{Var}\big[N^i\big] + \mathrm{Var}\big[N^{i+1}\big] && (29)\\
&= \left(1 - \bar{\mu}_k\right)^2 \mathrm{Var}\big[Y^i\big] + \mathrm{Var}\big[\Gamma^i\big] \\
&\quad + \left(2\bar{\mu}_k - 1\right) \mathrm{Var}\big[N^i\big] + \mathrm{Var}\big[N^{i+1}\big] && (30)
\end{aligned}$$

from which it follows that the variance does not diverge for step sizes $0 < \bar{\mu}_k \leq 1$, assuming bounded variance of N and Γ.

Because the Fourier series coefficients are approximately Gaussian, their mean and variance fully describe their approximate distribution. Assuming that the approximation as a Gaussian distribution is sufficiently accurate, the learning update law results in the Fourier series coefficients of the tracking error being zero mean with bounded variance for constant step sizes $0 < \bar{\mu}_k \leq 1$.

It is important to note that this result only holds for the Fourier series coefficients $k = 0, 1, \ldots, N$ of $Y(k\Omega_0)$. No adaptation occurs for higher-frequency components, and the coefficient dynamics for $k > N$ are therefore simply those of Equation (22) without the input:

$$Y^{i+1} = D^i + \Gamma^i + N^{i+1}\,. \tag{31}$$

Minimum Mean Square Error Step Size We now consider time-varying step sizes μ_k^i, and derive the step size sequence that minimizes the trace of the tracking output variance, also called the minimum mean square estimate [3]. Computing the trace of the variance analogously to Equation (30) from Equation (27), and setting the derivative with respect to μ_k^i to zero, it follows that the optimal step size $\overset{*}{\mu}_k{}^i$ is

$$\overset{*}{\mu}_k{}^i = \frac{\mathrm{Tr}[\mathrm{Var}[Y^i(k\Omega_0)]] - \mathrm{Tr}[\mathrm{Var}[N^i(k\Omega_0)]]}{\mathrm{Tr}[\mathrm{Var}[Y^i(k\Omega_0)]]} \tag{32}$$

where $\mathrm{Tr}[\cdot]$ denotes the trace operator, and the argument $(k\Omega_0)$ is stated explicitly in order to highlight that the optimal step size may differ for each multiple of the fundamental frequency. The output variance $\mathrm{Var}[Y^i]$ can be computed recursively from Equation (27), starting with the statistical properties of $d^0(t)$ and $n^0(t)$ to determine $\mathrm{Var}[Y^0]$. Note that the step size sequence $\overset{*}{\mu}_k{}^i$ only depends on the statistical properties of the random signals $d^0(t)$, $\gamma^i(t)$ and $n^i(t)$ and not on their actual realizations, and can therefore be entirely precomputed before starting experiments if desired.

3.4 Time Scaling for High Performance Maneuvers

When initializing the learning of a maneuver, an initial guess of the compensation input $u^0(t)$ is used to execute the first iteration. When no other information is available, a typical 'naive' choice would be to simply choose $u^0(t) = 0$. However, for high performance motions that approach the feasibility limits of the system (e.g., due to actuator saturation), the naive initial guess may be so poor that it becomes impossible to apply the learning algorithm successfully. This may be caused, for example, by the tracking errors growing large enough to cause the vehicle to collide with its environment, invalidating the error measurement. Furthermore, tracking errors may be so large that the approximate dynamics (18) used to compute the correction input no longer accurately predict the behaviour of the closed-loop control system, leading to instabilities in the learning algorithm.

To allow the algorithm to be applied to such maneuvers, we extend it by introducing a speed scaling factor λ, giving control over the execution speed of the maneuver. The core idea is that most motions become 'easier' to execute when the motion duration is increased (some examples of this were given in [49]), where easier loosely refers to the amplitude of the required control inputs. The speed scaling factor allows the motion to be initially executed and improved at relatively low speeds, where there is no danger of collisions and the learning algorithm works reliably, and to then use the learnt compensation inputs to generate an improved initial guess of the compensation input $u(t)$ for higher execution speeds. Since the initial tracking errors should be lower due to the better initial guess, it is more likely for the learning algorithm to successfully compensate for errors as the initial execution is more likely to be successful, and the approximate dynamics (18) are then only used to compensate for relatively small errors.

We define the scaled maneuver duration

$$T_\ell = \frac{T}{\lambda_\ell} \tag{33}$$

and assume that we have successfully learnt the motion for the execution speed λ_1, i.e. the tracking error output $y_1(t)$ associated with the execution speed λ_1 is sufficiently small for all $t_1 \in [0\, T_1]$. The objective is then to use the learnt compensation input at speed λ_1, which we denote $u_1(t)$, to initialize the compensation input for the (typically higher) execution speed λ_2, which we analogously denote $u_2(t)$.

Ideally, the values for $u_2(t)$ should result in a tracking error at the new execution speed $y_2(t)$ that is as small as $y_1(t)$. This would require knowing how the disturbance $d(t)$ changes with the time scaling, such that $u(t)$ can be chosen accordingly as seen in Equation (12). However, due to $d(t)$ capturing unmodeled disturbances, a model of its dependence on execution speed is not readily available.

An obvious choice for the transfer between two maneuver speeds is to keep the learnt input corrections coefficients and simply re-map them to the corresponding frequencies (i.e., $U_2\left(k\lambda_2\Omega_0\right) = U_1\left(k\lambda_1\Omega_0\right)$). However, the varying sensitivity of the closed-loop transfer function at the two different frequencies, as well as the changing disturbances, could potentially lead to large errors.

In order to account for this, we use a linear extrapolation method (see e.g. [18]) to predict the correct value of the compensation term $H(\lambda k\Omega_0)$ in Equation (20) from past time scaling changes. The first-order prediction of $H(\lambda_3 k\Omega_0)$ for the execution speed λ_3 from the past execution speeds λ_1 and λ_2 is then

$$H(\lambda_3 k\Omega_0) = H(\lambda_2 k\Omega_0) + \frac{\lambda_3-\lambda_2}{\lambda_2-\lambda_1}\left(H(\lambda_2 k\Omega_0) - H(\lambda_1 k\Omega_0)\right) \tag{34}$$

which we can expand using Equation (14) to find the initial guess for the correction Fourier series coefficients:

$$\begin{aligned} U(\lambda_3 k\Omega_0) = {} & G^{+}(\lambda_3 k\Omega_0)\left(G(\lambda_2 k\Omega_0)U(\lambda_2 k\Omega_0)\left(1+\frac{\lambda_3-\lambda_2}{\lambda_2-\lambda_1}\right)\right. \\ & \left. -G(\lambda_1 k\Omega_0)U(\lambda_1 k\Omega_0)\frac{\lambda_3-\lambda_2}{\lambda_2-\lambda_1}\right) . \end{aligned} \tag{35}$$

The prediction requires the storage of the learnt correction series coefficients of the past two execution speeds ($U(\lambda_1 k\Omega_0)$ and $U(\lambda_2 k\Omega_0)$), as well as the respective execution speeds (λ_1 and λ_2). For the first prediction, only one past set of learnt series coefficients is available, and we therefore resort to a zeroth-order prediction by eliminating the second summand in Equation (34).

3.5 Design Parameters

The learning algorithm presented in this section comprises a number of parameters that are user-defined, and can be modified to influence the learning performance:

1. The order of the compensation Fourier series N fundamentally determines the frequencies of the error output $y(t)$ that are compensated for since the compensation input $u(t)$ is characterized by a truncated Fourier series of order N, and no adaptation occurs at higher frequencies. While an upper bound for N is in principle only given by the discrete-time measurements used to estimate the coefficients of $y(t)$, the high-frequency components of the tracking error are often inherently small and can therefore be neglected by choosing a lower order. If the approximate model (18) only captures the underlying closed-loop dynamics well for low frequencies, it may also be beneficial to avoid learning at higher frequencies by limiting the series order.

2. The iteration- and order-dependent learning step size μ_k^i can be treated as a tuning parameter chosen by the user, or its optimal value may be derived from the statistical properties of $n(t)$, $d^0(t)$, and $\gamma(t)$. While the explicit statistical properties of these signals may be difficult to identify for an experimental platform, they can also be considered design parameters. In this context, the properties of $\gamma(t)$ capture the likely change of the disturbance from iteration to iteration due to unmodeled dynamics (for example, nonlinearities of the underlying model), and can therefore be used to encode model uncertainty. A typical choice would be to assume the disturbance changes to be large initially to account for significant changes in the correction input that could highlight nonlinear behaviour, and smaller changes as the algorithm converges. The properties of the non-repetitive noise term $n(t)$ encode measurement noise and the level of repeatability of the experiment, and large values thereof enforce smaller step sizes (as seen in Equation (32)), reducing the level of 'trust' in the measurements. The statistical properties of $d^0(t)$ capture the confidence in the initial guess of what the systematic disturbances are, and thereby influences the step size during the initial phase of the learning. Assuming little prior knowledge implies the coefficients of $D^0(k\Omega_0)$ having very large variance, which in turn implies that the first step size $\overset{*}{\mu}^0$ will approach 1 according to Equation (32).

3. Finally, when time scaling is applied, the time scale sequence $\lambda_0, \lambda_1, \ldots$ remains to be chosen. This is typically done based on past experiments, where the performance with no initial compensation input determines λ_0, and the following time scales are determined based on the performance of the predictor (34). Note that, if the optimal step sequence size $\overset{*}{\mu}_k{}^i$ is used, $\gamma(t)$ should be chosen to have significant influence after a change in execution speed in order to account for modeling mismatches caused by the change of execution speed.

4. Experiments

Experiments demonstrating the use of the learning algorithm presented herein for quadrocopter flight are presented in this section. After introducing the experimental setup, we present two different flight tasks to which the algorithm was applied, discuss its performance, and demonstrate the influence of a selection of the tuning parameters introduced in Section 3.5.

4.1 Experimental Setup

The flight experiments were carried out in the Flying Machine Arena, an aerial vehicle research platform at ETH Zurich [32]. The vehicles used for the experiments are custom-built quadrocopters that are based on Ascending Technologies 'Hummingbird' vehicles [20]. The custom electronics [31] mounted on-board each vehicle provide inertial measurements and implement the body rate control law (10) based on the filtered measurements. The desired rotational rates $\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$ and the collective thrust command a (as defined in Equation (2)) are communicated to the vehicle from a desktop computer through a low-latency wireless communication channel at a rate of 50 Hz.

A commercial motion capture system [57] provides position and attitude information, which is filtered by a Luenberger observer. The filtered full state information is used on the desktop computer to implement the feedback controller (5)-(9), and the filtered position information is used in the learning algorithm.

The feedback control system used in these experiments introduces additional, albeit small, disturbances to the system in addition to the dynamic effects described in Section 2.1: Due to the wireless radio link between the quadrocopter and the desktop computer as well as the image processing required by the motion capture system, variable delays are introduced into the feedback loop. Furthermore, approximately 1 % to 5 % of the feedback control commands are lost in transmission over the wireless channel.

4.2 Learning Implementation

The learning algorithm was implemented on the desktop computer that also executes the feedback control law, and uses the position data provided by the Luenberger observer to compute the Fourier coefficients of the tracking error. In all experiments, the tracking error output $y(t)$ was the three-dimensional deviation of the position from the reference trajectory and the control input $u(t)$ was a set point correction to the feedback controller, as seen in Figure 2. Each iteration of the learning algorithm execution then consisted of the following steps:

1. Measure the tracking error for at least one period T. Note that, by measuring for more than one iteration, the variance of $N(k\Omega_0)$ can be reduced [52], therefore allowing the use of larger step sizes μ_k^i.
2. Apply the learning update law (19), using appropriate step sizes chosen as discussed in Section 3.5.

3. Wait for the system to converge under the new control inputs. Note that the instantaneous change of the correction input in Step 2 represents a non-periodic excitation of the system, making this step necessary.

4.3 Figure-Eight Maneuver

The first motion we consider is a periodic figure-eight maneuver flown at high speed in the horizontal plane around two obstacle points, as shown in Figure 3. This motion is used to demonstrate the basic working of the learning algorithm, as well as the influence of a number of learning parameters.

To execute the maneuver in closed loop, the feedback linearizing control law presented in Section 2.2 is used, and the transfer function $G(\omega)$ required for the iteration-domain feedback law (19) was derived from the approximate dynamics (11). The three degrees of freedom are decoupled in the approximate dynamics, and the transfer function matrix $G(\omega)$ therefore only has non-zero entries on its diagonal.

Maneuver Design The nominal maneuver is designed based on the quadrocopter dynamics (1)-(4). Assume, without loss of generality, that the first obstacle point lies at the origin of the inertial coordinate system O, and the second obstacle point is located at a distance L in the p_1-direction. The maneuver is composed of two half-circles about the obstacle points, and two splines connecting the half-circles. The maneuver is executed as fast as possible, with the speed being limited by the limitation of the collective thrust

$$a_{\min} \leq a \leq a_{\max} \tag{36}$$

and the limitation of the allowable body rate commands

$$|\omega_i| \leq \omega_{\max} \text{ for } i = \{x, y, z\} \; . \tag{37}$$

The first constraint, limiting the collective thrust, is given by the minimum and maximum rotational speed of the propellers. The second constraint, limiting the rotational rate of the quadrocopter, is given by the limited range of the gyroscopic inertial sensors. Because the quadrocopter has low rotational inertia and high achievable torques due to the outwards mounting of the propellers, the body rate control loop (10) has very high bandwidth. We therefore do not explicitly limit rotational accelerations, and assume that the rate constraint (37), with $\omega_{\max}$ suitably chosen, along with (36) suffices to ensure feasibility.

Based on the constraints (36) and (37), we will now design the two components of the figure-eight maneuver, i.e. the half-circles about the obstacle points and the splines connecting them:

Semi Circles: A circular trajectory, covering an angle of 180°, surrounds each obstacle point at a user-defined radius R_s. The thrust a and rotational rates ω_x, ω_y along this

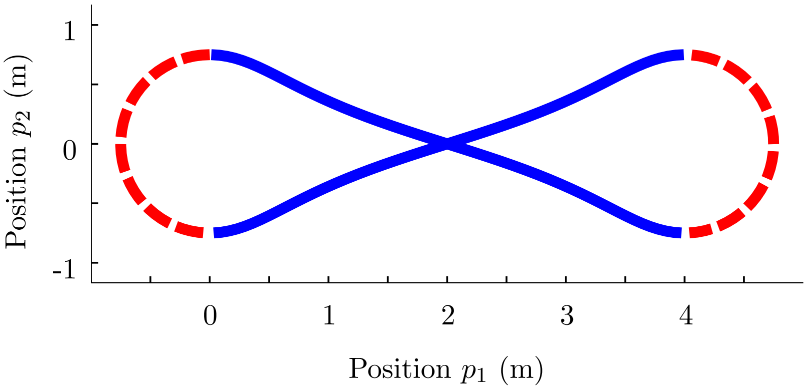

Figure 3. The reference trajectory of the figure-eight maneuver, highlighting the composition from two half-circles (dashed red) and two connecting splines (solid blue). The period of the maneuver is $T = 3.3\,\mathrm{s}$, and the maximum speed reached is $6\,\mathrm{m\,s^{-1}}$.

circular trajectory are then computed [49], and the time required for each semi circle is chosen to be the fastest time for which the control input constraints (36)-(37) are satisfied.

Connecting Splines: The two semi circle trajectories are connected through polynomial trajectories. The trajectory is continuous in position, velocity, attitude, and rotational rate of the quadrocopter if the polynomial trajectories match the position, velocity, acceleration, and jerk of the circular trajectories at either end [35]. In order to satisfy the four boundary constraints on both ends of the spline, we construct a seventh-order polynomial. The remaining degree of freedom in the spline design is the duration of maneuver. In order to achieve high speed, we iterate over the duration until the fastest maneuver that satisfies the control input bounds (36)-(37) is found, using the algorithm from [49] to compute the inputs and verify feasibility.

The figure-eight maneuver is then fully defined through the concatenation of the first semi circle, the first spline, the second semi circle, and the second spline. Due to the continuity and feasibility conditions imposed on each component of the trajectory, the resulting trajectory is feasible with respect to the constraints (36)-(37), and it is periodic.

An example of the figure-eight motion can be seen in Figure 3. In this specific example, the parameters were chosen to be $L = 4\,\mathrm{m}$, $R_s = 0.75\,\mathrm{m}$, $a_{\max} = 1.8\mathrm{g} = 17.65\,\mathrm{m\,s^{-2}}$, and $\omega_{\max} = 500\,{}^\circ\,\mathrm{s^{-1}}$. The resulting maneuver duration is $T = 3.3\,\mathrm{s}$, with an average speed of $4\,\mathrm{m\,s^{-1}}$ and a maximum speed of $6\,\mathrm{m\,s^{-1}}$. The duration of each half circle is $0.71\,\mathrm{s}$, and the duration of each connecting spline is $0.93\,\mathrm{s}$. This example was used as the reference trajectory to be learnt in the following results.

