Diss. ETH No. 20593

# Improving Tracking Performance by Learning from Past Data

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by

ANGELA P. SCHOELLIG
Diplom-Ingenieurin (Technische Kybernetik), Universität Stuttgart
MSc in Engineering Science and Mechanics, Georgia Institute of Technology, Atlanta

born July 17, 1983
citizen of Germany

accepted on the recommendation of

Prof. Dr. Raffaello D'Andrea, examiner
Prof. Dr. Andrew Alleyne, co-examiner

2012

Institute for Dynamic Systems and Control
ETH Zurich
Switzerland

# Abstract

The main theme of this thesis is the development of machine learning algorithms for high-performance trajectory tracking. We consider dynamic systems that are required to precisely follow predefined trajectories. The goal of this research is to explore how past data (for example measurements from previous executions) can be used to improve a system's tracking performance.

A typical means of imposing a desired behavior on a dynamic system is feedback control. In such a setup, the motion of the system is guided by an external reference signal and the influence of noise and unexpected disturbances is reduced by feeding back the measured system output. The design of feedback control systems is often based on a mathematical model of the underlying system. The performance of such control schemes is limited by the accuracy of the dynamics model and the causality of the control action that is compensating for disturbances only as they occur.

We address these limitations by proposing a data-based control approach that is able to store and interpret information from past experiments, and infer the correct control actions for future performances. This research is motivated by recent computational advances, which provide enormous possibilities for storing, processing and evaluating large amounts of data. We aim to exploit these new possibilities with three main contributions:

First, we present an algorithm that exploits data from a repeated operation in order to learn to precisely follow a predefined trajectory. We adapt the feed-forward reference signal to the system with the goal of achieving high tracking performance – even under the presence of model errors and other recurring disturbances. The approach is based on a coarse model of the system dynamics and uses measurements from past executions to optimize the tracking performance. We combine traditional optimal filtering methods with state-of-the-art optimization techniques in order to obtain an effective and computationally efficient learning strategy. The proposed approach falls into the area of iterative learning control. Novel features of our approach are the direct treatment of input and state constraints when updating the feed-forward reference, an identification routine that extracts the required system model from a numerical simulation, and a termination condition that stops an execution early if the deviation from the nominal trajectory exceeds a given bound. The latter allows for a safe learning that gradually extends the time horizon of the trajectory. These new features are particularly relevant when we apply the algorithm to highly maneuverable quadrotor vehicles in the ETH Flying Machine Arena. We aim to exploit their full dynamic potential and to improve on time-optimized trajectories. The learning scheme has proven to be effective both when directly learning the thrust and rotational rate inputs sent to the quadrocopter, and when building the learning scheme on

top of a quadrocopter system that is guided by a trajectory following controller. The numerical identification routine was used in the latter case to avoid extensive analytical modeling.

For the second project, we consider iterative learning control in a multi-agent framework, wherein a group of agents simultaneously and repeatedly perform the same task. Assuming similarity between the agents, we investigate whether exchanging information between the agents improves an individual's learning performance. That is, does an individual agent benefit from the experience of the other agents? We derive analytical bounds on the performance improvement due to joint learning.

The third project uses learning in a different context. Here, we aim to precisely track periodic trajectories with a quadrocopter. We develop a learning scheme that determines feedforward correction parameters for a large class of periodic motions from a small set of identification experiments. This research is motivated by our vision of designing and executing multi-vehicle flight that is coordinated to music.

In addition to these three main results, we studied the dynamic limits of quadrotor vehicles and developed algorithms for both generating feasible trajectories, and checking the feasibility of a given trajectory. The first is important for the iterative learning project, where the goal is to perform and improve on highly agile motions. We must therefore design trajectories that are dynamically challenging but still feasible. The latter is important for the design of rhythmic performances, where we wish to check the feasibility of a choreography prior to actual flight.

Aerial robots and, in particular, quadrocopters are a great platform for showcasing the algorithms we have developed to both scientific and non-scientific audiences. Several demonstrations were developed with the goal of visually communicating the key concepts of this work to a large audience.

# Zusammenfassung

Im Mittelpunkt dieser Dissertation steht die Entwicklung von Algorithmen für maschinelles Lernen. Das Ziel der Algorithmen ist es, hochgenaue Trajektorienfolge zu erreichen. Wir betrachten dynamische Systeme, die präzise einer vorgegebenen Trajektorie folgen sollen, und untersuchen, inwiefern Daten aus vorherigen Experimenten (zum Beispiel Messwerte von früheren Durchführungen eines Experiments) verwendet werden können, um die Folgegenauigkeit eines Systems zu verbessern.

Eine klassische Methode, um ein gewünschtes Systemverhalten zu erreichen, ist die Regelung mit Hilfe einer Ausgangsrückführung. Dabei wird das dynamische Verhalten eines Systems durch ein externes Stellsignal gesteuert. Der Einfluss von Rauschen und unerwarteten Störungen wird durch die Rückführung des Systemausgangs reduziert. Dieser Regelkreis wird oftmals basierend auf einem mathematischen Modell des zugrunde liegenden Systems entworfen. Die Güte einer solchen Regelstrategie ist beschränkt durch die Genauigkeit des dynamischen Modells und durch die Kausalität des Regelvorgangs, der auf Störungen erst reagiert, sobald sie auftreten.

Wir stellen einen datenbasierten Regelansatz vor, der die genannten Probleme aufgreift und löst. Informationen von vorhergegangenen Experimenten werden gespeichert und gedeutet. Aus diesen Daten werden die richtigen Regelstrategien für zukünftige Experimente abgeleitet. Jüngste technologische Fortschritte, die es ermöglicht haben, große Datenmengen zu speichern, zu verarbeiten und auszuwerten, machen einen solchen Ansatz möglich. Unser Ziel ist es, diese neuen technologischen Möglichkeiten auszuschöpfen. In dieser Dissertation machen wir dazu drei wesentliche Beiträge:

Das erste Ergebnis ist ein Lernalgorithmus, der Messdaten von dem wiederholten Durchführen eines Experiments nutzt, um zu lernen präzise einer vorgegebenen Trajektorie zu folgen. Nach jedem Durchgang wird das Steuersignal (das heißt, das externe Stellsignal; auch Führungsgröße genannt) angepasst, mit dem Ziel hohe Folgegenauigkeit trotz Modellfehlern und anderen wiederkehrenden Störungen zu erreichen. Der Ansatz basiert auf einem einfachen mathematischen Modell des dynamischen Systems und verwendet Messdaten von früheren Durchgängen, um die Folgegenauigkeit zu verbessern. Der Algorithmus verbindet klassische Methoden der optimalen Filterung mit modernsten Optimierungsverfahren. Das Ergebnis ist eine effektive und recheneffiziente Lernstrategie. Der entwickelte Lernalgorithmus fällt in die Kategorie der Iterativ-Lernenden-Regelung (engl.: Iterative Learning Control). Wesentliche Neuerungen unseres Ansatzes sind die direkte Berücksichtigung von Eingangs- und Zustandsgrößen-Beschränkungen bei der Berechnung des neuen Steuersignals, eine Identifikationsroutine, die das benötigte Systemmodell aus einer numerischen Simulation ableitet, und