Learning at Fixed Maneuver Speed As a first test case, we show the learning performance when the maneuver is executed at the fixed speed of $\lambda = 0.7$, i.e. $70\,\%$ of the maneuver speed the motion was nominally designed for. The order of the correction input series was chosen to be $N = 10$, which showed to be a good compromise between

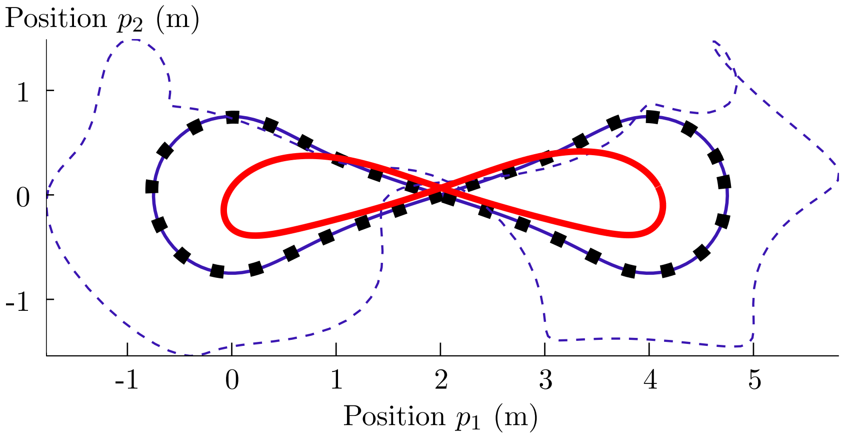

Figure 4. Top-down view of the flown trajectory before (thick solid red) and after (thin solid blue) learning. The dotted black line denotes the nominal trajectory, and the dashed blue line shows the set point $\hat{p}(t)$ provided to the feedback control law (as seen in Figure 2) after the learning is applied.

the ability to compensate for temporally localized tracking errors and robustness to high-frequency uncertainty. We chose the learning step size series to be

$$\mu_k^i = \min\left(1, \frac{3}{i+1}\right) \quad \text{for all k} \tag{38}$$

which provides fast initial convergence with the first three steps being of size 1, and good robustness to non-repetitive noise since the steps reduce in size as the iteration number increases.

At this speed, the maneuver could be safely executed with no initial correction input (i.e. $u^0(t) = 0$), and the learning algorithm converged. Figure 4 shows the trajectories of the vehicle and the set points in the horizontal plane, both at the start of the learning process and after convergence. It can be seen that the deviations from the nominal trajectory after the learning process are minimal, and that the learning algorithm significantly improved the tracking performance.

Figure 5 shows the evolution of the error coefficients over 22 iterations of the learning algorithm. It can be seen that the error coefficient magnitude quickly reduced from values in excess of 100 cm to values below 2 cm. The peak tracking error was reduced from approximately 200 cm to 5 cm. A significant outlier can be seen at the 18th iteration, likely caused by a non-repeatable disturbance to the vehicle such as a wireless communication failure. Because the step size $\mu_k^{18} = 3/19$ is relatively small, the large tracking error in this iteration was not strongly propagated to following iterations.

It should be noted that the feedback control law used in these experiments was not tuned to provide the best possible tracking performance without the adaptation; as discussed in Section 2.2, a straightforward improvement would be the inclusion of feedforward terms in the control law. What can, however, be seen from these results is that such

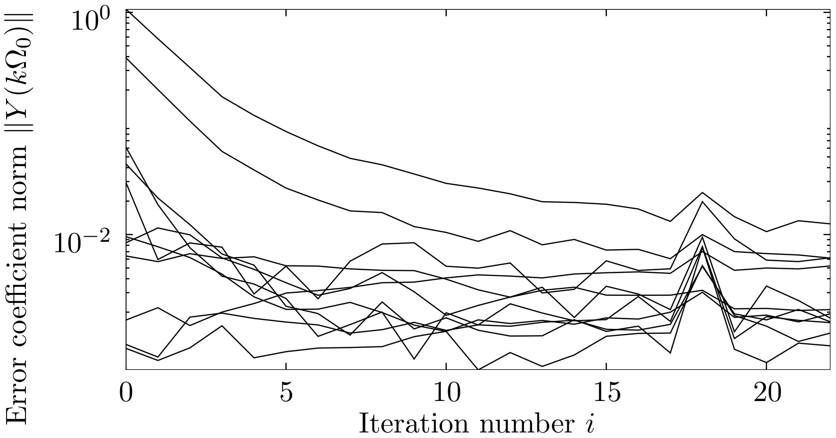

Figure 5. Evolution of the Fourier series coefficients during learning at a fixed maneuver speed of $\lambda = 0.7$. The lines at iteration zero are from top to bottom: $k = \{1, 2, 4, 5, 0, 6, 3, 7, 10, 8, 9\}$. Note the significant outlier for iteration $i = 18$, the influence of which on subsequent iterations was relatively small because $\mu_k^{18} = 3/19$ for all k is relatively small.

improvements to the feedback control law do not appear necessary because the adaptation law can largely eliminate repeatable tracking errors.

Influence of Fourier Series Order In a further experiment, we demonstrate the effect of varying the order of the compensation input Fourier series N, as given in Equation (15). The learning experiment was repeated at a constant speed of $\lambda = 0.8$, and the learning process was executed with Fourier series of the orders $N = \{0, 1, 2, 5, 8, 9, 10\}$. The resulting progression of the root mean square (RMS) position tracking error (computed over one motion period) over 25 iterations is shown in Figure 6. Note that the Fourier series of order $N = 0$ provided no significant improvement as it consists only of a constant set point shift. Increases in the Fourier series order consistently improved the learning performance initially. The orders $N = 5$ and $N = 8$ provided nearly identical performance, showing that the predominant tracking error components were sufficiently covered. The error then increased for $N = 9$, and the vehicle collided with the floor after five iterations for $N = 10$. This was caused by the insufficient accuracy of the approximate system dynamics (11) in the high-frequency range, and a possible remedy would be to reduce the step size for higher frequencies. Note that the order $N = 10$ could successfully be used for the experiments in Section 4.3 because the speed scaling factor λ was chosen lower, thereby mapping each component of the series to a lower frequency.

Figure 6 also shows the repeatability limit of our experimental system during the tests presented herein. In order to determine this limit, the maneuver was executed multiple times with an identical correction input, and the RMS deviation from the average flight trajectory was computed for each execution. With identical execution parameters as used for the other data in Figure 6, the average RMS repeatability error over fifteen executions was 13.6 mm, with a standard deviation of 2.2 mm. This can be taken to be a measure of the level of non-repeatable noise, as captured by $n(t)$, in the system. For

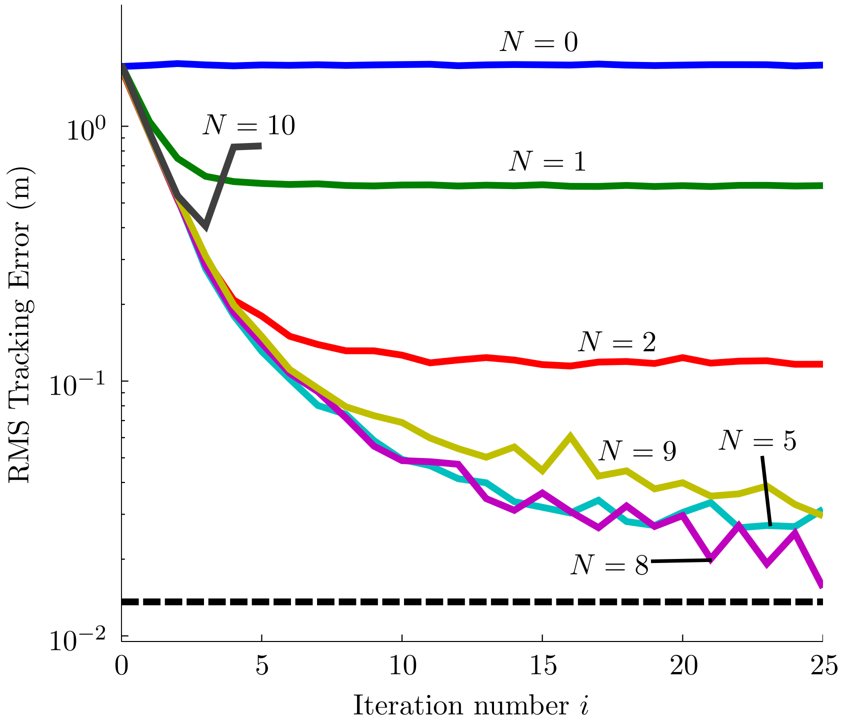

Figure 6. Learning performance (as measured by the RMS position tracking error) for the high-performance figure-eight maneuver executed at a speed of $\lambda = 0.8$, with varying order N of the correction Fourier series. The dashed black horizontal line represents the estimated repeatability limit of 1.4 cm. Note that the relatively large uncertainty at higher frequencies causes the learning performance to decrease for $N = 9$, and the vehicle collided with the floor during the sixth iteration for $N = 10$.

the flight trajectory learnt using a Fourier correction series of order $N = 8$, the average RMS tracking error over the last five iterations (i.e., $i = 21, 22, 23, 24, 25$) was 24.2 mm (with a standard deviation of 4.5 mm), indicating that the compensations by the learning algorithm corrected for almost all systematic errors. In other words, if an input signal $u(t)$ exists that would entirely eliminate all systematic tracking errors given the true dynamics of the quadrocopter, then it would allow the tracking error to be reduced by a further approximately 11 mm compared to the learning algorithm presented here.

Transfer to Increasing Speeds When initializing the learning algorithm with no correction signal for the full maneuver speed ($\lambda = 1$), the maneuver could not be learnt successfully. This is likely due to the maneuver being designed to use the full dynamic envelope of the quadrocopter, and the LTI system approximation not being sufficiently accurate when considering large initial deviations. Figure 7 demonstrates the use of the time scaling method (introduced in Section 3.4) to learn the maneuver at full speed. We

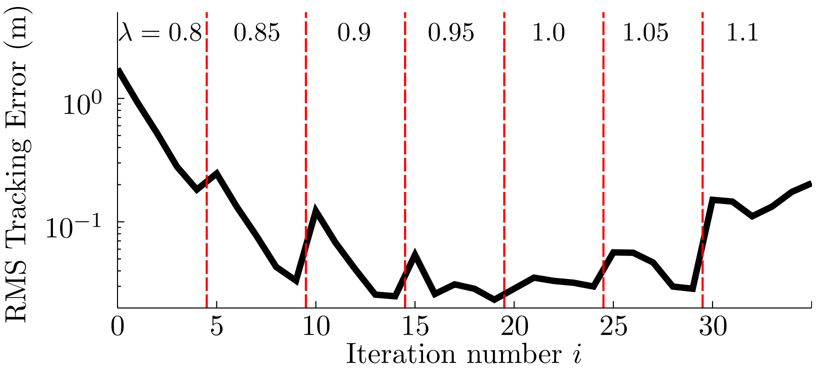

Figure 7. Logarithmic plot of the root mean square position tracking error during execution of the figure-eight maneuver with time scaling. The maneuver execution speed (shown at the top of the graph) is increased every five iterations. Note that $\lambda = 1.0$ denotes the nominal maneuver speed, and values $\lambda > 1.0$ are not nominally feasible.

initialized the speed factor to $\lambda_0 = 0.8$, and then applied the update law

$$\lambda_{\ell+1} = \lambda_\ell + 0.05 \tag{39}$$

after every five learning iterations of the maneuver. We used the same learning step size sequence (38) as before, but at iterations where time scaling occurs, we reset the iteration counter to $i = 0$ in order to compensate for the significant model uncertainty after a change of the time scale. The order of the correction Fourier series was chosen to be $N = 5$.

In Figure 7, the results of the experiment show that the RMS tracking error was initially reduced from 1.7 m to 0.18 m over the first four iterations, which were executed at $\lambda = 0.8$. The RMS error increased by values of 0.005 m to 0.09 m during the first four speed changes. Beginning with the 25th iteration, the execution speed was $\lambda > 1$, i.e. the maneuver is faster than was computed to be feasible according to Section 4.3. It can be seen that the tracking error increased as one would expect, and the learning algorithm ultimately failed to reduce the error at an execution speed of $\lambda = 1.1$ due to significant input saturations. These experimental results correspond well to the predicted feasibility.

4.4 Flying Inverted Pendulum

To further test the performance of the learning algorithm, we will now augment the quadrotor system to perform a more complex task. We attach an inverted pendulum to the quadrocopter, and design a new controller that takes the stabilization of the pendulum dynamics into account. The dynamic system controlled in this case thus consists of two coupled unstable systems (that is, the quadrocopter and the inverted pendulum), and can be considered to be an example of a controller design for quadrocopters executing more complex tasks (other examples of such tasks include slung load transportation [55] and cooperative control of multiple vehicles [29]). The flying inverted pendulum experiment was first presented in [21], and preliminary learning results have been presented in [23].

We begin by presenting the equations of motion of the pendulum on top of the quadrocopter and the controller design used to stabilize the overall system consisting of quadrocopter and pendulum. Following this, the learning is demonstrated for a sinusoidal trajectory.

System Design The inverted pendulum is modeled as a point mass with two degrees of freedom, chosen to be the horizontal position of the pendulum center of mass relative to its base in O (r in the direction of p_1, s in the direction of p_2). Using Lagrangian mechanics, the nonlinear equations of motion of the pendulum can be derived [21]. Assuming that the pendulum is attached to the center of mass of the quadrocopter (i.e. rotational motion of the vehicle does not affect the pendulum), the overall system dynamics linearized about the hover point are

$$
\begin{aligned}
\ddot{r} &= r\frac{\mathrm{g}}{L_p} - R_{13}\mathrm{g} && (40)\\
\ddot{s} &= s\frac{\mathrm{g}}{L_p} - R_{23}\mathrm{g} && (41)\\
\ddot{p}_1 &= R_{13}\mathrm{g} && (42)\\
\ddot{p}_2 &= R_{23}\mathrm{g} && (43)\\
\dot{R}_{13} &= \omega_y && (44)\\
\dot{R}_{23} &= -\omega_x && (45)\\
\ddot{p}_3 &= a - \mathrm{g} && (46)
\end{aligned}
$$

where L_p is the length of the pendulum from its base to its center of mass and we omitted the rotational dynamics (1) under the assumption that they remain controlled as before using the feedback law (10) and that they are significantly faster than the remaining dynamics.

The two horizontal degrees of freedom represent fifth-order systems, with the vehicle forming a triple integrator from the body rate to its position. The vertical motion is represented by a double integrator from thrust to position.

The feedback controller for the dynamics (40)-(46) is designed using an infinite-horizon linear quadratic regulator (LQR; see e.g. [30]) design. The design penalizes only the vehicle position p_1, p_2, p_3 and the control effort ω_x, ω_y, a, and there is no penalty on the pendulum state. The ratio of the penalties on position and control effort controls the speed at which the position set point is tracked. Values for this ratio are tuned manually until the system shows fast performance without saturating the control inputs. Note that, for this specific controller, no effort of feedback linearization (as presented in Section 2.2 for the vehicle only) is made.

The approximate system dynamics (14) are derived by computing the closed-loop

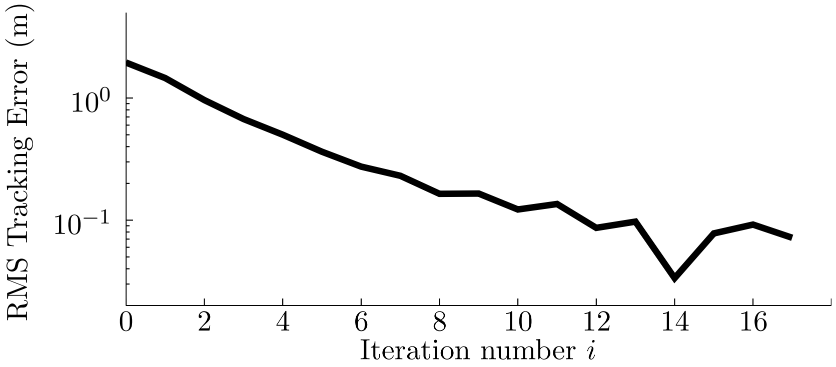

Figure 8. Logarithmic plot of the root mean square position tracking error during the inverted pendulum tracking task, as seen in Figure 10. The learning step size for this experiment was chosen to be constant with $\bar{\mu}_k = 0.6$ for all k.

transfer function of the linear dynamics under LQR feedback control.

We define the nominal trajectory to be two sinusoids in the two horizontal degrees of freedom:

$$p_1^*(t) = A_1 \sin(2\pi t/T) \tag{47}$$

$$p_2^*(t) = A_2 \sin(4\pi t/T)\,. \tag{48}$$

For the experiments presented herein, the parameters were chosen to be $T = 5\,\mathrm{s}$, $A_1 = 1.5\,\mathrm{m}$, and $A_2 = 1\,\mathrm{m}$. The resulting average vehicle speed is $2.1\,\mathrm{m\,s^{-1}}$, with a maximum of $3.1\,\mathrm{m\,s^{-1}}$, and was chosen slower than in the previous experiments in order to account for the added complexity of balancing the inverted pendulum.

For this experiment, the learning step size was chosen to be constant and identical for all frequencies, i.e. $\mu_k^i = 0.6$ for all i and all k. The order of the correction input Fourier series was chosen to be $N = 4$.

Results The vehicle position RMS tracking error during the learning process is shown in Figure 8. It can be seen that the initial execution caused very high tracking errors (RMS error of 1.95 m), and that the tracking error was continuously reduced, with tracking errors below 0.1 m achieved for all iterations $i > 11$. Figure 9 shows the flight path before and after application of the learning algorithm, and a series of snapshots of the experiment can be seen in Figure 10 with the reference trajectory superimposed on the pictures.

5. Advantages and Limitations

The experimental results in the previous section demonstrated the ability of the learning algorithm to significantly improve the tracking performance for periodic maneuvers. A

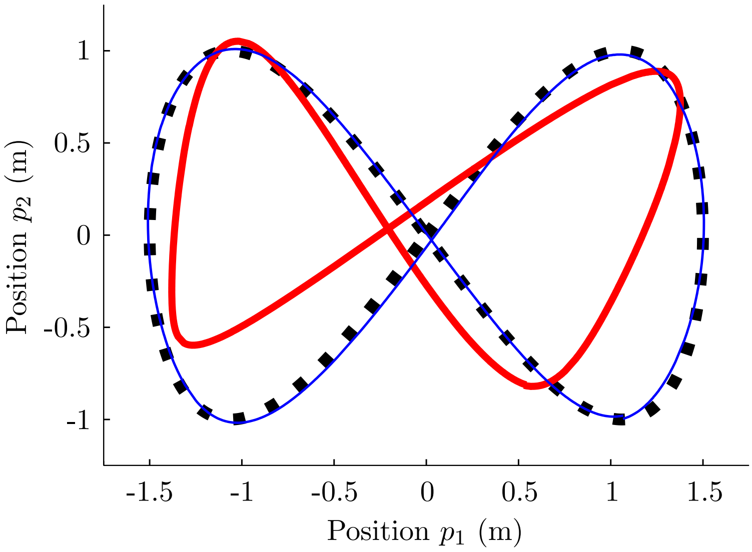

Figure 9. Top view of the flown trajectories during the inverted pendulum learning experiment. The dotted black lines shows the reference trajectory $p^*(t)$ as described in Section 4.4. The thick red line shows the flown trajectory before learning, and the thin blue line shows the trajectory after applying the learning algorithm.

key enabler for this is that the learning is performed on the closed-loop system of the quadrocopter under feedback control, which makes repeated maneuver executions highly repeatable, and therefore well-suited for the non-causal correction of tracking errors.

The parametrization of the correction input $u(t)$ as a truncated Fourier series allows the use of the order of the series to define a trade-off between 1) the ability to compensate for highly localized tracking errors, and 2) computational complexity and memory requirements. Relatively low Fourier series orders often suffice to significantly improve the tracking performance, as seen in our experiments. The limitation to relatively low Fourier series orders also provides a convenient way to suppress high-frequency jitter in the learnt compensation, an effect that can be frequently observed in iterative learning control approaches [51]. Furthermore, inaccuracies of the dynamic model at high frequencies can be circumvented by limiting the learning to lower frequencies, or can be modeled through the characteristics of the noise models used to capture changing dynamics and non-repeatable effects. Through the appropriate choice of these noise characteristics, the learning algorithm can be tuned to quickly compensate for disturbances at frequencies where an accurate model of the system is available, and to perform more cautious adaptation at frequencies where significant model uncertainty is present.

It can also be seen from our experiments that highly simplified models of the closed-loop dynamics, though only capturing a relatively rough approximation of the true behaviour, suffices to guide the learning process both for the stand-alone quadrocopter as well as when more complex tasks are performed. Similar results have been demonstrated for other learning algorithms, e.g. reinforcement learning [28]. The simplified model,

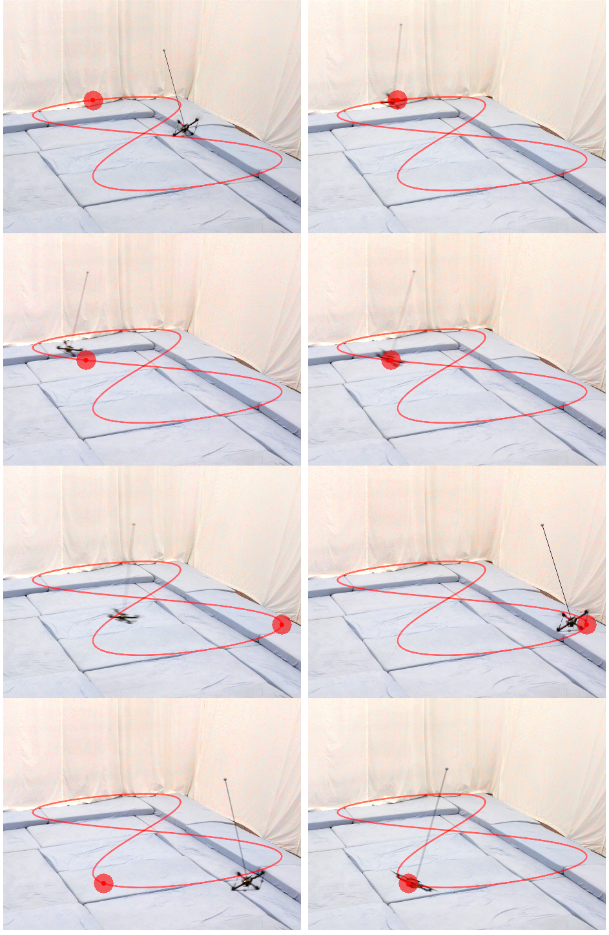

Figure 10. Snapshots of the inverted pendulum trajectory tracking experiment. The red line denotes the nominal trajectory $p^*(t)$, and the point on it represents the instantaneous nominal position at the time of the snapshot. Performance before learning ($i = 0$) is seen on the left, and performance after ten learning iterations ($i = 10$) is seen on the right. The snapshots are taken every 1 s, and are ordered from top to bottom. Notice that, before learning, the vehicle lags behind the nominal trajectory significantly.

and therewith the uncomplicated derivation of such a model, make the application of the algorithm to changing conditions (such as specific tasks, differing controllers, or different vehicles) a straightforward undertaking.

The algorithm presented particularly lends itself to applications where computational power and memory are limited. During the execution of the maneuver, the algorithm only requires the estimation of the coefficients of the tracking error Fourier series up to order N, and the control input update can be reduced to a matrix multiplication according to Equation (19) by pre-computing the step size sequence μ_k^i and the inverse transfer function $G^+ (k\Omega_0)$ for the considered frequencies $k = 0, 1, \ldots, N$.

Due to the simplified dynamic model, the serial architecture, and the frequency domain approach in the learning algorithm, it is not trivial to incorporate additional constraints. Examples of potentially useful constraints are the explicit consideration of input saturations, or the penalization of control effort. Their inclusion can improve the learning performance because they help capture the underlying dynamics more accurately, or can help to shape the outcome of the learning process.

Furthermore, maneuvers exhibiting very large initial tracking errors highlight the limitations of using a simplified model in the learning process. The system dynamics may not be captured with sufficient accuracy for large errors, causing the learning algorithm to fail. The time-scaling method presented herein can offer a viable solution to this problem when it is possible to initially reduce the execution speed of the maneuver: The frequency domain approach to iterative learning provides a convenient way to transfer learnt correction inputs between different execution speeds of the same maneuver. This allows initial learning to occur at reduced speeds, thus providing a safe way to learn high-speed maneuvers where a poor initial guess of the correction input can lead to a crash or to non-convergence.

6. Conclusion

This paper presents the use of a frequency-domain iterative learning scheme for periodic quadrocopter flight. The iteration-domain feedback law leverages an underlying feedback control law that stabilizes the vehicle and makes its motion highly repeatable. The nominally linear time-invariant closed-loop dynamics of the quadrocopter feedback system are used to determine correction values from observed errors in a straightforward manner. By parameterizing the non-causal tracking error compensation as Fourier series, the algorithm is computationally lightweight and easy to adapt. Uncertainties, such as measurement noise, inaccuracies of the approximate transfer function, or model uncertainty can be accounted for in the learning step size, and the optimal step size for each frequency component can be computed from the statistical properties thereof. The approach also allows the learning of high-performance maneuvers by executing them at a reduced speed initially and then transferring learnt corrections to higher speeds, where a disturbance prediction scheme is used to initialize the learning at higher speeds.

The approach was experimentally verified for a quadrocopter executing a high-performance motion, and for it executing the complex task of balancing an inverted pendulum while tracking a trajectory. The experimental results also highlighted the advantages of learning at reduced speeds, with a the high-performance motion only being learnt successfully at full speed when using learnt parameters from lower speeds to initialize the learning process.

References

[1] Airrobot GmbH & Co KG. AirRobot Product Information (Available online at www.airrobot-us.com/images/Airrobot_info.pdf), 2007.

[2] K Alexis, G Nikolakopoulos, and A Tzes. Constrained Optimal Attitude Control of a Quadrotor Helicopter subject to Wind-Gusts: Experimental Studies. In *American Control Conference*, 2010.

[3] Brian D O Anderson and John B Moore. *Optimal Filtering*. Dover Publications, 2005.

[4] Gianluca Antonelli, Filippo Arrichiello, Stefano Chiaverini, and Paolo Robuffo Giordano. Adaptive Trajectory Tracking for Quadrotor MAVs in Presence of Parameter Uncertainties and External Disturbances. In *International Conference on Advanced Intelligent Mechatronics*, 2013.

[5] S Arimoto, S Kawamura, and F Miyazaki. Bettering Operation of Dynamic Systems by Learning: A New Control Theory for Servomechanism or Mechatronic Systems. In *Conference on Decision and Control*, 1984.

[6] Moses Bangura and Robert Mahony. Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor. In *Australasian Conference on Robotics and Automation*, 2012.

[7] T.K. Barlas and G.a.M. van Kuik. Review of state of the art in smart rotor control research for wind turbines. *Progress in Aerospace Sciences*, 46(1):1–27, January 2010.

[8] Sergio Ronaldo Barros dos Santos, Sidney N Givigi, and Cairo Lucio Nascimento. Autonomous Construction of Structures in a Dynamic Environment using Reinforcement Learning. In *Systems Conference*, 2013.

[9] Kira Barton, Sandipan Mishra, and Enric Xargay. Robust Iterative Learning Control: L1 Adaptive Feedback Control in an ILC Framework. In *American Control Conference*, 2011.

[10] Dennis S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[11] Pierre-jean Bristeau, Philippe Martin, Erwan Salaün, and Nicolas Petit. The Role of Propeller Aerodynamics in the Model of a Quadrotor UAV. In *European Control Conference*, 2009.

[12] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A Survey of Iterative Learning Control. *Control Systems Magazine*, 26(3):96–114, 2006.

[13] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 3rd edition, 1998.

[14] Insik Chin, S. Joe Qin, Kwang S. Lee, and Moonki Cho. A Two-Stage Iterative Learning Control Technique Combined with Real-Time Feedback for Independent Disturbance Rejection. *Automatica*, 40(11):1913–1922, 2004.

[15] Li Cuiyan, Zhang Dongchun, and Zhuang Xianyi. A Survey of Repetitive Control. In *International Conference on Intelligent Robots and Systems*, 2004.

[16] Travis Dierks and Sarangapani Jagannathan. Output Feedback Control of a Quadrotor UAV Using Neural Networks. *IEEE Transactions on Neural Networks*, 21(1):50–66, 2010.

[17] J Dunfied, M Tarbouchi, and G Labonte. Neural Network Based Control of a Four Rotor Helicopter. In *International Conference on Industrial Technology*, 2004.

[18] Walter Gautschi. *Numerical Analysis*. Birkhäuser Boston, 2012.

[19] Alexandra Suzanne Gibb. *Droning the Story*. Master thesis, Simon Fraser University, 2011.

[20] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[21] Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. In *International Conference on Robotics and Automation*, 2011.

[22] Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Feed-Forward Learning Scheme for High Performance Periodic Quadrocopter Maneuvers. In *International Conference on Intelligent Robots and Systems*, 2013.

[23] Markus Hehn and Raffaello D'Andrea. An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories. In *European Control Conference*, 2013.

[24] Ivo Houtzager, Jan-Willem van Wingerden, and Michel Verhaegen. Rejection of Periodic Wind Disturbances on a Smart Rotor Test Section Using Lifted Repetitive Control. *IEEE Transactions on Control Systems Technology*, 21(2):347–359, 2013.

[25] Hwei P. Hsu. *Schaum's Outline of Signals and Systems*. McGraw-Hill, 1995.

[26] Peter C. Hughes. *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

[27] T Inoue, S Iwai, and M Nakano. High Accuracy Control of a Proton Synchrotron Magnet Power Supply. In *IFAC World Congress*, 1981.

[28] J Zico Kolter and Andrew Y Ng. Policy Search via the Signed Derivative. In *Robotics: Science and Systems*, 2009.

[29] Alex Kushleyev, Daniel Mellinger, and Vijay Kumar. Towards A Swarm of Agile Micro Quadrotors. In *Robotics: Science and Systems*, 2012.

[30] Huibert Kwakernaak and Raphael Sivan. *Linear Optimal Control Systems*. John Wiley & Sons, 1972.

[31] Sergei Lupashin and Raffaello D'Andrea. Adaptive Fast Open-Loop Maneuvers for Quadrocopters. *Autonomous Robots*, 33(1-2):89–102, 2012.

[32] Sergei Lupashin, Angela P. Schoellig, Markus Hehn, and Raffaello D'Andrea. The Flying Machine Arena as of 2010. In *International Conference on Robotics and Automation*, 2011.

[33] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, 2012.

[34] D. Mellinger, N. Michael, and V. Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[35] Daniel Mellinger and Vijay Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *International Conference on Robotics and Automation*, 2011.

[36] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *International Symposium on Experimental Robotics*, 2010.

[37] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[38] Fabian L. Mueller, Angela P. Schoellig, and Raffaello D'Andrea. Iterative Learning of Feed-Forward Corrections for High-Performance Tracking. In *International Conference on Intelligent Robots and Systems*, 2012.

[39] Mark Muller, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Ball Juggling. In *International Conference on Intelligent Robots and Systems*, 2011.

[40] C. Nicol, C.J.B. Macnab, and A. Ramirez-Serrano. Robust Adaptive Control of a Quadrotor Helicopter. *Mechatronics*, 21(6):927–938, 2011.

[41] Alan V Oppenheim and Ronald W Schafer. *Discrete-Time Signal Processing.* Pearson Prentice Hall, third edition, 2010.

[42] Ivana Palunko and Rafael Fierro. Adaptive Control of a Quadrotor with Dynamic Changes in the Center of Gravity. In *IFAC World Congress*, 2011.

[43] Pong-in Pipatpaibul and P R Ouyang. Application of Online Iterative Learning Tracking Control for Quadrotor UAVs. *ISRN Robotics*, 2013:Article ID 476153, 2013.

[44] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and Control of a Quad-Rotor Robot. In *Australasian Conference on Robotics and Automation*, 2006.

[45] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, July 2010.

[46] Caitlin Powers, Daniel Mellinger, Aleksandr Kushleyev, and Bruce Kothmann. Influence of Aerodynamics and Proximity Effects in Quadrotor Flight. In *International Symposium on Experimental Robotics*, 2012.

[47] Oliver Purwin and Raffaello D'Andrea. Performing and Extending Aggressive Maneuvers Using Iterative Learning Control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[48] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative Quadrocopter Ball Throwing and Catching. In *International Conference on Intelligent Robots and Systems*, 2012.

[49] Angela Schoellig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasiblity of Motion Primitives for Choreographed Quadrocopter Flight. In *American Control Conference*, 2011.

[50] Angela Schoellig, Clemens Wiltsche, and Raffaello D'Andrea. Feed-Forward Parameter Identification for Precise Periodic Quadrocopter Motions. In *American Control Conference*, 2012.

[51] Angela P Schoellig, Fabian L Mueller, and Raffaello D'Andrea. Optimization-Based Iterative Learning for Precise Quadrocopter Trajectory Tracking. *Autonomous Robots*, 33(1-2):103–127, 2012.

[52] J Schoukens and J Renneboog. Modeling the Noise Influence on the Fourier Coefficients After a Discrete Fourier Transform. *IEEE Transactions on Instrumentation and Measurement*, IM-35(3):278–286, 1986.

[53] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Autonomous Mobile Robots*. The MIT Press, second edition, 2011.

[54] A K Sinha, S Atyeo, A Schauenburg, A Rathore, B Quinn, and T Christoffersen. Investigation of the Application of UAV Technology in Power Line Inspection. In *International UAV Systems Conference*, 2006.

[55] Koushil Sreenath, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control of a Quadrotor with a Cable-Suspended Load A Differentially-Flat Hybrid System. In *International Conference on Robotics and Automation*, 2013.

[56] Stefania Tonetti, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Distributed Control of Antenna Array with Formation of UAVs. In *IFAC World Congress*, 2011.

[57] Tara Volgoi, Hayley Roberts, Lindsay Gerber, and Nathan Aswege. Tracking The Future. In *The Standard*, pages 6–7. Vicon Motion Systems Ltd, first edition, 2011.