eine Abbruchbedingung, die einen Durchgang frühzeitig stoppt, falls die Abweichung von der nominellen Trajektorie eine gegebene Grenze übersteigt. Letzteres ermöglicht ein gefahrloses Lernen, bei dem der Zeithorizont der Trajektorie schrittweise verlängert wird. Die Besonderheiten des neuen Algorithmus spielen insbesondere für unsere Anwendung eine wichtige Rolle. Wir evaluieren den Ansatz auf kleinen, wendigen Quadrokoptern in der ETH Flying Machine Arena. Unser Ziel ist dabei das dynamische Potenzial der Quadrokopter auszunutzen und zeitoptimierte Trajektorien zu lernen. Das Lernverfahren hat sich als erfolgreich erwiesen, sowohl wenn die Eingangsgrößen des Quadrokopters (Gesamt-Schub und Drehraten) direkt gelernt werden, als auch wenn der Lernalgorithmus auf ein Quadrokopter-System angewendet wird, das von einem Trajektorien-Folgeregler gesteuert wird. Für Letzteres wurde die erwähnte numerische Identifikationsroutine verwendet, um das aufwändige Herleiten eines analytischen Modells zu umgehen.

Das zweite Projekt beschäftigt sich mit der Iterativ-Lernenden-Regelung für Multiagenten-Systeme. Genauer gesagt betrachten wir eine Gruppe von Agenten, die gleichzeitig und wiederholt die gleiche Aufgabe ausführen. Unter der Annahme, dass eine Ähnlichkeit zwischen den Agenten besteht, untersuchen wir, ob der Austausch von Information zwischen den Agenten den Lernerfolg eines einzelnen Agenten erhöht. Das heißt: Profitiert ein einzelner Agent von den Erfahrungen der anderen Agenten? Wir leiten eine analytische Schranke her, die die Verbesserung des Lernerfolgs durch ein gemeinschaftliches Lernen nach oben beschränkt.

Das dritte Projekt betrachtet Lernen in einem anderen Zusammenhang. Das Ziel ist hier, dass Quadrokopter periodischen Trajektorien präzise folgen. Wir entwickeln einen Lernalgorithmus, der Steuersignal-Korrekturparameter für eine große Klasse periodischer Bewegungen mithilfe einer sehr begrenzten Anzahl von Identifikationsexperimenten bestimmt. Motivation für diese Arbeit ist unsere Vision, Flugdarbietungen für mehrere Quadrokopter abgestimmt zur Musik zu entwerfen und auszuführen.

Neben diesen drei Hauptresultaten, haben wir insbesondere die dynamischen Grenzen von Quadrokoptern untersucht und Algorithmen entwickelt, die (für Quadrokopter) ausführbare Trajektorien generieren und die die Ausführbarkeit einer gegebenen Trajektorie testen. Ersteres ist wichtig für das iterative Lernen. Hier ist unser Ziel außerordentlich schnelle Bewegungen zu lernen. Wir müssen daher Trajektorien generieren, die dynamisch anspruchsvoll, aber dennoch ausführbar sind. Letzteres is wichtig für das Design von rhythmischen Flug-Darbietungen. Hier ist es entscheidend, die Ausführbarkeit einer Choreographie vor dem eigentlichen Flug zu testen.

Flugroboter und insbesondere Quadrokopter sind eine großartige Plattform, um die Algorithmen, die wir entwickelt haben, sowohl einem wissenschaftlichen Publikum als auch der breiten Öffentlichkeit zu präsentieren. Verschiedene Vorführungen wurden entwickelt, mit dem Ziel visuell die grundlegenden Konzepte dieser Arbeit einem großen Publikum zu vermitteln.

# Acknowledgements

My deepest gratitude goes to my advisor, Raffaello D'Andrea. His thinking was inspiring, his trust encouraging, his strive for excellence challenging, his excitement contagious, his feedback motivating, and his support invaluable. He created a phenomenal research environment with outstanding people and truly unique opportunities. Over the past four years I have learned so much from him.

I would like to extend my gratitude to Andrew Alleyne not only for being my co-examiner but also for providing strong support beyond my PhD.

My PhD has been an exciting journey, where every day was different, where there were always more ideas than time, and which was shaped by incredible people. I owe much to Sebastian Trimpe for being a great friend, colleague and office mate; I enjoyed his honest and thoughtful comments, his humor, his reliability, and his organizational skills – especially when working together as teaching assistants. I am thankful for having worked closely with Sergei Lupashin: his intuitive and straight-forward approach to problems and his unconventional views on things taught me a lot. I thank Philipp Reist for his enthusiasm, the organization of many team events, and his great help with understanding the ETH system and the Swiss in general. I learned a great deal by performing experiments in the Flying Machine Arena and working together with an amazing team of people: Markus Hehn, Mark Mueller, Robin Ritz and Federico Augugliaro. It has always been a great pleasure to work with them. I would like to thank Markus Waibel for his advice and his support with disseminating our research results. I am grateful for inspiring discussions and many unforgettable moments with my lab mates, Raymond Oung, Mohanarajah Gajamohan, Nico Hübel, Max Kriegleder, and past group members, Geo Robson, Michael Sherback, Guillaume Ducard, Felix Althaus and Frédéric Bourgault.

I am greatly indebted to the students who worked with me and contributed to the results of this thesis; in particular, I would like to thank Federico Augugliaro, Fabian Mueller, Javier Alonso-Mora and Clemens Wiltsche who continued their research beyond their thesis or semester project and co-authored papers with me.

Many thanks also to the technical staff, Igor Thommen, Marc-Andre Corzillius, Hans Ulrich Honegger, and Matthew Donovan, and the communication and design experts, Hallie Siegel and Carolina Flores. Their contributions greatly improved the quality and extended the outreach of my research results. I thank the office team, Katharina Munz, Aleksandra Vukovic, Annina Fattor, Claudia Wittwer and Brigitte Rohrbach, for making the institute's administration as simple as possible for us.

ETH is an exceptional place for doing research. My special thanks go to Profs. Lino Guzzella, Roland Siegwart, Manfred Morari, John Lygeros and Robert Riener. Their courses

and the interactions with their teams have been a great extension to my research experience.

I have had the great opportunity to spend four weeks at Lund University during the focus period "Adaptation and Learning in Autonomous Systems". I would like to thank Prof. Rantzer for the organization and financial support. Special thanks go to Leif Andersson from Lund University who provided the basic LaTeX template for this thesis.

I would not be where I am now without the continuous support and trust of my former advisors and academic colleagues, Prof. Frank Allgöwer, Prof. Magnus Egerstedt, Dr. Peter Gath, Prof. Laurence J. Jacobs and Prof. Bozenna Pasik-Duncan.

My sincerest gratitude goes to my family: to my parents for supporting me as much as possible and for encouraging me to go my own way, my brother and sister for always being there, and my big family clan for inspiring me in many different ways.

And finally, I thank my partner Christian Menkens for his unconditional support. He has been my rock throughout this amazing, and at times crazy journey.

Zurich, Summer 2012 *Angela Schoellig*

## Financial Support

Grateful acknowledgements go to the Swiss National Science Foundation (SNSF), which funded this research in part under the grant "High-performance maneuvers and trajectory generation for quadrotor flying vehicles", under the equipment grant "Optical motion capture system for robot experiments in real world environments", and through the National Centre of Competence in Research Robotics.

# Contents

# Preface

This section provides a brief overview of the thesis structure. The thesis consists of four introductory chapters followed by eight self-contained publications. The introductory chapters highlight the ideas and concepts that stand behind this thesis. We put the introductory chapters at the beginning of the thesis to establish a connection between the individual papers and to summarize their main results.

Chapter 1 sketches the potential of learning-based control and serves as motivation of this work. It also defines the scope of this research in terms of the type of problems we consider and the kind of learning approach we take. We focus on feed-forward adaptation schemes for improving the performance of control systems. Two different projects were conducted as part of this thesis: iterative learning for precise trajectory tracking (Part A), and rhythmic multi-vehicle flight performances based on periodic motion primitives (Part B). Chapter 2 describes the key contribution of both projects and explains the relationship between the individual results. We also refer to related papers that are not included in this thesis, and to videos illustrating the experimental results. The algorithms developed in this thesis are demonstrated on small flying robots, so-called quadrocopters, operated in the ETH Flying Machine Arena (FMA), an indoor flight test facility. Chapter 3 contains a short description of the experimental setup. In Chapter 4 we reflect on potential future directions of the presented research.

Subsequent to the preliminary chapters, the individual papers are included. They are sorted by project. The papers are peer-reviewed, published conference and journal contributions.

# 1

# Introduction

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*
>
> – Definition of 'Machine Learning' from [1].

This thesis presents machine learning algorithms for high-performance, high-precision trajectory tracking. Driven by recent computational advances for storing, processing and evaluating large amounts of data, the goal of this research is to improve the tracking performance of dynamic systems based on measurements collected during previous experiments.

Specifically, we consider dynamic systems that precisely follow predefined trajectories, and present a computer program that is capable of controlling one or more of these systems. We find that experience (in the form of measurements of the systems' actual motion) obtained from performing one trajectory may be used to improve this specific trajectory, or to improve an entire set of trajectories. We also find that the performance measure is usually related to the deviation of the actual motion from the desired trajectories.

Below we state the research objectives in more detail and define the scope of the work in terms of the problems we consider, the approach we use, and the results we aim for in this work.

## 1.1 Goal and Motivation

To impose a desired behavior on a dynamic system, a feedback-control setup such as the one shown in Fig. 1.1 is typically implemented. A controller manipulates the system input to obtain the desired effect on the system output. The actions of the controller are based on the measured system output and an external reference signal. Feeding back the measured output enables the control system to correct for noise and unexpected disturbances. The behavior of the closed-loop system can be influenced by (i) designing an appropriate controller and/or (ii) choosing a suitable reference signal. The controller design is often based on a mathematical model of the underlying system dynamics, and the reference signal may be found by an (approximate) inversion of the closed-loop dynamics model. The performance of such closed-loop systems is thus limited by the accuracy of the dynamics model and by the causality of the control action that is compensating for disturbances only as they occur. Unfavorable effects of these limitations are observed especially in regimes where feedback is not able to react in time and

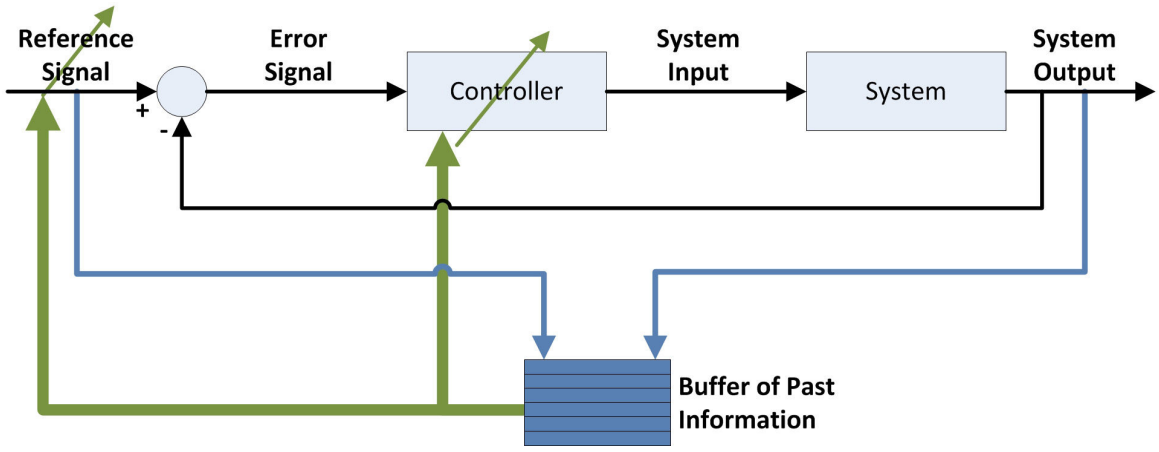



**Figure 1.1** Schematics of a typical control system in black. Past experience (depicted in dark blue) can be incorporated by adapting the controller and/or the reference input as shown by the green arrows.

where the dynamic behavior is difficult to identify or understand.

To achieve high performance in such cases, we propose a data-based control approach that is able to store and interpret information from past executions, and to infer the correct actions or control laws for future experiments. Information from past experiments can be used to (i) adapt the feedback controller of the system, or to (ii) change the feed-forward reference signal, see Fig. 1.1. In (i), the system learns rules on how to 'react' better, while in (ii), the system learns how to 'act' in the first place. These two approaches may also be combined.

In this thesis, we focus on case (ii) and develop algorithms that adapt the feed-forward reference signal based on past data. Such an approach outperforms pure feedback control, since it is not limited to a causal action. Instead, recurring disturbances are anticipated and proactively compensated for before they occur. Recurring disturbances are often caused by modeling errors or nonidealities in the system dynamics. That is, such an adaptation scheme is particularly beneficial when the nominal model differs from the actual dynamic behavior of the system.

Looking at the problem from a different perspective, the goal is to develop algorithms that enable autonomous systems (for example, robots) to get better at a given task through practice. A useful analogy for understanding this approach is an orchestral performance. Musicians would never go on stage without first practicing the piece of music that is to be performed. Over many iterations, they learn the selected piece of music, repeating difficult passages to refine the movement of their fingers or tongues; they *learn how to perform*. If *a priori* practice benefits musicians, would it not stand to reason that practice might benefit robots as well?

In this thesis, two different research projects were carried out, each demonstrating the benefits of *a priori* practice: iterative learning for precise trajectory tracking (Part A), and rhythmic flight performances based on periodic motion primitives (Part B). Both research projects led to theoretic and experimental contributions. The resulting algorithms were demonstrated on quadrotor vehicles operated in the Flying Machine Arena (FMA), an indoor flight test facility (see Sec. 3 for more information). Below we briefly introduce both projects and highlight the benefits of a feed-forward learning scheme.