[58] Youqing Wang and Francis J. Doyle. Stability Analysis for Set-Point-Related Indirect Iterative Learning Control. In *Conference on Decision and Control*, 2009.

[59] Youqing Wang, Furong Gao, and Francis J. Doyle. Survey on Iterative Learning Control, Repetitive Control, and Run-to-Run Control. *Journal of Process Control*, 19(10):1589–1600, 2009.

[60] Steven L. Waslander, Gabriel M. Hoffmann, Jung Soon Jang, and Claire J. Tomlin. Multi-Agent Quadrotor Testbed Control Design: Integral Sliding Mode vs. Reinforcement Learning. In *International Conference on Intelligent Robots and Systems*, 2005.

Paper V

An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories

Markus Hehn and Raffaello D'Andrea

Abstract

Quadrocopters allow the execution of high-performance maneuvers under feedback control. However, repeated execution typically leads to a large part of the tracking errors being repeated. This paper evaluates an iterative learning scheme for an experiment where a quadrocopter flies in a circle while balancing an inverted pendulum. The scheme permits the non-causal compensation of periodic errors when executing the circular motion repeatedly, and is based on a Fourier series decomposition of the repeated tracking error and compensation input. The convergence of the learning scheme is shown for the linearized system dynamics. Experiments validate the approach and demonstrate its ability to significantly improve tracking performance.

Published in *Proceedings of the 2013 European Control Conference (ECC)*, 2013.

©2013 EUCA. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the European Control Association (EUCA).

1. Introduction

The capability of quadrocopters to perform highly dynamic, complex, and precise motions has been demonstrated repeatedly in recent years (see, for example, [12, 14, 15, 18]). Such motions are commonly executed by using a first-principles model of the vehicle dynamics to determine nominal control inputs, and a feedback control law to ensure tracking of the nominal trajectory.

In order to account for model mismatches, a number of learning schemes have been developed. Examples include those based on sliding mode control and reinforcement learning techniques [20], neural networks [5], and adaptive control [16].

When performing the same motion repeatedly, a further opportunity to improve tracking performance may arise because many of the disturbances that degrade tracking performance will be similar each time the vehicle performs the motion. It is thus possible to non-causally correct for these repeatable disturbances: By using data from previous executions to characterize them, model-based correction inputs can be computed before executing the motion again. Ideally, these correction inputs are able to fully compensate for the repeatable disturbances, such that the feedback controller is only required to compensate for non-repeatable disturbances.

Such iteration-based learning approaches have been successfully demonstrated for multi-rotor vehicles performing high-performance maneuvers. Broadly speaking, the learning approaches may be separated into two groups:

The first group is characterized by its ability to learn motions that are parameterized. The motion is thus described by a (finite) set of design parameters, chosen by the user. After the execution of the motion, these parameters are adapted to compensate for disturbances. The direction and magnitude of the correction may be model-based, or based on the user's intuition. A discussion on the importance of choosing 'good' design parameters may be found in [10], where a learning algorithm for this kind of parameterized motions is demonstrated for multiple flips and fast translations with quadrocopters. A further demonstration of this class of learning algorithms is provided in [13]. The ability to shape the tracking performance strongly depends on the number of parameters that are optimized; in the above examples, the objective is to minimize the error at specific time instants ('key frames'), and a relatively small number of parameters is sufficient to do so. This makes the methods computationally lightweight.

The second group of learning approaches considers more generic motions that need not be specified by parameters. The system dynamics are considered in discrete time, and the correction consists of correction values (typically control inputs or set points) for each discrete time step. After execution of the motion, a numerical optimization over the correction values is performed in order to minimize a metric related to the tracking error. In this optimization, a model of the system dynamics provides the mapping from corrections to the tracking error. This approach is commonly known as a form of iterative learning control [3], and its application to high performance quadrocopter flight has been demonstrated [17, 19].

The delimitation between the two groups is not strict. Indeed, the second group of learning approaches could be seen as using a very large number of values to parameterize the correction.

This paper evaluates a technique for non-causally compensating repeated, periodic tracking errors. Specifically, we consider a motion where the linearized dynamics around the nominal motion are time-invariant under an appropriate coordinate transformation. Similar to the second group of learning algorithms, we do not assume a parameterized motion. However, we reduce the dimensionality of the corrections that we intend to learn by assuming that they are periodic. This allows us to parameterize the corrections as the coefficients of a truncated Fourier series. The order of the Fourier series provides a means to trade off computational complexity and the ability to compensate for temporally local or high-frequency disturbances.

The specific motion considered in this paper consists of a quadrocopter balancing an inverted pendulum and tracking a circular trajectory [7]. This problem is an interesting platform for learning algorithms for multiple reasons: Due to the highly dynamic nature of the motion and the agility required to balance the pendulum, the full range of the vehicle dynamics is used. Due to relatively high flight speeds and fast vehicle attitude changes, unmodeled aerodynamic effects (see e.g. [9]) significantly influence the dynamics of the vehicle. Furthermore, the control system employs a time-varying coordinate system transformation, the errors in which potentially introduce further periodic errors in the feedback control loop.

The remainder of this paper is structured as follows: We explain our motivation for developing the learning algorithm in Chapter 2 by introducing the flying inverted pendulum experiment and recapitulating experimental results that highlight its performance without learning. We then describe the iterative learning strategy in Chapter 3, and show convergence of repeatable errors. Chapter 4 shows experimental results from the application of the algorithm to the experiment. Finally, we summarize and discuss potential research directions in Chapter 5.

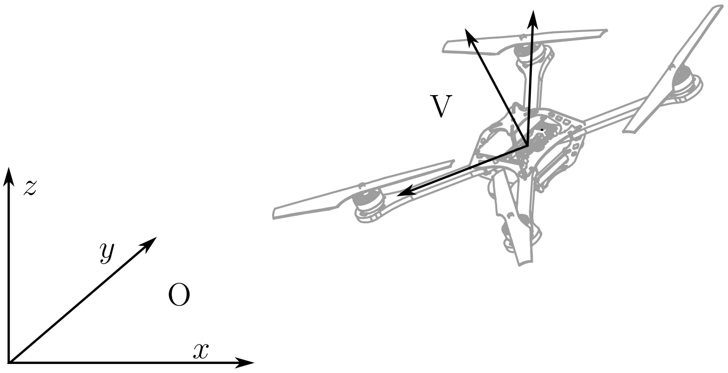

Figure 1. The inertial coordinate system O and the vehicle coordinate system V, used to describe the dynamics of the system.

2. The Flying Inverted Pendulum Experiment

This chapter introduces the flying inverted pendulum [7], an experiment that demonstrates the performance and agility of quadrocopters. This experiment is the motivation behind the algorithm presented in Chapter 3, and is the basis for the experimental results in Chapter 4. The objective of this experiment is to balance an inverted pendulum on a quadrocopter while the vehicle flies horizontal circles. The derivations from [7] are reproduced here in abbreviated form for the purpose of completeness; the reader is referred to the previously published paper for a more thorough discussion.

2.1 Dynamics

The quadrocopter is modeled as a rigid body with six degrees of freedom: Its position (x, y, z) in the inertial coordinate system O, and its attitude, represented by the rotation between the inertial coordinate system O and the body-fixed coordinate system V, as shown in Figure 1. The rotation is parameterized by three Euler angles, representing rotations about the z-axis (α), the y-axis (β) and the x-axis(γ), executed in this order:

$$ {}^{\mathrm{O}}_{\mathrm{V}}R(\alpha, \beta, \gamma) = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma)\ . \tag{1} $$

The control inputs are the rotational rates of the vehicle about the three body axes $(\omega_x, \omega_y, \omega_z)$ and the collective, mass-normalized thrust applied by the vehicle along its body z-axis, (a; in units of acceleration). It follows that the differential equations governing the vehicle motion are

$$ \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}^{\mathrm{O}}_{\mathrm{V}}R(\alpha, \beta, \gamma) \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} \tag{2} $$

$$ \begin{bmatrix} \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\gamma & -\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\gamma & 0 \\ -\sin\beta & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3} $$

where g denotes gravitational acceleration.

The inverted pendulum is modeled as a point mass with two degrees of freedom, chosen to be the horizontal position of the pendulum center of mass relative to its base in O (r along the x-axis, s along the y-axis). Using Lagrangian mechanics, the nonlinear equations of motion of the pendulum can be derived. In the interest of compactness, we state only that the pendulum acceleration depends on its position and velocity, and the vehicle acceleration:

$$\begin{bmatrix} \ddot{r} \\ \ddot{s} \end{bmatrix} = \mathrm{h}\left(r, s, \dot{r}, \dot{s}, \ddot{x}, \ddot{y}, \ddot{z}\right) . \tag{4}$$

2.2 Coordinate Transformation for Circular Trajectories

The objective of the experiment is to fly circular trajectories while balancing the pendulum. We introduced a transformation into rotating coordinate systems for the vehicle position (C) and attitude (W) in [7]. It is then possible to transform the equations of motion such that, for circular flight, the nominal states and the linearized dynamics about them can be described in a time-invariant manner:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} =: \begin{bmatrix} \cos \Omega t & -\sin \Omega t & 0 \\ \sin \Omega t & \cos \Omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{5}$$

$${}^{\mathrm{O}}_{\mathrm{V}}R(\alpha, \beta, \gamma) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} =: {}^{\mathrm{O}}_{\mathrm{W}}R(\Omega t, \mu, \nu) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} r \\ s \end{bmatrix} =: \begin{bmatrix} \cos \Omega t & -\sin \Omega t \\ \sin \Omega t & \cos \Omega t \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} . \tag{7}$$

A horizontal circle of radius R, flown at a constant rate Ω and at constant height, is described by $u = R$, $v = 0$, and $\dot{w} = 0$. The quadrocopter and pendulum equations of motion (2)-(4) can be rewritten in the rotating coordinate system using the above transformations. It was shown that a nominal circular trajectory for pendulum and quadrocopter is then given by the constant nominal state

$$\bar{\mu} = \arctan\left(-\frac{\Omega^2 R}{\mathrm{g}}\right) \tag{8}$$

$$\bar{\nu} = 0 \tag{9}$$

$$\bar{a} = \sqrt{\mathrm{g}^2 + \left(\Omega^2 R\right)^2} \tag{10}$$

$$\bar{q} = 0 \tag{11}$$

$$\Omega^2 (R + \bar{p}) + \frac{\mathrm{g}\bar{p}}{\sqrt{L^2 - \bar{p}^2}} = 0 \tag{12}$$

where L represents the length of the pendulum from its base to its center of mass. The vehicle yaw angle, α, is not defined by the motion, and may be chosen separately. In our experiments, we chose a constant yaw angle $\alpha = 0$.

2.3 Feedback Control Design

About this constant nominal state, the dynamics of the quadrocopter-pendulum system in the rotating C-W coordinate system were approximated by a first-order Taylor expansion. We denote the system state under the coordinate transformation (5)-(7) by

$$\mathrm{x} := (u, v, w, \dot{u}, \dot{v}, \dot{w}, \alpha, \mu, \nu, p, q, \dot{p}, \dot{q}) \tag{13}$$

and use the transformed control input $\mathrm{v} := (a, \dot{\mu}, \dot{\nu}, \dot{\alpha})$. The true rotational rate control inputs $(\omega_x, \omega_y, \omega_z)$ can be recovered for known values of v and x through Equations (6) and (3). The linear dynamics resulting from a Taylor expansion about the nominal trajectory are then time-invariant:

$$\dot{\tilde{\mathrm{x}}} = \frac{\partial \mathrm{f}\left(\bar{\mathrm{x}}, \bar{\mathrm{v}}\right)}{\partial \mathrm{x}} \tilde{\mathrm{x}} + \frac{\partial \mathrm{f}\left(\bar{\mathrm{x}}, \bar{\mathrm{v}}\right)}{\partial \mathrm{v}} \tilde{\mathrm{v}} \tag{14}$$

where a tilde denotes small deviations from the nominal trajectory, and $\mathrm{f}\left(\bar{\mathrm{x}}, \bar{\mathrm{v}}\right)$ denotes the system dynamics consisting of Equations (2)-(4) under the coordinate transformation (5)-(7). An infinite-horizon linear quadratic regulator [2] was designed to stabilize the system around the nominal trajectory. We denote the resulting time-invariant state feedback law by $\tilde{\mathrm{v}} = K\tilde{\mathrm{x}}$.

2.4 Flight Performance

Results from the previous experiment [7] demonstrate the ability of the presented control system design to reliably stabilize the vehicle-pendulum system during circular flight. However, significant trajectory tracking errors occur, as can be seen for an exemplary experiment in Figure 2. Note that the errors contain two clearly distinguishable components: A mean error, and an error that oscillates at the rate at which the circle is flown (Ω). A possible explanation for these oscillating errors is the rotating coordinate system transformation (5)-(7), which would map a constant error in the inertial coordinate system to such an oscillating error in the rotating coordinate system.

Figure 2 also shows that the tracking error largely repeats for every round of the circle that is flown: The experiment may be considered to be a periodic motion with relatively large repeatable disturbances deteriorating the tracking performance of the feedback control law. As discussed in Chapter 1, the repeatability of the disturbances offers the opportunity to non-causally compensate for them. We develop a learning algorithm for this purpose in the following chapter, and will revisit the above experiment to present experimental results thereafter.

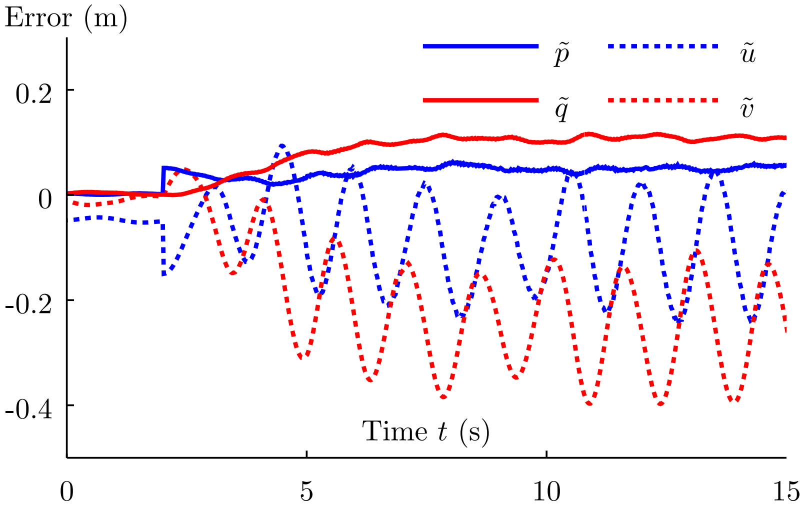

Figure 2. Errors in the rotating coordinate system (from [7]): Pendulum position error $(\tilde{p}, \tilde{q})$ and quadrocopter position error $(\tilde{u}, \tilde{v})$. At $t = 2\,\mathrm{s}$, the controller is switched from a constant nominal position to a circular trajectory with $R = 0.1\,\mathrm{m}$. Large repeating errors can be seen.

3. Learning Algorithm

The approach presented in this paper is conceptually similar to [10], which presents an adaptation strategy to correct for state errors at discrete points in time of parameterized motion primitives. However, we consider periodic errors (instead of errors at specific points in time), and do not require parameterization of the maneuver. We propose a correction strategy, and show that, under the assumption of linear time-invariant system dynamics, this strategy indeed fully corrects for such recurring errors.

3.1 Core Concept

The core idea of the adaptation law presented herein is to measure the tracking error over the period of the maneuver, and approximate it as a truncated Fourier series. In order to compensate for recurring errors, a correction input – also consisting of a truncated Fourier series – is applied to the system. For a known tracking error, the coefficients of the correction input Fourier series are computed by inverting the linear time-invariant dynamics of the system about the nominal trajectory. In order to increase the robustness against non-repeatable disturbances, the adaptation law uses a step size, permitting a trade-off between speed of convergence and noise rejection.

3.2 Adaptation Control Input

In addition to the state feedback control input, we apply an adaptation control input $\hat{\mathrm{v}}$ to the system. Using A and B to denote the derivatives with respect to the state and input, respectively, we can then rewrite Equation (14) as

$$\begin{aligned}\dot{\tilde{\mathrm{x}}} &= A\tilde{\mathrm{x}} + B\left(K\tilde{\mathrm{x}} + \hat{\mathrm{v}}\right) && (15)\\ &= \underbrace{(A+BK)}_{=:\bar{A}}\tilde{\mathrm{x}} + B\hat{\mathrm{v}} && (16)\end{aligned}$$

where $\bar{A}$ represents the closed-loop linearized dynamics of the system.