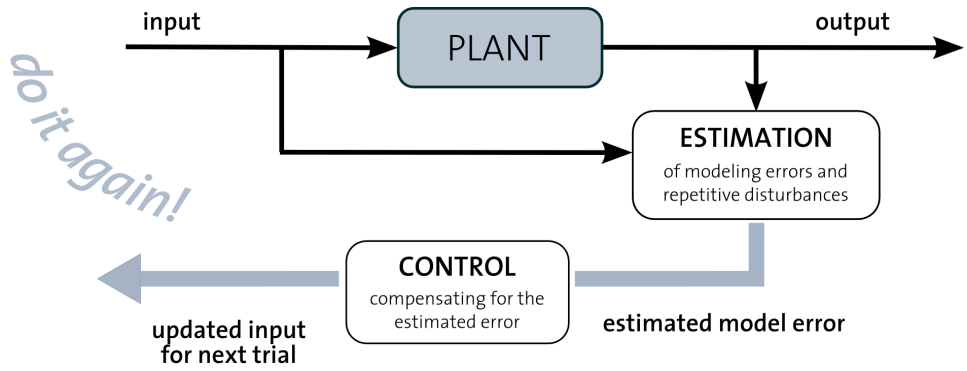



**Figure 1.2** General iterative learning framework.

### Part A. Iterative Learning

In this project, we focus on learning a specific trajectory from iterative experiments. The tracking performance is improved from trial to trial by exploiting the experience gained from previous repetitions. As illustrated in Fig. 1.2, the information of past executions is used to identify recurring disturbances. To compensate for the identified disturbances, we adapt the feed-forward reference input after each trial. The proposed learning scheme is based on a rough dynamics model of the system and, thus, applies to any system for which a nominal model is available.

In the context of the definition provided at the beginning of this chapter, the class of tasks T we improve on contains only one particular trajectory. We gain experience in the form of measurements taken while executing this specific motion. Different (nonlinear) performance objectives influencing the overall learning behavior can be specified. The result of the learning is an optimized reference input, which is a discrete-time signal of finite length.

We applied this learning algorithm to quadrotor vehicles with the goal of performing aggressive, (near) time-optimal maneuvers. Quadrotor vehicles offer exceptional agility in the translational and rotational degrees of freedom. When operating these vehicles at high speeds, complex dynamic effects such as aerodynamics, battery behavior, and motor dynamics have a significant impact on the vehicle behavior. These effects are difficult to model but can be compensated for through iterative execution. Since the correcting action is executed only after a complete run of the trajectory, the approach is not restricted by slow feedback rates or system latencies. Such a feed-forward adaptation scheme is thus particularly suited for fast motions.

Motivated by the fleet of quadrocopters in the FMA, we began to investigate iterative learning in a multi-agent framework.While most of the previous work in multi-agent iterative learning is on cooperative or coordinated learning schemes [2–5], we study a novel question: Is the learning curve of an individual agent steeper when taking into account the experience of the other agents? Is there a benefit to exchanging information between the agents? We focus on the potential for individual agents to improve their performance when conducting a task alongside a group of similar agents conducting the same task. We derive theoretic results that show that information exchange between agents engaged in learning the same task provides only very limited benefit for an individual's learning performance. In this analysis, we as-

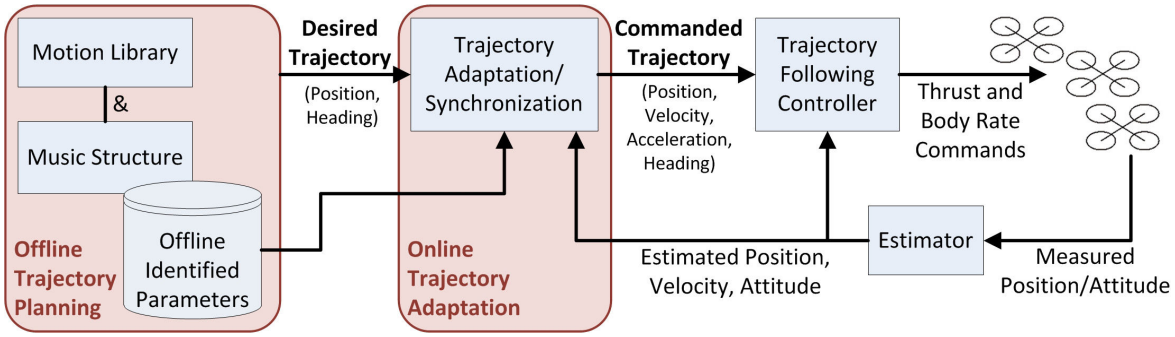



**Figure 1.3** High-level control architecture used for implementing rhythmic flight performances. Key components are the offline trajectory planning and the online trajectory adaptation. (Position, velocity and acceleration refer to the translational coordinates and heading corresponds to the vehicle yaw.)

sumed that agents are similar but not identical. This assumption was strongly motivated by our quadrocopters, which differ slightly in their parameters.

### Part B. Rhythmic Flight Performances

The second project is driven by the vision of creating a novel visual musical experience: a dance performance featuring multiple flying vehicles that move in synchrony with the music, and perform flips, eights, circles and other aggressive maneuvers timed with the beat of the music. Connecting vehicle motion to music is key to a rhythmic flight performance, as it is this link that transforms movement into dance. Periodic motion primitives are the basic building blocks of such choreographies, and their synchronization to the music beat is essential for a convincing dance performance. We develop a control strategy that builds upon the same basic trajectory-tracking controller for periodic motions at all frequencies, but adapts the parameters of the actual input to the controller in order to guarantee precise tracking and synchronization, see Fig. 1.3 for an illustration. These parameters are identified prior to the flight performance to effectively reduce transient time and tracking errors. One experiment per translational direction and music frequency is enough to identify the feed-forward correction parameters valid for all periodic motions (with varying amplitudes) in this direction at this given frequency. Parameters are–for each translational direction–an amplitude amplification factor and a phase shift. That is, given the music's frequency, we perform and learn from one periodic motion at the given frequency for all other periodic movements at this frequency.

In the context of the definition provided above, experience is collected by performing one (arbitrary) three-dimensional periodic motion at frequencies chosen from a set of relevant music frequencies. The class of tasks we improve on is the class of all periodic motion primitives at these frequencies, where the performance objective is defined in terms of phase lag and amplitude amplification between quadrocopter motion and the desired nominal periodic motion. The result of the learning is an optimized set of motion parameters, which shape the feed-forward input sent to the vehicle controller.

When recalling the objective and the inherent requirements of the project, the benefits of adapting the feed-forward reference signal (instead of the vehicle controller) become obvious. During a performance, we expect the quadrocopter to accurately track the selected sequence

of periodic motions, without incurring large transients at the beginning of each motion. When applying pure feedback control, precise temporal and spatial tracking (for motions of varying angular frequencies) are only achievable when changing the controller depending on the motion to be performed. For choreographies in which motions are changed in quick succession, designing separate controllers for different motions is impractical. Switching between different controllers may even cause instability. Moreover, the causality of the control action, imperfect initial conditions and model errors effect an initial transient phase, in which the tracking errors are typically substantial. A feed-forward scheme allows for smooth transitions between periodic motions, reduces transients, and guarantees stability.

## 1.2 Scope of Research Work

This section describes the main focus and some key aspects of the proposed approach.