3.3 Propagation of Fourier Series Inputs Through the System

Now we will use the adaptation control input $\hat{\mathrm{v}}$ to non-causally compensate for repeated disturbances. We parameterize this compensation by a Fourier series of order N and fundamental frequency ω:

$$\hat{\mathrm{v}} = r_0 + \sum_{k=1}^{N} r_k \cos\left(k\omega t\right) + \sum_{k=1}^{N} s_k \sin\left(k\omega t\right)\ . \tag{17}$$

It is straightforward to show that, to first order, the perturbation state $\tilde{\mathrm{x}}$ in reaction to this adaptation control input will also be a Fourier series of order N [4]:

$$\tilde{\mathrm{x}} = a_0 + \sum_{k=1}^{N} a_k \cos\left(k\omega t\right) + \sum_{k=1}^{N} b_k \sin\left(k\omega t\right) \tag{18}$$

and that the coefficients relate to the coefficients of the input by

$$\begin{aligned} 0 &= \bar{A}a_0 + Br_0 && (19)\\ -k\omega a_k &= \bar{A}b_k + Bs_k && (20)\\ k\omega b_k &= \bar{A}a_k + Br_k && (21)\end{aligned}$$

for $k = 1 \ldots N$.

3.4 Iteration-Domain Feedback Law

Consider a tracking error output that is a linear combination of the states:

$$\tilde{\mathrm{y}} = C\tilde{\mathrm{x}}\,. \tag{22}$$

Our objective is to eliminate the effects of errors described by the Fourier series (18) on the output. Due to the truncation of the Fourier series, higher-frequency components are not considered, and the order of the series would have to be increased in order to compensate for them.

The dimension of ỹ is not defined by the problem, and may be chosen by the user as the set of error outputs that should be minimized. If the dimension of ỹ is larger than the dimension of the adaptation control input v̂, a full correction of all errors cannot be expected.

The above equations (19)-(21) can be written in matrix form, and multiplied by C in order to describe the output tracking errors (22):

$$\begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} s_k \\ r_k \end{bmatrix} = \begin{bmatrix} -k\omega & -\bar{A} \\ -\bar{A} & k\omega \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix} \tag{23}$$

$$\underbrace{\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} -k\omega & -\bar{A} \\ -\bar{A} & k\omega \end{bmatrix}^{-1} \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix}}_{=:J} \begin{bmatrix} s_k \\ r_k \end{bmatrix} = \underbrace{\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix}}_{=:e_k} . \tag{24}$$

The matrix J represents the linear mapping of input coefficients s_k, r_k to tracking error output coefficients e_k, and is the equivalent to the nominal maneuver Jacobian for parameterized motions described in [10].

Let J^+ denote the Moore-Penrose pseudoinverse [1] of J (in the special case that J is square, $J^+ = J^{-1}$ holds). Note that the existence of the inverse is not given for all systems. For the purpose of this paper however, we assume its existence and intend to investigate this question further in the future. Assume that we have executed the trajectory for iteration $i-1$, and measured the tracking error $\tilde{y}^{i-1}$. Let e_k^{i-1} be the Fourier coefficients of the tracking error output of iteration $i-1$. The iteration-domain feedback law is

$$\begin{bmatrix} s_k \\ r_k \end{bmatrix}^i = \begin{bmatrix} s_k \\ r_k \end{bmatrix}^{i-1} - \gamma J^+ e_k^{i-1} \tag{25}$$

where γ is the step size parameter. The step size parameter can be used to trade off convergence of errors and noise rejection. This may be necessary because e_k^{i-1} usually contains components caused by non-repetitive process noise and measurement errors, and is therefore only an estimate of the true repeatable tracking error.

3.5 Convergence

Using the above feedback law, the tracking error Fourier coefficients e_k evolve as follows:

$$\begin{aligned} e_k^i &= e_k^{i-1} + J\left(\begin{bmatrix} s_k \\ r_k \end{bmatrix}^i - \begin{bmatrix} s_k \\ r_k \end{bmatrix}^{i-1}\right) && (26) \\ &= e_k^{i-1} - \gamma J J^+ e_k^{i-1} . && (27) \end{aligned}$$

If J^+ is the right inverse of J (the interpretation of this is that there are not more tracking error outputs than adaptation control inputs), it follows that the tracking error converges to zero for $0 < \gamma \leq 1$:

$$e_k^i = (1 - \gamma)\, e_k^{i-1}\,. \tag{28}$$

Note that if the tracking error dimension is lower than the input dimension, the correction term $J^+ e_k$ is the least-squares solution to an underconstrained set of equations [1], implying that the Euclidian norm of the Fourier coefficients is minimized. By Parseval's theorem [8], this is equivalent to minimizing the energy of the correction input signal $\hat{\mathrm{v}}$.

Now consider the case where J^+ is not the right inverse, implying that there are more tracking error outputs than inputs. In this case, the error dynamics are

$$\begin{aligned} e_k^i &= \left(I - \gamma J J^+\right) e_k^{i-1} && (29)\\ &= \left(I - \gamma J \left(J^T J\right)^{-1} J^T\right) e_k^{i-1} && (30) \end{aligned}$$

where I denotes the identity matrix and $(\cdot)^T$ denotes the transpose of a matrix. Assuming that J is full rank, we apply a singular value decomposition [1]:

$$\begin{aligned} J &= U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T && (31)\\ e_k^i &= \underbrace{\left(I - \gamma U \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U^T\right)}_{=:M} e_k^{i-1}\,. && (32) \end{aligned}$$

Using a similarity transformation [1], we show that the error does not diverge by computing the eigenvalues of M:

$$\begin{aligned} \operatorname{eig}(M) &= \operatorname{eig}\left(U^T M U\right) && (33)\\ &= \operatorname{eig}\left(I - \gamma \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}\right) && (34)\\ &= \left\{1-\gamma, \ldots, 1-\gamma, 1, \ldots, 1\right\}\,. && (35) \end{aligned}$$

This implies that, under the appropriate coordinate transformation, the tracking error coefficients either reduce to zero over time, or remain constant. Note that because this case implies fewer control inputs than tracking errors, it is to be expected that not all components of the tracking error Fourier series can be driven to zero.

4. Experimental results

This chapter presents the application of the learning algorithm to the flying inverted pendulum experiment introduced in Chapter 2. We start by presenting the experimental setup, then discuss the implementation of the learning algorithm, and finally show results obtained.

4.1 Experimental Setup

Experiments were carried out in the Flying Machine Arena, an aerial vehicle development platform at ETH Zurich [11]. The quadrocopters used in the experiments are modified Ascending Technologies 'Hummingbird' vehicles [6] equipped with custom electronics to allow greater control over the low-level control algorithms. A small cup-shaped pendulum mounting point is attached to the top of the vehicle, and the pendulum's center of mass is 59 cm away from its base. A photograph of the experimental setup is shown in Figure 3. An infrared motion tracking system measures the position and attitude of the vehicle, as well as the position of the pendulum. A Luenberger observer is used to filter the sensory data and provide full state information to the controller. The observer also compensates for systematic latencies occurring in the control loop by using the control inputs to project the system state into the future.

4.2 Implementation of the Learning Algorithm

The learning algorithm was implemented such that adaptation of the control inputs occurs without interrupting the circling motion of the vehicle. Each iteration of the learning algorithm consists of three steps:

1. Measure the tracking error $\tilde{\mathrm{y}}$ for multiple circles. While measuring the error over one circle would suffice, averaging it over several circles improves the rejection of non-repetitive disturbances. In our experiments, the tracking error was measured and averaged over three circles.
2. Compute the updated input correction Fourier series $\hat{\mathrm{v}}$ according to Equation (25).
3. Wait for several circles to allow the system to converge under the new input correction. Our experiments showed that this step is important in order to correctly measure the systematic tracking error in the next iteration. For the experiments presented in this paper, we chose to wait for two circles before starting the error measurement for the next iteration.

We chose the tracking error output to be the vehicle position error in the rotating coordinate system: $\tilde{\mathrm{y}} = (\tilde{u}, \tilde{v}, \tilde{w})$. This implies that the objective is for the vehicle to correctly track the horizontal circle in size, phase, and height. The nominal trajectory was set to a circle radius R of 0.3 m. Errors in the pendulum position $\tilde{p}, \tilde{q}$ are not penalized in these experiments.

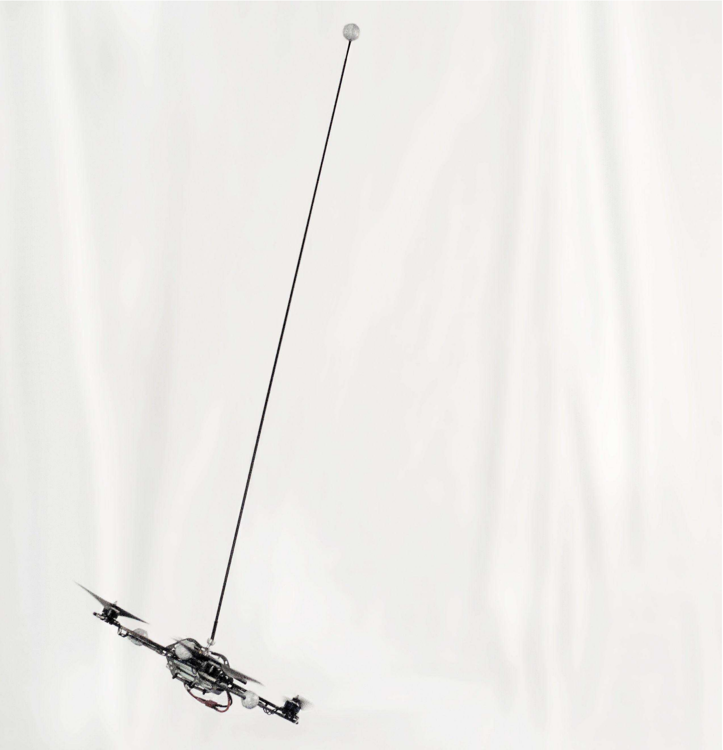

Figure 3. Photograph of experiments: The vehicle flying a circle while balancing the inverted pendulum.

The update rate γ was chosen to be 0.3, a value that showed a good trade-off between noise rejection and speed of convergence in our experiments. The order of the error and input Fourier series, N, was varied during experiments. It was found that orders higher than $N = 1$ had little impact on the tracking performance during our experiments.

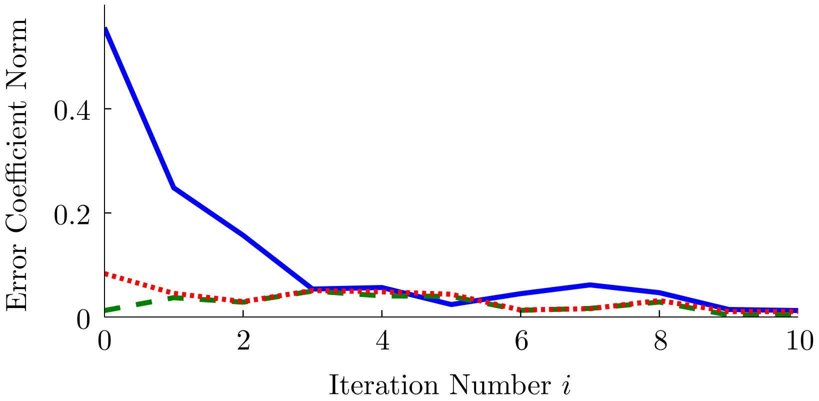

Figure 4. Evolution of the tracking error output Fourier coefficients during learning. The lines shown represent the Euclidian norm of the Fourier coefficients: $||Ca_0||$ (solid blue), $||Ca_1||$ (dashed green) and $||Cb_1||$ (dotted red). Higher order terms are significantly smaller, and have been omitted in this figure.

4.3 Results

The circular motion of the pendulum was started with all correction coefficients set to zero, i.e. $\hat{\mathrm{v}} = 0$. Figure 4 shows the Euclidian norm of the error Fourier series coefficients over ten iterations. It can be seen that initial tracking performance is relatively poor with peak tracking errors of 64 cm. The errors are then quickly reduced over the first three iterations, after which non-repeatable disturbances cause them to vary from iteration to iteration while small improvements are made. After ten iterations of the learning algorithm, the peak tracking error is reduced to 11 cm, and subsequent iterations do not improve tracking performance significantly.

Figure 5 shows the flight path of the vehicle in the horizontal plane before learning and after the tenth learning iteration. Note that the initial flight path shows large errors, with the flown circle being much too large, shifted from the desired centre point, and warped. After ten learning iterations, the tracking performance has improved considerably, although remaining, largely unrepeatable, disturbances still prevent the vehicle from following the trajectory perfectly.

5. Conclusion and Outlook

This paper evaluated an iterative adaptation scheme that improves tracking performance when periodic disturbances cause poor tracking under feedback control. We have derived convergence properties for the presented method, and have shown that our approach greatly improves performance in an experiment where a quadrocopter balances an inverted pendulum while flying circles.

The method was presented with a focus on the specific problem of trajectories for quadrocopters, and a number of assumptions were made in order to simplify derivations. As is, the method is limited to dynamic systems where the linearization around the nominal trajectory is – under an appropriate coordinate transformation – time-invariant. We intend to investigate transferability to more general system descriptions, and hope to perform more experiments to verify its applicability to other problems.

One open question is the choice of the order of the Fourier series used to represent tracking errors and correction inputs. While we chose the order manually for our experiments, more systematic approaches could be considered.

Furthermore, it would be worthwhile to investigate whether learnt correction values could be transferred to similar maneuvers, as proposed for Iterative Learning Control in [17]. This would be particularly applicable to our experiment because attempts to fly large circles with the inverted pendulum currently lead to crashes before learning permits correct tracking of the trajectory.

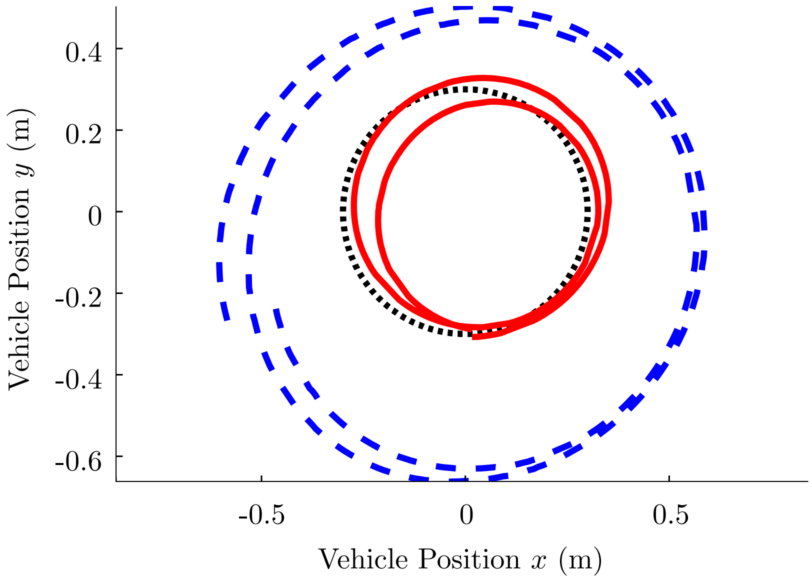

Figure 5. Horizontal flight trajectory of the quadrocopter over two rounds of the circular trajectory. The dashed blue line shows the flight path before learning, the solid red line shows flight after ten iterations of learning. The nominal trajectory is shown in dotted black. Errors in height are not shown; they were reduced from a maximum error of 0.42 m to a maximum of 0.01 m.

References

[1] Dennis S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, third edition, 2005.

[3] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A Survey of Iterative Learning Control. *Control Systems Magazine*, 26(3):96–114, 2006.

[4] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 3rd edition, 1998.

[5] Travis Dierks and Sarangapani Jagannathan. Output Feedback Control of a Quadrotor UAV Using Neural Networks. *IEEE Transactions on Neural Networks*, 21(1):50–66, 2010.

[6] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[7] Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. In *International Conference on Robotics and Automation*, 2011.

[8] Hwei P. Hsu. *Schaum's Outline of Signals and Systems*. McGraw-Hill, 1995.

[9] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering. In *International Conference on Robotics and Automation*, 2009.

[10] Sergei Lupashin and Raffaello D'Andrea. Adaptive Fast Open-Loop Maneuvers for Quadrocopters. *Autonomous Robots*, 33(1-2):89–102, 2012.

[11] Sergei Lupashin, Angela P. Schoellig, Markus Hehn, and Raffaello D'Andrea. The Flying Machine Arena as of 2010. In *International Conference on Robotics and Automation*, 2011.

[12] D. Mellinger, N. Michael, and V. Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[13] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *International Symposium on Experimental Robotics*, 2010.

[14] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[15] Mark Muller, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Ball Juggling. In *International Conference on Intelligent Robots and Systems*, 2011.

[16] Ivana Palunko and Rafael Fierro. Adaptive Control of a Quadrotor with Dynamic Changes in the Center of Gravity. In *IFAC World Congress*, 2011.

[17] Oliver Purwin and Raffaello D'Andrea. Performing and Extending Aggressive Maneuvers Using Iterative Learning Control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[18] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative Quadrocopter Ball Throwing and Catching. In *International Conference on Intelligent Robots and Systems*, 2012.

[19] Angela P Schoellig, Fabian L Mueller, and Raffaello D'Andrea. Optimization-Based Iterative Learning for Precise Quadrocopter Trajectory Tracking. *Autonomous Robots*, 33(1-2):103–127, 2012.

[20] Steven L. Waslander, Gabriel M. Hoffmann, Jung Soon Jang, and Claire J. Tomlin. Multi-Agent Quadrotor Testbed Control Design: Integral Sliding Mode vs. Reinforcement Learning. In *International Conference on Intelligent Robots and Systems*, 2005.