**Model-based learning** Fundamental to the learning approach taken herein is to incorporate prior knowledge about the system into the learning scheme. The algorithms are based on a dynamics model that captures the key behavior of the underlying system. The model may be derived from first principles or may be obtained from a numerical simulation of the system behavior. We use the model to tell us how to correct for errors, while the collected data allows us to estimate the unknown effects resulting from unmodeled dynamics and disturbances. Note that the model is used for both interpreting the collected data and for updating the feed-forward reference signal. By leveraging the *a priori* knowledge about the system's dominating dynamics, we are able to increase the efficiency of the learning and the speed of convergence. In Part A, we rely on a nominal model that describes the mapping from input deviations to output deviations. In Part B, the algorithm builds upon the assumption that the quadrocopter's closed-loop dynamics can be approximated by three directionally decoupled linear systems.

**Feed-forward adaptation** We incorporate information gained from past experiments by adapting the feed-forward reference signal of the system. As highlighted in Sec. 1.1, this approach has several advantages over pure feedback control for the problems considered in this thesis.

**Task-based learning objective** The developed learning schemes are aimed at improving the tracking performance of selected tasks. Generalizing this knowledge and making it applicable to new tasks is not considered.

**Experience through a repeated operation** In Part A, a selected task is performed repeatedly and thus recurring disturbances are iteratively identified. In Part B, we perform a selected periodic motion over a longer time horizon to obtain the steady-state parameter correction values.

**Motion feasibility** In this work, the intrinsic motivation for applying learning schemes is the desire to perform highly agile motions that exploit the full dynamic potential of a given system. In order to design trajectories that are dynamically challenging but still feasible, we study the dynamic limits of the systems under consideration (in particular, of

quadrotor vehicles) and develop algorithms for both generating feasible trajectories, and checking the feasibility of given trajectories.

**Computationally tractable algorithms** We aim for learning algorithms that run on normal desktop PCs and that require not more than a few minutes to compute a learning update. Nevertheless, we leverage computational advances and, for example, build upon state-of-the-art optimization techniques.

**Experimental validation** As part of this research project, algorithms are tested on small aerial vehicles, commonly referred to as quadrocopters, see Sec. 3. These vehicles not only serve as demonstrator for the developed control and learning algorithms but also provide insight into the characteristics of these algorithms and, more important, provide ideas for further investigations and improvements.

**Outreach** Aerial robots and, in particular, quadrocopters are a great platform for showcasing the algorithms that we have developed to both scientific and non-scientific audiences. They are a means to 'visually' communicate key concepts of controls and learning to a larger audience. The FMA hosts hundreds of visitors a year, and the work has been featured at two large exhibitions and in numerous science TV shows. It has also been extensively covered in radio, print, and online media (e.g., the Youtube channel of `www.FlyingMachineArena.org` has received more than 4'500'000 views). These public events and appearances are a great motivation for developing demonstrations that operate reliably and are comprehensible to the general public. A list of demonstrations that were created in the context of this work is found in Sec. 2.4.

**Theoretical analysis** Experimental validation is complemented by theoretical analysis to derive analytical performance bounds of systems and algorithms. We obtained, for example, an upper bound for an individual's performance improvement when sharing information with peers during the learning process. This bound told us that there is not a significant benefit of having multiple agents learn the same task together.

# 2

# Contributions

This chapter briefly summarizes the key contributions of the papers. We explain the relationship between the individual results and refer to related video contributions. It should be emphasized that the results were obtained in close collaboration with colleagues and master students, as indicated below. At the end of this chapter, we provide a complete list of peer-reviewed publications and video contributions that resulted from the author's PhD studies.

## 2.1 Part A. Iterative Learning

The papers listed below describe algorithms that fall in the area of iterative learning control (ILC) [6]. The objective is to improve tracking performance through a repeated operation. We consider single- and multi-vehicle learning problems, in theory and practice.

### Paper I

[P1] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2012): "Optimization-based iterative learning for precise quadrocopter trajectory tracking." *Autonomous Robots*, **33:1**, pp. 103–127.

***Context*** We developed a new ILC algorithm that solves–at its core–a constrained convex optimization problem [7]. The result of the optimization is an updated reference signal that is applied during the next trial. This paper presents the details of the newly developed ILC algorithm. It also contains the first experimental results of applying this algorithm to quadrotor vehicles. In these first experiments, we do not close the loop on desired vehicle position and attitude but instead adapt the inputs to the quadrocopter; specifically, three rotational rate commands and the net force of the four rotors. In view of Fig. 2.1, the learning is directly applied to the thrust and body rate commands, leaving out the trajectory following controller.

***Contribution*** The contribution of this paper is an algorithm that structures the trajectory learning problem around a disturbance estimation and input update step. We use a Kalman filter for estimating the additive disturbance along the trajectory. This allows for a direct incorporation of process and measurement noise characteristics. By formulating the input update step as a convex optimization problem, input and state constraints can be explicitly taken into account–an important aspect when aiming to learn aggressive motions. Both steps rely on a nominal model of the underlying system. We obtain an effective and computationally efficient

learning strategy that makes use of state-of-the-art optimization techniques and software. A novel feature is the termination condition, which stops an execution early if the deviation from the nominal trajectory exceeds a given bound. This allows for a safe learning that gradually extends the time horizon of the trajectory.

We apply this algorithm to quadrotor vehicles and learn (near) time-optimal maneuvers in the vertical plane. As a preliminary step, an algorithm for generating feasible flight trajectories was developed. The theoretic approach, as well as the application of the algorithm to real quadrotor vehicles, makes this work an original contribution to the field.

***Related Publications and Videos*** The core algorithm was previously published in:

[R1] Schoellig, A. P. and R. D'Andrea (2009): "Optimization-based iterative learning control for trajectory tracking." In *Proceedings of the European Control Conference (ECC)*, pp. 1505–1510.

In this paper, the algorithm was applied to the benchmark problem of swinging up a pendulum using open-loop control only.

Videos that demonstrate the learning of both the pendulum up-swing and the quadrocopter maneuvers can be found in:

[V1] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2009): "Swing it up! Iterative learning control on a cart-pendulum system." At `www.tiny.cc/LearnSwingUp`.

[V2] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2011): "Learning to follow a trajectory – quadrocopters improve over time." At `www.tiny.cc/QuadroLearnsTrajectory`.

### Paper II

[P2] Mueller, F. L., A. P. Schoellig, and R. D'Andrea (2012): "Iterative learning of feed-forward corrections for high-performance tracking." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3276–3281.

***Context*** This paper represents a continuation of the work in Paper I. The difference is that we apply the learning algorithm to a quadrotor vehicle that is guided by a trajectory-following controller; that is, we learn reference positions, see Fig. 2.1. The behavior of the closed-loop quadrocopter system is more repeatable than the setup in Paper I. The real-time feedback component reduces the influence of non-repetitive noise while the ILC adjusts to the repetitive disturbance. Due to the higher repeatability, we obtain tracking errors (after learning) that lie in the range of a few centimeters.

***Contribution*** The novelty of this work is a system identification routine that uses a numerical simulation of the system dynamics to extract the required model information. Using the identification routine allows the learning algorithm to be applied to any dynamic system for which a dynamics simulation is available (including systems with underlying feedback loops). The result is a conceptually simple learning routine.