Paper VI

A Flying Inverted Pendulum

Markus Hehn and Raffaello D'Andrea

Abstract

We extend the classic control problem of the inverted pendulum by placing the pendulum on top of a quadrotor aerial vehicle. Both static and dynamic equilibria of the system are investigated to find nominal states of the system at standstill and on circular trajectories. Control laws are designed around these nominal trajectories. A yaw-independent description of quadrotor dynamics is introduced, using a 'Virtual Body Frame'. This allows for the time-invariant description of curved trajectories. The balancing performance of the controller is demonstrated in the ETH Zurich Flying Machine Arena testbed. Development potential for the future is highlighted, with a focus on applying learning methodology to increase performance by eliminating systematic errors that were seen in experiments.

Published in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011. The publication includes a multimedia attachment available at `www.hehn.be` and `ieeexplore.ieee.org`.
DOI: 10.1109/ICRA.2011.5980244

©2011 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

1. Introduction

The inverted pendulum is a classic control problem, offering one of the most intuitive, easily describable and realizable nonlinear unstable systems. It has been investigated for several decades (see, for example, [11], and references therein). It is frequently used as a demonstrator to showcase theoretical advances, e.g. in reinforcement learning [6], neural networks [14], and fuzzy control [13].

In this paper, we develop a control strategy that enables an inverted pendulum to balance on top of a quadrotor. Besides being a highly visual demonstration of the dynamic capabilities of modern quadrotors, the solution to such a complex control problem offers insight into quadrotor control strategies, and could be adapted to other tasks.

Quadrotors offer exceptional agility. Thanks to the off-center mounting of the propellers, extraordinarily fast rotational dynamics can be achieved. This is combined with typically high thrust-to-weight ratios, resulting in large achievable translational accelerations when not carrying a payload.

While most early work on quadrotors focused on near-hover operation (e.g. [5], and references therein), a growing community is working on using the full dynamical potential of these vehicles. Flips have been executed by several groups, some focusing on speed and autonomous learning [7] and some on safety guarantees [3]. Other complex maneuvers, including flight through windows and perching have been demonstrated [8].

In Section 2, we introduce the dynamic models used in the controller design. Section 3 presents static and dynamic nominal trajectories for the quadrotor to follow. The dynamics are then linearized around these trajectories in Section 4, and linear state feedback controllers are designed in Section 5. The experimental setup and results are shown in Section 6 and conclusions are drawn in Section 7, where an outlook is also presented.

2. Dynamics

We derive the equations of motion of the quadrotor and the inverted pendulum for the trajectory-independent general case.

Given that the mass of the pendulum is small compared to the mass of the quadrotor, it is reasonable to assume that the pendulum's reactive forces on the quadrotor are negligible. The dynamics of the quadrotor, then, do not depend on the pendulum, whereas the dynamics of the pendulum are influenced by the motion of the quadrotor. This assumption is justified by the experimental setup, with the weight of the pendulum being less than 5% of that of the quadrotor vehicle.

2.1 Quadrotor

The quadrotor is described by six degrees of freedom: The translational position (x, y, z) is measured in the inertial coordinate system **O** as shown in Figure 1. The vehicle

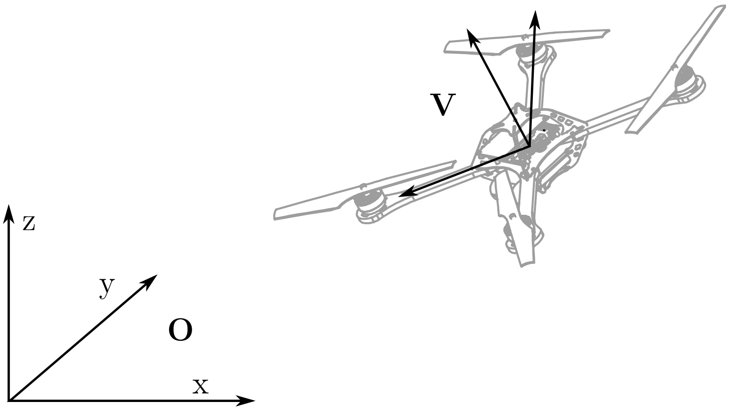

Figure 1. The inertial coordinate system **O** and the vehicle coordinate system **V**.

attitude **V** is defined by three Euler angles. From the inertial coordinate system, we first rotate around the z-axis by the yaw angle α. The coordinate system is then rotated around the new y-axis by the pitch angle β, and finally rotated about the new x-axis by the roll angle γ:

$$ {}^{O}_{V}R(\alpha,\beta,\gamma) = R_z\left(\alpha\right) R_y\left(\beta\right) R_x\left(\gamma\right) \ , \tag{1}$$

where

$$R_x\left(\gamma\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \ , \tag{2}$$

$$R_y\left(\beta\right) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \ , \tag{3}$$

$$R_z\left(\alpha\right) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \ . \tag{4}$$

The translational acceleration of the vehicle is dictated by the attitude of the vehicle and the total thrust produced by the four propellers. With a representing the mass-normalized collective thrust, the translational acceleration in the inertial frame is

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}^{O}_{V}R(\alpha,\beta,\gamma) \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix} \ . \tag{5}$$

The vehicle attitude is not directly controllable, but it is subject to dynamics. The control inputs are the desired rotational rates about the vehicle body axes, $(\omega_x, \omega_y, \omega_z)$, and the mass-normalized collective thrust, a, as shown in Figure 2. High-bandwidth controllers on the vehicle track the desired rates using feedback from gyroscopes. The quadrotor has very low rotational inertia, and can produce high torques due to the outward mounting of the propellers, resulting in very high achievable rotational accelerations on the order of $200\,\mathrm{rad\,s^{-2}}$. The vehicle has a fast response time to changes in the desired rotational rate (experimental results have shown time constants on the order of 20 ms). We will therefore assume that we can directly control the vehicle body rates and ignore rotational acceleration dynamics. As with the vehicle body rates, we assume that the thrust can be changed instantaneously. Experimental results have shown that the true thrust dynamics are about as fast as the rotational dynamics, with propeller spin-up being noticeably faster than spin-down.

The rates of the Euler angles are converted to the vehicle body coordinate system V through their respective transformations:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\gamma} \\ 0 \\ 0 \end{bmatrix} + R_x^{-1}(\gamma) \begin{bmatrix} 0 \\ \dot{\beta} \\ 0 \end{bmatrix} + R_x^{-1}(\gamma)\, R_y^{-1}(\beta) \begin{bmatrix} 0 \\ 0 \\ \dot{\alpha} \end{bmatrix} . \tag{6}$$

The above can be written more compactly by combining the Euler rates into a single vector, calculating the relevant rows of the rotation matrices, and solving for the Euler angle rates:

$$\begin{bmatrix} \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\gamma & -\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\gamma & 0 \\ -\sin\beta & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} . \tag{7}$$

2.2 Inverted Pendulum

The pendulum has two degrees of freedom, which we describe by the translational position of the pendulum center of mass relative to its base in $\mathbf{O}$ (r along the x-axis, s along the y-axis). For notational simplicity, we describe the relative position of the pendulum along the z-axis as

$$\zeta := \sqrt{L^2 - r^2 - s^2} \ , \tag{8}$$

where L to denotes the length from the base of the pendulum to its center of mass. We model the pendulum as an inertialess point mass that is rigidly attached to the mass center of the quadrotor, such that rotations of the vehicle do not cause a motion of the pendulum base. In the experimental setup, the point that the pendulum is attached to is mounted off-center by about 10% of the length of the pendulum. While this assumption causes modeling errors, it simplifies the dynamics to such a great extent that the problem

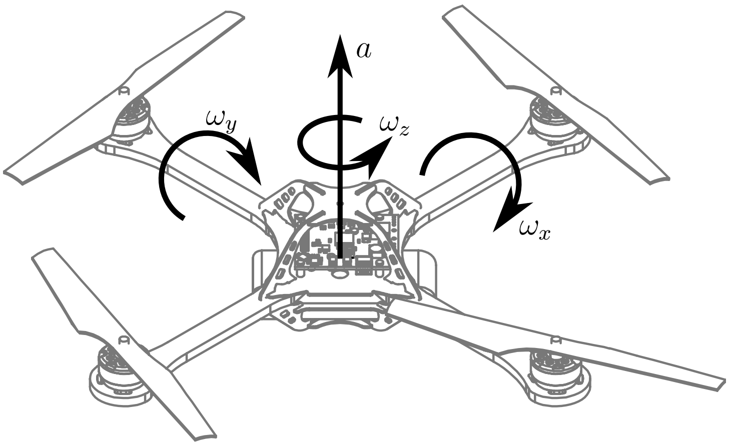

Figure 2. The control inputs of the quadrotor: The rotational rates ω_x, ω_y, and ω_z are tracked by an on-board controller, using gyroscope feedback.

becomes much more tractable. The Lagrangian [9] of the pendulum can be written as

$$\begin{aligned}\mathscr{L} = \frac{1}{2}\left((\dot{x}+\dot{r})^2 + (\dot{y}+\dot{s})^2 + (\dot{z} - \frac{r\dot{r}+s\dot{s}}{\zeta})^2\right) \\ - \mathrm{g}\,(z+\zeta)\ ,\end{aligned} \tag{9}$$

where we assume unit pendulum mass without loss of generality. The first term represents the kinetic energy of the pendulum, and the second the potential energy. The full, nonlinear dynamic equations can be derived from $\mathscr{L}$ using conventional Lagrangian mechanics:

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{r}}\right) - \frac{\partial \mathscr{L}}{\partial r} = 0 \tag{10}$$

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{s}}\right) - \frac{\partial \mathscr{L}}{\partial s} = 0\ , \tag{11}$$

resulting in a system of equations of the form

$$\begin{bmatrix}\ddot{r}\\ \ddot{s}\end{bmatrix} = \mathrm{f}\,(r, s, \dot{r}, \dot{s}, \ddot{x}, \ddot{y}, \ddot{z})\ , \tag{12}$$

where f are the nonlinear equations (13) and (14).

2.3 Combined dynamics

The full dynamics of the combined system are described entirely by Equations (5), (7), and (12). The three body rate control inputs $(\omega_x, \omega_y, \omega_z)$ control the attitude $\mathbf{V}$ of the

$$\begin{aligned}
\ddot{r} = \frac{1}{(L^2-s^2)\zeta^2}\Big(& -r^4\ddot{x} - \left(L^2-s^2\right)^2\ddot{x} \\
& - 2r^2\left(s\dot{r}\dot{s} + \left(-L^2+s^2\right)\ddot{x}\right) + r^3\left(\dot{s}^2 + s\ddot{s} - \zeta\left(\mathrm{g}+\ddot{z}\right)\right) \\
& + r\left(-L^2 s\ddot{s} + s^3\ddot{s} + s^2\left(\dot{r}^2 - \zeta\left(\mathrm{g}+\ddot{z}\right)\right) + L^2\left(-\dot{r}^2 - \dot{s}^2 + \zeta\left(\mathrm{g}+\ddot{z}\right)\right)\right)\Big)
\end{aligned} \tag{13}$$

$$\begin{aligned}
\ddot{s} = \frac{1}{(L^2-r^2)\zeta^2}\Big(& -s^4\ddot{y} - \left(L^2-r^2\right)^2\ddot{y} \\
& - 2s^2\left(r\dot{r}\dot{s} + \left(-L^2+r^2\right)\ddot{y}\right) + s^3\left(\dot{r}^2 + r\ddot{r} - \zeta\left(\mathrm{g}+\ddot{z}\right)\right) \\
& + s\left(-L^2 r\ddot{r} + r^3\ddot{r} + r^2\left(\dot{s}^2 - \zeta\left(\mathrm{g}+\ddot{z}\right)\right) + L^2\left(-\dot{r}^2 - \dot{s}^2 + \zeta\left(\mathrm{g}+\ddot{z}\right)\right)\right)\Big)
\end{aligned} \tag{14}$$

vehicle in a nonlinear fashion. This attitude, combined with the thrust a, controls the translational acceleration of the vehicle. While the acceleration drives the translational motion of the vehicle linearly, it also drives the motion of the pendulum through nonlinear equations. The combined system consists of thirteen states (three rotational and six translational states of the quadrotor, and four states of the pendulum), and four control inputs (three body rates, and the thrust).

3. Nominal trajectories

In this section, we find static and dynamic equilibria of the system that satisfy Equations (5), (7), and (12). These are used as nominal trajectories to be followed by the quadrotor. Corresponding nominal control inputs are also described. We denote nominal values by a zero index (x_0, r_0, etc.).

3.1 Constant position

In a first case, we require x_0, y_0, and z_0 to be constant. Substituting these constraints into (5), it can be seen that $\beta_0 = 0$ and $\gamma_0 = 0$ solve the equations with $a_0 = g$, while α_0 can be chosen freely. We arbitrarily choose to set $\alpha_0 = 0$.

Using the given angles $(\alpha_0, \beta_0, \gamma_0)$, equation (7) can be solved with the body rate control inputs being $\omega_{x0} = \omega_{y_0} = \omega_{z0} = 0$.

Inserting the nominal states (x_0, y_0, z_0) into the pendulum equations of motion (12), they simplify to

$$\ddot{r} = r\frac{\mathrm{g}\zeta^3 - L^2\left(\dot{r}^2+\dot{s}^2\right) + \left(s\dot{r} - r\dot{s}\right)^2}{L^2\zeta^2}\ , \tag{15}$$

$$\ddot{s} = s\frac{\mathrm{g}\zeta^3 - L^2\left(\dot{r}^2+\dot{s}^2\right) + \left(s\dot{r} - r\dot{s}\right)^2}{L^2\zeta^2}\ . \tag{16}$$

These equations are solved by the static equilibrium

$$r = r_0 = 0 \tag{17}$$

$$s = s_0 = 0, \tag{18}$$

meaning that, as expected, the inverted pendulum is exactly over the quadrotor.

3.2 Circular trajectory

As a second nominal trajectory, the quadrotor is required to fly a circle of a given radius R at a constant rotational rate Ω, at a constant altitude z_0.

We seek to transform the equations of motion into different coordinate systems, such that the nominal states and the linearized dynamics about them can be described in a time-invariant manner.

To describe the vehicle position, the following coordinate system **C** is introduced, with (u, v, w) describing the position in **C**:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} =: R_z(\Omega t) \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \cos\Omega t & -\sin\Omega t & 0 \\ \sin\Omega t & \cos\Omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} . \tag{19}$$

To describe the vehicle attitude, a second set of Euler angles is introduced, describing the ‘virtual body frame’ **W** and named η, μ, and ν:

$$ {}^{O}_{W}R(\eta, \mu, \nu) = R_z(\eta)\, R_y(\mu)\, R_x(\nu) \ , \tag{20}$$

subject to the constraint that

$$ {}^{O}_{V}R(\alpha, \beta, \gamma) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = {}^{O}_{W}R(\eta, \mu, \nu) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \ . \tag{21}$$

The above equation defines values for three elements of the rotation matrices. As every column of a rotation matrix has unit norm [12], however, this equation only defines two of the angles (η, μ, ν).

Comparing the constraint (21) with the translational equation of motion of the vehicle (5), it is straightforward to see that the virtual body frame **W** represents an attitude that is constrained such that it effects the same translational motion of the quadrotor as the vehicle attitude **V**. The remaining degree of freedom represents the fact that rotations about the axis along which ω_z acts have no affect on the quadrotors translational motion.

Applying (19), its derivatives, (21), and setting the free parameter $\eta = \Omega t$, the quadrotor equation of motion (5) simplifies to

$$\begin{bmatrix} \ddot{u} \\ \ddot{v} \\ \ddot{w} \end{bmatrix} = \begin{bmatrix} a \sin\mu \cos\nu + \Omega^2 u + 2\Omega\dot{v} \\ -a \sin\nu - 2\Omega\dot{u} + \Omega^2 v \\ a \cos\mu \cos\nu - \mathrm{g} \end{bmatrix} . \tag{22}$$

The circular trajectory is described by $u_0 = R$, $v_0 = 0$, and $\dot{w}_0 = 0$. Using these values, the nominal Euler angles μ_0 and ν_0, and the nominal thrust a_0 can be calculated:

$$\mu_0 = \arctan(-\frac{\Omega^2 R}{\mathrm{g}}) , \tag{23}$$

$$\nu_0 = 0 , \tag{24}$$

$$a_0 = \sqrt{\mathrm{g}^2 + (\Omega^2 R)^2} . \tag{25}$$

Knowing the nominal values for (η_0, μ_0, ν_0), we solve for $(\alpha_0, \beta_0, \gamma_0)$ using Equation (21). Analogous to the constant position case, we set $\alpha_0 = 0$, simplifying (21) to

$$\begin{bmatrix} \sin\beta_0 \cos\gamma_0 \\ -\sin\gamma_0 \\ \cos\beta_0 \cos\gamma_0 \end{bmatrix} = \begin{bmatrix} \cos\Omega t \sin\mu_0 \cos\nu_0 + \sin\Omega t \sin\nu_0 \\ \sin\Omega t \sin\mu_0 \cos\nu_0 - \cos\Omega t \sin\nu_0 \\ \cos\mu_0 \cos\nu_0 \end{bmatrix} , \tag{26}$$

which can be solved for β_0 and γ_0. This completes the description of the nominal states required for the translational motion (5): In the coordinate systems **C** and **W**, the nominal position and attitude are constant. Using Equations (19) and (26), the time-varying nominal states in **O** and **V** may be found.

To calculate the rotational rate control inputs in Equation (7), we take the first derivative of Equation (26). It can be shown that

$$\dot{\beta}_0 = \frac{R\Omega^3 \cos^{-1}\gamma_0 (\tan\beta_0 \tan\gamma_0 \cos(\Omega t) + \cos^{-1}\beta_0 \sin(\Omega t))}{\sqrt{\mathrm{g}^2 + (\Omega^2 R)^2}} \tag{27}$$

$$\dot{\gamma}_0 = \frac{R\Omega^3 \cos^{-1}\gamma_0 \cos(\Omega t)}{\sqrt{\mathrm{g}^2 + (\Omega^2 R)^2}} . \tag{28}$$

Combining these equations with the results from Equations (26) and (7), the nominal states can be solved for the nominal control inputs $(\omega_{x0}, \omega_{y_0}, \omega_{z0})$. The full derivation is made available online at `www.idsc.ethz.ch/people/staff/hehn-m` .

Identically to the vehicle, the pendulum relative coordinates r and s are rotated by Ωt:

$$\begin{bmatrix} r \\ s \end{bmatrix} =: \begin{bmatrix} \cos \Omega t & -\sin \Omega t \\ \sin \Omega t & \cos \Omega t \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} . \tag{29}$$

Applying this rotation to the Lagrangian derivations of the motion of the pendulum (10), (11) and setting the base motion $(\ddot{x}, \ddot{y}, \ddot{z})$ to the circular trajectory, the pendulum dynamics can be shown to be

$$\begin{aligned} & p\left(\frac{p\ddot{p} + \dot{p}^2 + q\ddot{q} + \dot{q}^2}{\zeta^2} + \frac{q^2\dot{q}^2 + p^2\dot{p}^2 + 2pq\dot{p}\dot{q}}{\zeta^4} - \frac{\mathrm{g}}{\zeta} - \Omega^2\right) \\ & + \ddot{p} - 2\Omega\dot{q} - R\Omega^2 = 0 \end{aligned} \tag{30}$$

$$\begin{aligned} & q\left(\frac{q\ddot{q} + \dot{q}^2 + p\ddot{p} + \dot{p}^2}{\zeta^2} + \frac{p^2\dot{p}^2 + 2pq\dot{p}\dot{q} + q^2\dot{q}^2}{\zeta^4} - \frac{\mathrm{g}}{\zeta} - \Omega^2\right) \\ & + \ddot{q} + 2\Omega\dot{p} = 0 \end{aligned} \tag{31}$$

We seek a solution where $\dot{p}_0 = \dot{q}_0 = \ddot{p}_0 = \ddot{q}_0 = 0$, leading to the following constraints for equilibrium points:

$$\Omega^2(q_0) + \frac{\mathrm{g}q_0}{\zeta_0} = 0 \ , \tag{32}$$

$$\Omega^2(R + p_0) + \frac{\mathrm{g}p_0}{\zeta_0} = 0 \ . \tag{33}$$

The only solution to the first equation is $q_0 = 0$. The second equation defines a nonlinear relationship between Ω, R, and p_0, with a solution p_0 always existing in the range $-R \leq p_0 \leq 0$. This result is intuitive: The center of mass of the pendulum must lie towards the center of the circle, such that the centripetal force acting on it is compensated for by gravity. If the center of mass were to lie further than R inwards, the centripetal force would change sign, making the pendulum fall.

4. Dynamics about nominal trajectories

Linear approximate dynamics are derived through a first-order Taylor expansion of the equations of motion (5), (7), and (12) about the nominal trajectories found in Section 3. We denote small deviations by a tilde ($\tilde{x}$, $\tilde{r}$, etc.). We present only the resulting linearized dynamics. The corresponding derivations are not shown in this paper due to space constraints, but are made available online.

4.1 Constant position

Assuming a constant nominal position and zero yaw angle, the three translational degrees of freedom of the quadrotor-pendulum system decouple entirely along the three axes of the **O** coordinate system, resulting in the following equations:

$$\ddot{\tilde{r}} = \tilde{r}\frac{\mathrm{g}}{L} - \tilde{\beta}\mathrm{g} \tag{34}$$

$$\ddot{\tilde{s}} = \tilde{s}\frac{\mathrm{g}}{L} + \tilde{\gamma}\mathrm{g} \tag{35}$$

$$\ddot{\tilde{x}} = \tilde{\beta}\mathrm{g} \tag{36}$$

$$\ddot{\tilde{y}} = -\tilde{\gamma}\mathrm{g} \tag{37}$$

$$\dot{\tilde{\beta}} = \tilde{\omega}_y \tag{38}$$

$$\dot{\tilde{\gamma}} = \tilde{\omega}_x \tag{39}$$

$$\ddot{\tilde{z}} = \tilde{a} \tag{40}$$

The two horizontal degrees of freedom represent fifth-order systems, with the vehicle forming a triple integrator from the body rate to its position. The vertical motion is represented by a double integrator from thrust to position.

4.2 Circular trajectory

To derive linear dynamics about the circular trajectory, the Euler angle rates $\dot{\tilde{\mu}}$ and $\dot{\tilde{\nu}}$ are treated as control inputs. Solving the time-derivative of equation (21) (analogously to solutions (27) and (28) for the nominal case) allows the calculation of $(\dot{\tilde{\beta}}, \dot{\tilde{\gamma}})$. These can then be converted to the true inputs $(\tilde{\omega_x}, \tilde{\omega_y}, \tilde{\omega_z})$ using Equation (7).

In contrast to a constant nominal position, the dynamics on a circular trajectory do not decouple. The linearized equations of motion can be shown to be

$$\begin{aligned}\ddot{\tilde{p}} = \frac{\zeta_0^2}{L^2}\Bigg[&\tilde{p}(\Omega^2 + \frac{\mathrm{g}L^2}{\zeta_0^3}) + 2\dot{\tilde{q}}\Omega + \\ &\tilde{\mu}(-\frac{p_0}{\zeta_0}a_0 \sin\mu_0 - a_0\cos\mu_0) + \tilde{a}(\frac{p_0}{\zeta_0}\cos\mu_0 - \sin\mu_0)\Bigg]\end{aligned} \tag{41}$$

$$\ddot{\tilde{q}} = \tilde{q}(\Omega^2 + \frac{\mathrm{g}}{\zeta_0}) - 2\dot{\tilde{p}}\Omega + \tilde{\nu}a_0 \tag{42}$$

$$\ddot{\tilde{u}} = \tilde{a}\sin\mu_0 + \tilde{\mu}a_0\cos\mu_0 + 2\dot{\tilde{v}}\Omega + \tilde{u}\Omega^2 \tag{43}$$

$$\ddot{\tilde{v}} = -\tilde{\nu}a_0 - 2\dot{\tilde{u}}\Omega + \tilde{v}\Omega^2 \tag{44}$$

$$\ddot{\tilde{w}} = \tilde{a}\cos\mu_0 - \tilde{\mu}a_0\sin\mu_0 \tag{45}$$

We observe that the linearized dynamics are indeed time-invariant in the coordinate systems **C** and **W**.

The above equations reduce to Equations (34) – (40) if setting $R = 0$ and $\Omega = 0$. If $R = 0$, Ω is a free parameter (see Equation (33)), allowing the description of the dynamics (34) – (40) in a rotating coordinate system.

5. Controller Design

We design linear full state feedback controllers to stabilize the system about its nominal trajectories. We use an infinite-horizon linear-quadratic regulator (LQR) design [1] and determine suitable weighting matrices.

5.1 Constant position

Because the system is decoupled in its nominal state, the control design process can be separated. The two horizontal degrees of freedom are single-input, five-state systems. The vertical degree of freedom is a single-input, two-state system. Although simpler design methods exist for such systems, LQR is used to make the results easily transferable to the design for a circular trajectory. We design a lateral controller that is identically applied to the $\tilde{x}$-$\tilde{r}$-system and the $\tilde{y}$-$\tilde{s}$-system (except for different signs mirroring the signs in the equations of motion (34)–(37)). A controller for the vertical direction is designed separately.

For the lateral controllers, we penalize only the vehicle position ($\tilde{x}$ or $\tilde{y}$) and the control effort ($\tilde{\omega}_y$ or $\tilde{\omega}_x$). There is no penalty on the pendulum state. One tuning parameter remains: The ratio of the penalties on position and control effort controls the speed at which the position set point is tracked. Values for this ratio are tuned manually until the system shows fast performance, without saturating the control inputs. This tuning is initially carried out in simulation, and then refined on the experimental setup.

The vertical controller is tuned much in the same manner as the lateral controller. Here again, we tune only the ratio between penalties on position errors and control effort until satisfying performance is achieved.

5.2 Circular trajectory

On the circular trajectory, the system represents a thirteen-state system with three control inputs that cannot readily be decoupled. To more easily tune the weighting matrices, we use the same approach as in the constant position case and penalize only the position errors ($\tilde{u}$, $\tilde{v}$, and $\tilde{w}$) and control effort ($\dot{\tilde{\mu}}$, $\dot{\tilde{\nu}}$, and $\tilde{a}$). The relative size of weights on control inputs and states is carried over from the standstill design. Because the controller for the vertical axis was tuned separately in the standstill case, the relative size of the penalties on the horizontal positions ($\tilde{u}$, $\tilde{v}$) and the vertical position ($\tilde{w}$) is adapted. Again, this is first carried out in simulation and then improved upon using the experimental testbed.

6. Experimental results

The algorithms presented herein were implemented in the Flying Machine Arena, an aerial vehicle development platform at ETH Zurich [7]. We present results demonstrating the performance of the controllers designed in the previous Section.

6.1 Experimental setup

We use modified Ascending Technologies ‘Hummingbird’ quadrotors [4]. The vehicles are equipped with custom electronics, allowing greater control of the vehicle’s response to control inputs, a higher dynamic range, and extended interfaces [7]. A small cup-shaped pendulum mounting point is attached to the top of the vehicle, approximately 5 cm above the geometrical center of the vehicle. The pendulum can rotate freely about the mounting point up to an angle of approximately 50 degrees. At larger angles, the mounting point offers no support and the pendulum falls off the vehicle.

Commands are sent through a proprietary low-latency 2.4 GHz radio link at a frequency of 50 Hz. Command loss is in the range of 0.1%. An infrared motion tracking system provides precise vehicle position and attitude measurements at 200 Hz, using

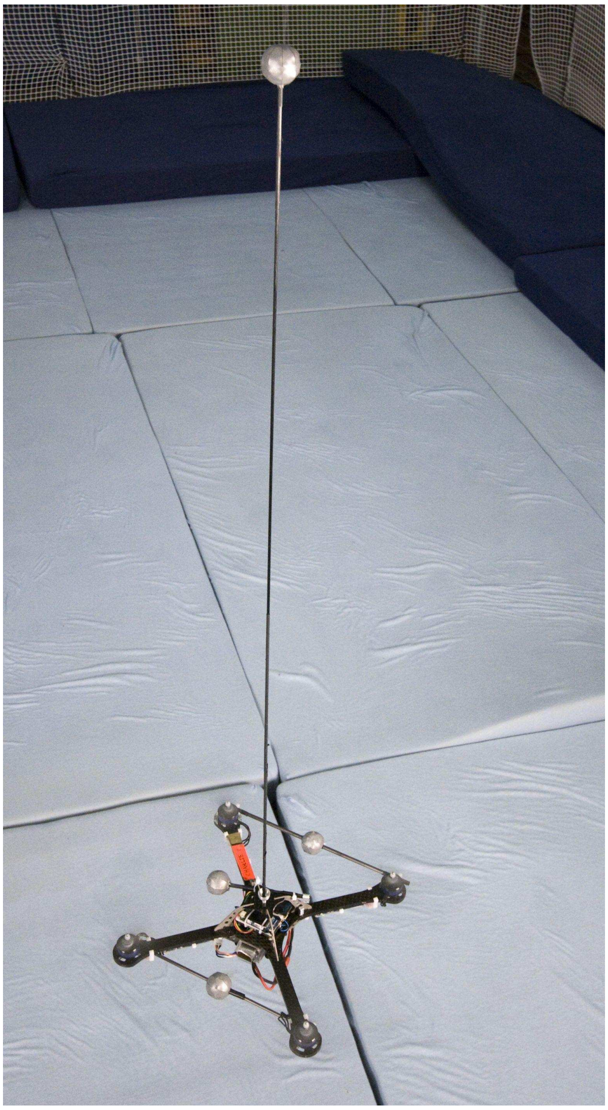

Figure 3. The quadrotor balancing the pendulum, at standstill. The mass center is about half-length of the pendulum.

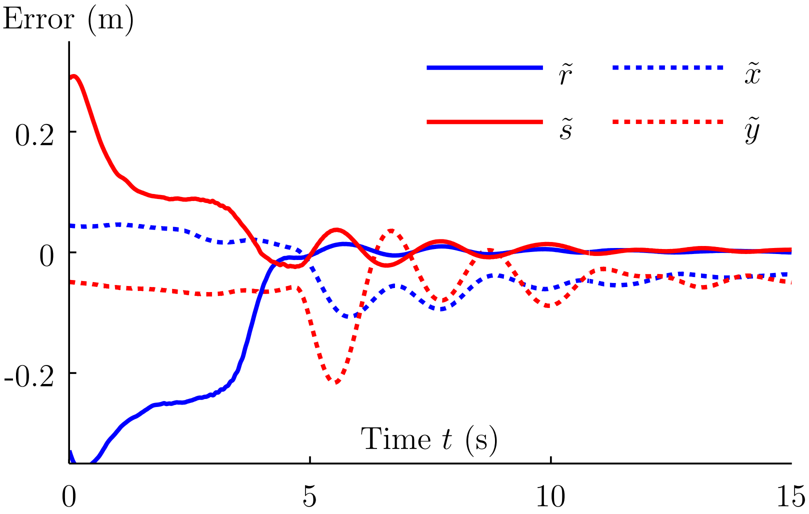

Figure 4. State errors during balancing: Pendulum position error $(\tilde{r}, \tilde{s})$ and quadrotor position error $(\tilde{x}, \tilde{y})$. The pendulum is manually placed on the vehicle at approximately $t = 4.25$ s, and the balancing controller is switched on at approximately $t = 4.75$ s.

retro-reflective markers mounted to the vehicle. The total closed-loop latency is approximately 30 ms.

The inverted pendulum consists of a carbon fiber tube, measuring 1.15 m in length. The top end of the pendulum carries a retro-reflective marker, allowing the position of this point to be determined through the motion tracking system in the same manner as vehicles are located. The center of mass of the pendulum is 0.565 m away from its base. Figure 3 shows the quadrotor and the pendulum.

Conventional desktop computers are used to run all control algorithms, with one computer acting as an interface to the testbed. Data is exchanged over Ethernet connections. A Luenberger observer is used to filter the sensory data and provide full state information to the controller. The observer also compensates for systematic latencies occurring in the control loop, using the known control inputs to project the system state into the future.

6.2 Constant position

Experiments are initialized by manually placing the pendulum on the mounting point. The vehicle holds a constant position using a separate controller, waiting for the pendulum. The balancing controller is switched on if $\tilde{r}$ and $\tilde{s}$ are sufficiently small for 0.5 seconds.

Figure 4 shows the pendulum position errors $(\tilde{r}, \tilde{s})$ and the horizontal quadrotor position errors $(\tilde{x}, \tilde{y})$. The pendulum is placed on the vehicle at approximately $t = 4.25$ s, and the control is switched from position holding to balancing at approximately $t = 4.75$ s. The pendulum position errors are relatively large in the beginning, but quickly converge to values close to zero. The vehicle settles at a stationary offset on the order of 5 cm from the desired position. Note that the balancing controller does not provide feedback on integrated errors. The main suspected reason for these steady-state errors are miscalibrations in the system: Errors in the vehicle attitude measurement lead to the linear controller trading off the attitude error and the position error, and biased measurements of the on-board gyroscopes result in a biased response to control inputs.

Though originally designed for a constant position, this controller has been successfully tested for set point tracking at moderate speeds. A video of this is available online at `www.idsc.ethz.ch/people/staff/hehn-m` .

6.3 Circular trajectory

We arbitrarily choose to set $p_0 = -\frac{R}{2}$, bringing the center of mass of the pendulum halfway between the vehicle and the circle center, nominally. Assuming a given R, this fixes Ω through the equilibrium constraint (33). Figure 5 shows the system performance when circling. The pendulum is first balanced at a constant position. At $t = 2\,\mathrm{s}$, a switch to a circular nominal trajectory and a corresponding controller occurs, with $R = 0.1\,\mathrm{m}$. The controller is seen to stabilize the pendulum, with the pendulum relative position errors in the rotating coordinate system (p, q) converging to non-zero values. The vehicle errors show two distinct components: Like the pendulum errors, there is a non-zero mean error. Additionally, the error oscillates at the rotational rate Ω, representing a near-constant position error in an inertial coordinate system. Figure 6 shows a comparison of the actual and nominal trajectories of the vehicle and pendulum in the inertial coordinate system $\mathbf{O}$. It confirms that the oscillating errors in the rotating coordinate system are constant position errors in the inertial coordinate system, with a magnitude of approximately 0.1m. The mean errors in $\mathbf{C}$ are represented by the circle radius being significantly larger than the nominal value R.