The experimental evaluation of the identification routine combined with the learning algorithm shows fast convergence, typically in less than 10 iterations. The time for identifying the

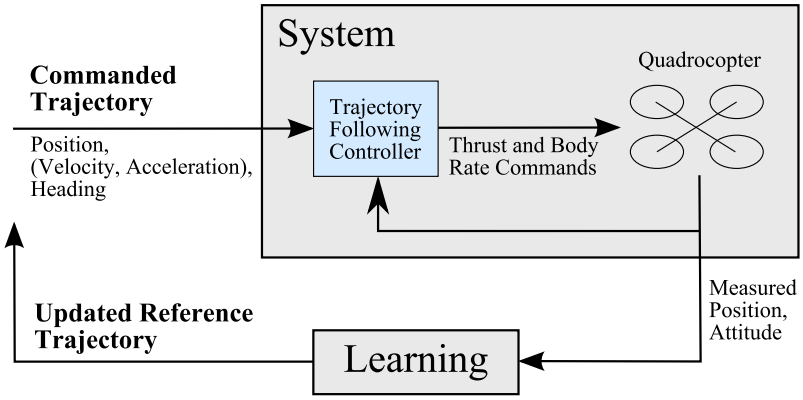



**Figure 2.1** The iterative learning scheme is applied to a quadrocopter that is controlled by a trajectory-following controller. Setup as considered in Paper II.

model via the numerical simulation is in the range of minutes and depends on the length of the trajectory.

***Related Publications and Videos*** The capabilities of the learning algorithm combined with the trajectory generation are demonstrated in the following video:

[V3] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2012): "Quadrocopter Slalom Learning." At `www.tiny.cc/SlalomLearning`.

Slalom poles are arbitrarily placed in the space and a time-optimized trajectory around the sticks is calculated. The quadrocopter executes its learning iterations above the poles. Once the vehicle is guaranteed to pass between the slalom poles, the height of the quadrocopter is lowered and the slalom is performed. In the video, we also show the slalom performance without learning.

### Paper III

[P3] Schoellig, A. P., J. Alonso-Mora, and R. D'Andrea (2012): "Limited benefit of joint estimation in multi-agent iterative learning." *Asian Journal of Control*, **14:3**, pp. 613–623.

***Context*** Motivated by the fleet of quadrocopters in our lab, we began to study iterative learning in a multi-agent framework. We were particularly interested in improving the learning performance of an individual agent when learning a task together with a group of other agents. One question that arose was: does an individual agent benefit from the experience of the other agents when all agents learn the same task simultaneously? Driven by first experimental results, we derived analytical bounds on the improvement due to joint learning.

***Contribution*** Novel to this paper is the type of question we ask, and consequently also the answers we get. The theoretical results derived in this paper are again based on the idea of dividing the learning into an estimation and update step. Thus the question becomes: to what

degree can the disturbance estimate of an individual agent be improved by taking into account the measurements of the other agents? We show that in the ideal case of identical agents and no process noise, instead of one agent performing a task $N$ times, $N$ agents performing the task once results in the same accuracy for the disturbance estimate. However, in a realistic scenario, where agents are similar but not identical and process noise is present, the benefits are minor.

***Related Publications and Videos*** A first, condensed version of the results was published in:

[R2] Schoellig, A. P., J. Alonso-Mora, and R. D'Andrea (2010): "Independent vs. joint estimation in multi-agent iterative learning control." In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 6949–6954.

### Paper IV

[P4] Schoellig, A. P. and R. D'Andrea (2012): 'Sensitivity of joint estimation in multi-agent iterative learning control." In *Proceedings of the IFAC (International Federation of Automatic Control) World Congress*, pp. 1204–1212.

***Context*** We continued the work of Paper III, and investigated a more realistic scenario, where the degree of similarity between the agents is not precisely known.

***Contribution*** The theoretical derivations in this paper underline the fact that the proposed joint learning approach is not robust to nonidealities in the system setup. If, for example, the similarity of agents is overestimated, joint learning performs worse than individual learning.

## 2.2 Part B. Rhythmic Flight Performances

The papers of this part are motivated by the objective of designing and executing multi-vehicle flight that is coordinated to music. The papers each address different aspects of the problem: motion-music synchronization, periodic motion primitives and their feasibility, and overall system setup.

### Paper V

[P5] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "Synchronizing the motion of a quadrocopter to music." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3355–3360.

***Context*** The first step towards a rhythmic quadrocopter performance was to investigate the feasibility of synchronizing the motion of a quadrocopter to music.

***Contribution*** The result of the paper is a proof-of-concept, in which we demonstrate that it is possible to precisely synchronize a periodic side-to-side motion of a quadrocopter to a music beat using concepts from phase-locked loops. When observing that, without additional synchronization and adaptation schemes, the quadrocopter had a constant time shift and amplitude amplification, the idea was born to identify these parameters prior to flight performance. This paper was a very first step towards dancing aerial robots.

***Related Publications and Videos*** The following videos demonstrate the side-to-side motion in action. The music is pre-processed and the desired swing motion is planned based on the beat times obtained.

[V4] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2009): "Synchronizing motion with music – a dancing quadrocopter." At `www.tiny.cc/FirstDance`.

[V5] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2009): "Synchronizing the motion of a quadrocopter to music – ICRA 2010." At `www.tiny.cc/DanceICRA2010`.

### Paper VI

[P6] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "A platform for dance performances with multiple quadrocopters." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) – Workshop on Robots and Musical Expressions*, pp. 1–8.

***Context*** Based on the results of Paper V, we began to think about the important building blocks that would need to be developed for a rhythmic multi-vehicle performance. At the workshop on robots and musical expression, we presented our idea of a rhythmic aerial dance to a community of roboticists who mainly work with humanoid robots or on automatic music feature extraction.

***Contribution*** Our contribution to this workshop was the description of a platform for multi-vehicle quadrocopter dancing. The paper states the results to-date, and highlights prospective features of the platform, related challenges and future steps.

***Related Publications and Videos*** A comprehensive summary of the current capabilities of the platform and of its user interface is given in:

[R3] Augugliaro, F., A.P. Schoellig, and R. D'Andrea (2013): "Dance of the Flying Machines." *IEEE Robotics and Automation Magazine*, to appear.

The first video below shows the swing motion performed by three vehicles at a time. as well as some of our software tools used for development. The latter videos link to our first two full-song-long multi-vehicle dance performances. The swing motion is a recurring element. Other motions are manually implemented.

[V6] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "Towards musical performances with multiple quadrocopters – IROS 2010." At `www.tiny.cc/DanceIROS2010`.

[V7] Schoellig, A. P., F. Augugliaro, S. Lupashin, and R. D'Andrea (2010): "Dance together! (Pirates of the Caribbean)" At `www.tiny.cc/Dance2gether`.

[V8] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "Dance with three (Rise Up)." At `www.tiny.cc/DanceWith3`.

### Paper VII

[P7] Schoellig, A. P., C. Wiltsche, and R. D'Andrea (2012): "Feed-forward parameter identification for precise periodic quadrocopter motions." In *Proceedings of the American Control Conference (ACC)*, pp. 4313–4318.

***Context*** This paper generalizes the results of Paper V. We consider three-dimensional periodic motions and use learning (prior to the actual performance) to achieve high spatial and temporal tracking performance.

***Contribution*** The main contribution of this paper is an identification scheme that tunes parameters for a large class of periodic motions, and requires only a small number of identification experiments prior to flight. This reduced identification is based on analysis and experiments showing that the quadrocopters' closed-loop dynamics can be approximated by three directionally decoupled linear systems. Experiments were performed to validate the reduced identification.

***Related Publications and Videos*** This video shows a dance performance of five vehicles, including various periodic motion primitives.

[V9] Augugliaro, F., R. D'Andrea, A. P. Schoellig, and S. Lupashin (2011): "Dance of the quadrocopters (Armageddon)." At `www.tiny.cc/Armageddon`.

### Paper VIII

[P8] Schoellig, A. P., M. Hehn, S. Lupashin, and R. D'Andrea (2011): "Feasibility of motion primitives for choreographed quadrocopter flight." In *Proceedings of the American Control Conference (ACC)*, pp. 3843–3849.

***Context*** Like humans, whose range of motion and speed of movement is limited, not all motions are feasible for a quadrocopter. This is obvious when considering the horizontal side-to-side motion (see Paper V); a combination of large motion amplitude and large frequency may exceed the capabilities of a quadrocopter. Aiming for an intuitive choreographic design process, we looked for a means of checking the feasibility prior to flight.

***Contribution*** The result of this paper is a validation tool that determines the feasibility of trajectories based on first principles models, and that ensures that desired trajectories respect both vehicle dynamics and motor thrust limits. The predicted feasibility constraints are compared against experimental results from quadrocopters in the FMA.

***Related Publications and Videos*** The feasibility results of Paper VIII were adapted in the following paper on collision-free trajectory generation in order to check the feasibility of the trajectories obtained from the sequential convex programming routine:

[R4] Augugliaro, F., A. P. Schoellig, and R. D'Andrea (2012): "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1917–1922.

## 2.3 List of Publications

Below, we provide a concise list of publications that are featured in this thesis, as well as a list of the author's related papers.

### Publications included in this thesis

[P1] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2012): "Optimization-based iterative learning for precise quadrocopter trajectory tracking." *Autonomous Robots*, **33:1**, pp. 103–127.

[P2] Mueller, F. L., A. P. Schoellig, and R. D'Andrea (2012): "Iterative learning of feed-forward corrections for high-performance tracking." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3276–3281.

[P3] Schoellig, A. P., J. Alonso-Mora, and R. D'Andrea (2012): "Limited benefit of joint estimation in multi-agent iterative learning." *Asian Journal of Control*, **14:3**, pp. 613–623.

[P4] Schoellig, A. P. and R. D'Andrea (2012): 'Sensitivity of joint estimation in multi-agent iterative learning control." In *Proceedings of the IFAC (International Federation of Automatic Control) World Congress*, pp. 1204–1212.

[P5] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2009): "Synchronizing the motion of a quadrocopter to music." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3355–3360.

[P6] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "A platform for dance performances with multiple quadrocopters." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)–Workshop on Robots and Musical Expressions*, pp. 1–8.

[P7] Schoellig, A. P., C. Wiltsche, and R. D'Andrea (2012): "Feed-forward parameter identification for precise periodic quadrocopter motions." In *Proceedings of the American Control Conference (ACC)*, pp. 3843–3849.

[P8] Schoellig, A. P., M. Hehn, S. Lupashin, and R. D'Andrea (2011): "Feasibility of motion primitives for choreographed quadrocopter flight." In *Proceedings of the American Control Conference (ACC)*, pp. 3843–3849.

### Related publications

[R1] Schoellig, A. P. and R. D'Andrea (2009): "Optimization-based iterative learning control for trajectory tracking." In *Proceedings of the European Control Conference (ECC)*, pp. 1505–1510.

[R2] Schoellig, A. P., J. Alonso-Mora, and R. D'Andrea (2010): "Independent vs. joint estimation in multi-agent iterative learning control." In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 6949–6954.

[R3] Augugliaro, F., A.P. Schoellig, and R. D'Andrea (2013): "Dance of the Flying Machines." *IEEE Robotics and Automation Magazine*, to appear.

[R4] Augugliaro, F., A. P. Schoellig, and R. D'Andrea (2012): "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1917–1922.

[R5] Lupashin, S., A. P. Schoellig, M. Sherback, and R. D'Andrea (2010): "A simple learning strategy for high-speed quadrocopter multi-flips." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1642–1648.

[R6] Lupashin, S., A. P. Schoellig, M. Hehn, and R. D'Andrea (2010): "The Flying Machine Arena as of 2010." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)–Video Submission*, pp. 2970–2971.

## 2.4 List of Videos

Several videos of this work were published online. Below is a complete list of video contributions.

[V1] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2009): "Swing it up! Iterative learning control on a cart-pendulum system." At `www.tiny.cc/LearnSwingUp`.

[V2] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2011): "Learning to follow a trajectory – quadrocopters improve over time." At `www.tiny.cc/QuadroLearnsTrajectory`.

[V3] Schoellig, A. P., F. L. Mueller, and R. D'Andrea (2012): "Quadrocopter Slalom Learning." At `www.tiny.cc/SlalomLearning`.

[V4] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2009): "Synchronizing motion with music – a dancing quadrocopter." At `www.tiny.cc/FirstDance`.

[V5] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2009): "Synchronizing the motion of a quadrocopter to music – ICRA 2010." At `www.tiny.cc/DanceICRA2010`.

[V6] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "Towards musical performances with multiple quadrocopters – IROS 2010." At `www.tiny.cc/DanceIROS2010`.

[V7] Schoellig, A. P., F. Augugliaro, S. Lupashin, and R. D'Andrea (2010): "Dance together! (Pirates of the Caribbean)" At `www.tiny.cc/Dance2gether`.

[V8] Schoellig, A. P., F. Augugliaro, and R. D'Andrea (2010): "Dance with three (Rise Up)." At `www.tiny.cc/DanceWith3`.

[V9] Augugliaro, F., R. D'Andrea, A. P. Schoellig, and S. Lupashin (2011): "Dance of the quadrocopters (Armageddon)." At `www.tiny.cc/Armageddon`.

[V10] Augugliaro, F., A. P. Schoellig, and R. D'Andrea (2011): "Fast Transitions of a Quadrocopter Fleet Using Convex Optimization." At `www.tiny.cc/QuadroFleetTransition`.

[V11] Lupashin, S., A. P. Schoellig, M. Sherback, and R. D'Andrea (2009): "FMA – Triple Adaptive Flips – ICRA2010 submission." At `www.tiny.cc/TripleFlip`.

[V12] Schoellig, A. P. and S. Lupashin (2009): "ETH FMA Quadrotors @ Nacht der Forschung, Zurich, Oct 25 2009." At `www.tiny.cc/FMANDF2009`.

[V13] Lupashin, S., M. Hehn, A. P. Schoellig, and R. D'Andrea (2010): "Echo in Concert (A Happy Quadrotor New Year!)." At `www.tiny.cc/FMAChristmas2010`.