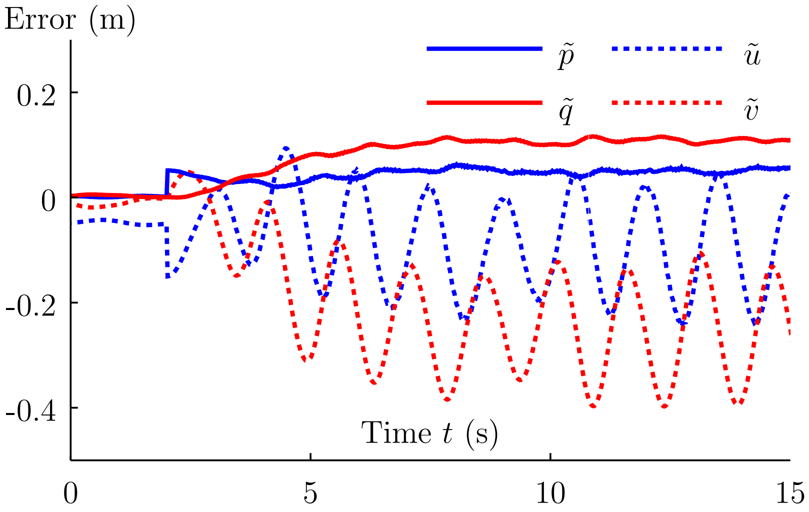

Figure 5. Errors in the rotating coordinate system: Pendulum position error $(\tilde{p}, \tilde{q})$ and quadrotor position error $(\tilde{u}, \tilde{v})$. The pendulum is balanced by the quadrotor during the entire duration shown on this plot. At $t = 2\,\mathrm{s}$, the controller is switched from a constant nominal position to a circular trajectory with $R = 0.1\,\mathrm{m}$.

A video showing the experiments of both cases presented herein is available online at `www.idsc.ethz.ch/people/staff/hehn-m` .

6.4 Comparison with simulation results

The controllers have also been tested in simulation. The Flying Machine Arena software environment allows the testing of controllers by simply re-routing the controller's outputs to a simulation. The simulation reproduces the behavior of the entire system. It includes the full dynamics of the quadrotor, including the on-board control loops, rotational accelerations, and propeller dynamics. It also reproduces system latencies and the noise characteristics of sensors. As the simulation output mimics the motion system's output as closely as possible, the same state observer is employed in reality and in simulation. This simulation environment has been extended to include the inverted pendulum. The pendulum is modeled with its full nonlinear dynamics (12), neglecting the off-center mounting on the vehicle for reasons of tractability, as discussed in Section 2.2. Figures 7 and 8 show the exact same circular trajectory experiment that was carried out on the testbed (Section 6.3). For this simulation, systematic errors in the gyroscopic sensors and the motion tracking system were disabled. In the initial standstill phase (the first two seconds in Figure 7), all errors are close to zero. During circling, the errors converge to nearly stationary values that are significantly smaller than in the real testbed. The close match of the vehicle's nominal trajectory and the pendulum's simulated trajectory in Figure 8 appears to be coincidental.

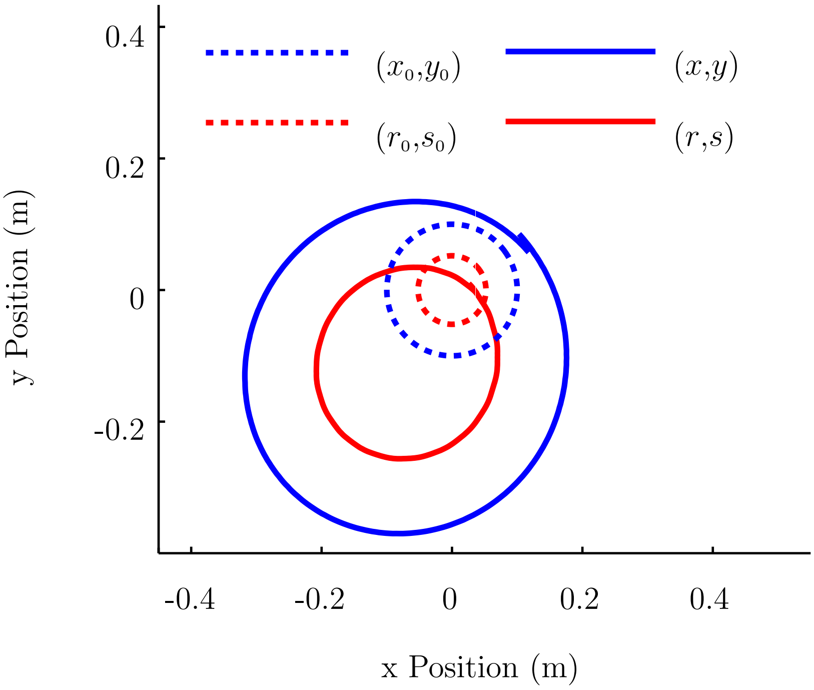

Figure 6. The trajectory of the quadrotor and pendulum in **O**, compared to the nominal trajectory.

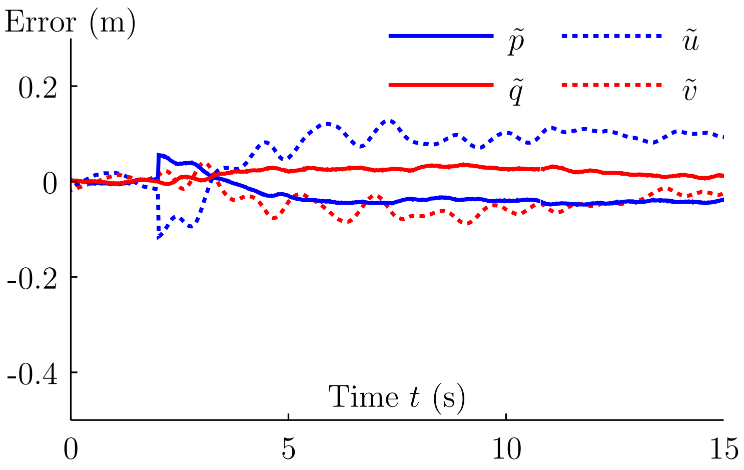

Figure 7. Simulation results: Errors in the rotating coordinate system: Pendulum position error $(\tilde{p}, \tilde{q})$ and quadrotor position error $(\tilde{u}, \tilde{v})$. At $t = 2\,\mathrm{s}$, the controller is switched from a constant nominal position to a circular trajectory with R =0.1m.

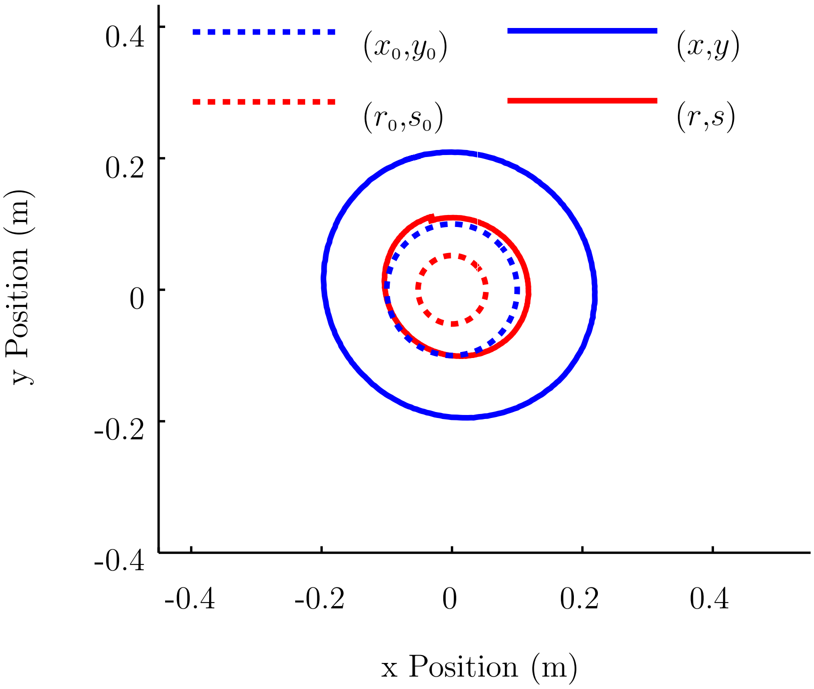

Figure 8. Simulation results: The trajectory of the quadrotor and pendulum in **O**, compared to the nominal trajectory.

This result highlights the influence of biased sensory information, leading to position errors in the inertial coordinate system **O**. These are observed as oscillating errors in **C**.

The mean errors on the trajectory also show different characteristics in simulation and reality. This is particularly noticeable in the pendulum position error $\tilde{p}$ and $\tilde{q}$. The simulation contains a detailed dynamic model of the quadrotor that has been validated in several experiments. It is therefore probable that the differences are due mainly to the simulated pendulum dynamics. The non-modeled off-center mounting of the pendulum could explain this discrepancy.

Circle radii R of up to $0.5\,\mathrm{m}$ have been successfully tested in simulation and reality. The vehicle and pendulum positioning errors increase with the circle size, but the controllers are still capable of keeping the pendulum in balance.

7. Conclusion and Outlook

We have developed linear controllers for stabilizing a pendulum on a quadrotor, which can be used for both static and dynamic equilibria of the pendulum. The virtual body frame is a useful tool to describe motions in a convenient coordinate system (e.g. allowing the use of symmetries), without enforcing this rotation for the vehicle. Using its properties and a rotating coordinate system, the system description is time-invariant on circular trajectories.

This allows the straightforward application of well-established state feedback design principles. Controllers for standstill and circular motion have been validated experimentally and are shown to stabilize the pendulum. This key milestone allows us to shift our focus towards improving system performance.

Experimental results revealed systematic errors when applying the control laws. There appear to be different sources of these errors:

- Miscalibrations of sensors cause biases in the experimental setup. These errors are observed in the attitude information from the motion capture system, and in the vehicle on-board control loops using gyroscope feedback.
- The simplifying assumption that the pendulum is mounted at the center of mass of the vehicle is violated in the experimental setup. Rotations of the vehicle therefore cause a motion of the pendulum base point.
- The equations of motion used to derive nominal trajectories and linear models neglect many real-world effects, such as drag and underlying dynamics of the control inputs.
- The control laws are designed assuming continuous-time control, while the vehicle is controlled at only 50Hz.

We have identified two approaches for extending the controller design presented in this paper. The first, and most straightforward approach, is to include states that represent the integrated errors, and to weigh them appropriately in the controller design. This would permit compensation for some of the systematic errors. For instance, one would expect this to drive the vehicle position errors in the standstill case to zero.

Alternatively, a machine learning approach could be applied. The measurement data indicates that systematic errors greatly dominate stochastic errors. During the circular trajectory in particular, there are systematic, repeated errors that could well be learned and compensated for in a feed-forward fashion. The system could therefore 'learn' better nominal trajectories, resulting in a correction of the nominal control inputs. This could, for instance, be accomplished with iterative learning control [10], [2]. The present problem is especially well suited to this type of approach due to its repetitive nature. We are planning to use this experimental setup as a testbed and benchmark for learning methods.

The concept of the virtual body frame is applicable to a wide range of quadrotor control problems that goes beyond balancing a pendulum. It allows the time-invariant description of general circular trajectories if the circle size and rate are constant. We are investigating extensions of this concept to allow its application to more general problems.

References

[1] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Volume 2*. Athena Scientific, 2007.

[2] Insik Chin, S. Joe Qin, Kwang S. Lee, and Moonki Cho. A Two-Stage Iterative Learning Control Technique Combined with Real-Time Feedback for Independent Disturbance Rejection. *Automatica*, 40(11):1913–1922, 2004.

[3] JH Gillula, Haomiao Huang, MP Vitus, and CJ Tomlin. Design of Guaranteed Safe Maneuvers Using Reachable Sets: Autonomous Quadrotor Aerobatics in Theory and Practice. *International Conference on Robotics and Automation*, 2010.

[4] Daniel Gurdan, Jan Stumpf, Michael Achtelik, Klaus-Michael Doth, Gerd Hirzinger, and Daniela Rus. Energy-Efficient Autonomous Four-Rotor Flying Robot Controlled at 1 kHz. In *International Conference on Robotics and Automation*, 2007.

[5] Jonathan P. How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.

[6] Michail G. Lagoudakis and Ronald Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4(6):1107–1149, January 2003.

[7] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips. In *International Conference on Robotics and Automation*, 2010.

[8] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010.

[9] Francis C Moon. *Applied Dynamics: With Applications to Multibody and Mechatronic Systems.* Wiley, 1998.

[10] Angela Schöllig and Raffaello D'Andrea. Optimization-Based Iterative Learning Control for Trajectory Tracking. In *European Control Conference*, 2009.

[11] Jinglai Shen, Amit K. Sanyal, Nalin A. Chaturvedi, Dennis Bernstein, and Harris McClamroch. Dynamics and Control of a 3D Pendulum. In *Conference on Decision and Control*, 2004.

[12] Gilbert Strang. *Linear Algebra and its Applications*. Thomson Brooks/Cole, 2006.

[13] H.O. Wang, K. Tanaka, and M.F. Griffin. An Approach to Fuzzy Control of Nonlinear Systems: Stability and Design Issues. *IEEE Transactions on Fuzzy Systems*, 4(1):14–23, 1996.

[14] Victor Williams and Kiyotoshi Matsuoka. Learning to Balance the Inverted Pendulum Using Neural Networks. In *International Joint Conference on Neural Networks*, 1991.

Curriculum Vitae

Markus Hehn

born 14th March 1985

2009 – 2014	*ETH Zurich, Switzerland* Doctoral studies at the Institute for Dynamic Systems and Control (advisor: Prof. Raffaello D'Andrea).
2003 – 2009	*TU Darmstadt, Germany* Undergraduate and graduate studies in mechanical engineering (focus: mechatronics); graduated with Diplom-Ingenieur.
2009	*ETH Zurich, Switzerland* Master thesis at the Institute for Dynamic Systems and Control (advisor: Prof. Lino Guzzella).
2007 – 2008	*Robert Bosch GmbH, Germany* Project student in the hybrid electric vehicle system engineering group.
2006	*Mercedes-Benz High Performance Engines Ltd, United Kingdom* Internship in the Formula 1 engine performance development group.
2003	*BMW AG, Germany* Internship in the apprentice workshop.
1992 – 2003	*European School Brussels I, Belgium* School education; graduated with European Baccalaureate.

Peer-Reviewed Publications

Journals

- Michael Benz, Markus Hehn, Christopher H. Onder, and Lino Guzzella. Model-Based Actuator Trajectories Optimization for a Diesel Engine Using a Direct Method. Journal of Engineering for Gas Turbines and Power, Volume 133, Issue 3, 2011.
- Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control. Autonomous Robots, Volume 33, Issue 1-2, pp 69-88, 2012.
- Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A Platform for Aerial Robotics Research and Demonstration: The Flying Machine Arena. Mechatronics, Volume 24, Issue 1, pp 41-54, 2014.

Refereed Conference Proceedings

- Markus Hehn and Raffaello D'Andrea. A Flying Inverted Pendulum. Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011.
- Sergei Lupashin, Angela P. Schoellig, Markus Hehn, and Raffaello D'Andrea. The Flying Machine Arena as of 2010. Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA) - Video Submission, 2011.
- Angela Schoellig, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Feasibility of Motion Primitives for Choreographed Quadrocopter Flight. Proceedings of the 2011 American Control Conference (ACC), 2011.
- Markus Hehn and Raffaello D'Andrea. Quadrocopter Trajectory Generation and Control. Proceedings of the 2011 IFAC World Congress, 2011.
- Stefania Tonetti, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Distributed Control of Antenna Array with Formation of UAVs. Proceedings of the 2011 IFAC World Congress, 2011.
- Robin Ritz, Markus Hehn, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter Performance Benchmarking Using Optimal Control. Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.
- Markus Hehn and Raffaello D'Andrea. Real-Time Trajectory Generation for Interception Maneuvers with Quadrocopters. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
- Robin Ritz, Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. Cooperative Quadrocopter Ball Throwing and Catching. Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
- Markus Hehn and Raffaello D'Andrea. An Iterative Learning Scheme for High Performance, Periodic Quadrocopter Trajectories. Proceedings of the 2013 European Control Conference (ECC), 2013.

- Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Quadrocopter Pole Acrobatics. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.
- Markus Hehn and Raffaello D'Andrea. A Frequency Domain Iterative Feed-Forward Learning Scheme for High-Performance Periodic Quadrocopter Maneuvers. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.
- Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. A Computationally Efficient Algorithm for State-to-State Quadrocopter Trajectory Generation and Feasibility Verification. Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.