[V14] Lupashin, S., A. P. Schoellig, M. Hehn, and R. D'Andrea (2011): "The Flying Machine Arena as of 2010 (Final Version)." At `www.tiny.cc/FMA2010`.

[V15] Lupashin, S., C. Fischer, A. P. Schoellig, and M. Waibel (2011): "Juliet's Christmas Tree (Quadrotor New Year 2!)." At `www.tiny.cc/FMAChristmas2011`.

## 2.5 List of Students Advised

Several students were supervised as part of the author's doctoral studies. Below is a complete list of student projects, sorted by the type of project performed.

### MSc Thesis

*The MSc Thesis corresponds to a 6 month, full-time research project.*

[M1] Fabian L. Mueller (Spring 2011): "Implementation and evaluation of iterative learning algorithms for precise quadrocopter trajectory tracking."

[M2] Federico Augugliaro (Spring 2011): "Dancing quadrocopters: trajectory generation, feasibility, and user interface."

[M3] Philippe Goffin (Spring 2009): "Can we do better than humans do? Learning aerobatic maneuvers from observation."

### MSc Semester Project

*The MSc Semester Project describes a semester-long research project, equivalent to 6 weeks full-time work.*

[S1] Clemens Wiltsche (Spring 2011): "Precise synchronized periodic quadrocopter motion in three dimensions based on feed-forward parameter identification."

[S2] Raphael Wüest (Fall 2010): "New synchronized quadrocopter motions: bounce motions in 2D."

[S3] Federico Augugliaro (Spring 2010): "A platform for dance performances with multiple quadrocopters: graphical user interface and demonstration."

[S4] Javier Alonso-Mora (Spring 2009): "Extending iterative learning control to multi-agent systems."

[S5] Sonja Stüdli, (Spring 2009): "Fly! Iterative learning control for quadrocopters."

### BSc Thesis

*The BSc Thesis corresponds to a 3 months, full-time research project.*

[B1] Raphael Schottenhaml (Spring 2011): "Extensions to the rhythmic side-to-side motion."

[B2] Benjamin Troxler (Spring 2009): "Generation of acrobatic trajectories for quadrocopters."

[B3] Fabian L. Mueller (Spring 2009): "An automated testing platform for learning algorithms."

[B4] Federico Augugliaro (Spring 2009): "Synchronizing motion and music beat–a dancing quadrocopter."

### BSc Studies on Mechatronics

*The BSc Studies on Mechatronics is a semester-long literature review, equivalent to 4 weeks full-time work.*

[L1] Timon Heinis (Spring 2011): "Exploring software tools for music analysis."

[L2] Robert Stettler (Spring 2010): "Interaction and information sharing between multiple systems."

[L3] Benjamin Troxler (Spring 2009): "A quadrocopter learns acrobatic maneuvers – trajectory generation and control methods."

[L4] Fabian L. Mueller (Spring 2009): "Swing-up of a pendulum: a benchmark problem."

[L5] Federico Augugliaro (Spring 2009): "Synchronizing motion and music beat."

# 3

# The Flying Machine Arena

The Flying Machine Arena (FMA) is an experimental platform used primarily for research in aerial robotics and autonomous control. Physically, it features a 10 x 10 x 10 meter space protected by nets and padding, with motion capture cameras for vehicle state observation, see Fig. 3.1. In the following sections, we describe the system architecture, the type of vehicles we use, and the vehicle control in more detail.

Many people were instrumental in creating the infrastructure for the FMA. In particular, Sergei Lupashin, Markus Hehn, Michael Sherback and Guillaume Ducard made major contributions. Felix Althaus, Christoph Wegmüller, Lorenz Meier, Thomas Kägi and Flavio Fontana provided further useful features.

### System Architecture

The overall organization of the FMA is similar to the MIT and GRASP testbeds [8, 9]. Fig. 3.2 provides an overview on the overall architecture. The space is equipped with an 8-camera motion capture system that, for any properly marked vehicle, provides millimeter-accurate position information and degree-precise attitude data at 200 Hz with a latency of about 10 ms. The localization data is sent to a PC, which runs estimation and control algorithms–including the learning and identification algorithms proposed in this thesis. These in turn deliver commands

**Figure 3.1** Conceptual drawing (left) and recent photo (right) of the ETH Flying Machine Arena (FMA). The object in the corner of the FMA photo is the 1.2m-wide ETH Balancing Cube. Videos from the FMA can be found at `www.FlyingMachineArena.org`.

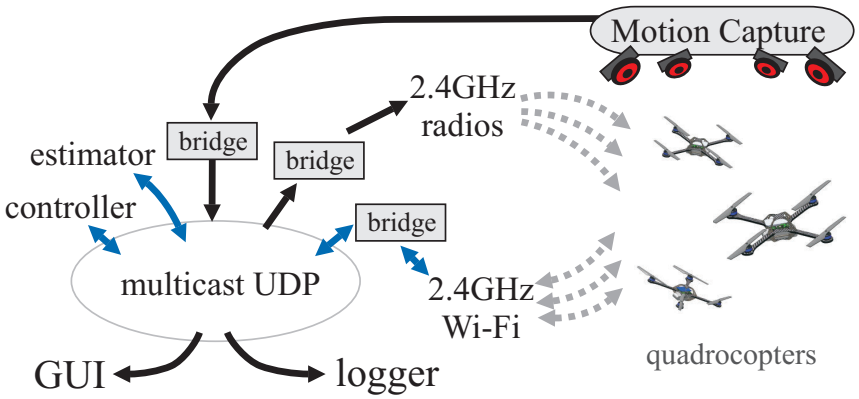



**Figure 3.2** System architecture of the FMA testbed. Picture taken from [R6].

to the quadrocopters with an approximated latency of 20 ms. Communication with vehicles is done via wireless channels as described below (see section 'Vehicles'). The overall system time delay, from sending a vehicle command to detecting the corresponding effects in the vehicle's pose data, varies between 10 ms and 40 ms with a mean value of 35 ms. We take into account this mean value when estimating the vehicle pose and computing the control commands.

Data is sent via a multicast UDP (User Datagram Protocol) scheme, allowing for convenient online visualization of all data sent over the network, and also for recording, playback, and export of arbitrary customized data series. A convenient side-effect of this setup is that estimation and control components are binary-identical whether running in the real system or in simulation. The wireless and Vicon data bridges are simply replaced by a simulator process, with all of the other components remaining completely unaffected and unaware of any difference.

### Vehicles

The vehicles of choice in the FMA are small quadrocopters, based on the Ascending Technologies Hummingbird platform described in [10]. Fig. 3.3 shows the current flying vehicle. Measuring 53 cm in diameter, the vehicle's total weight including the onboard battery is 460 g. The operational flying time varies between 10 and 20 minutes depending on the aggressiveness of the performed flight maneuvers. The vehicle is equipped with four brushless DC motors, which together are able to produce a vertical acceleration of approximately 12.5 m/s$^2$. The vehicles have proven to be agile and robust platforms, that can sustain even a hard crash with typically minor consequences.

Though the original propulsion system, the motor controllers and the frame of the standard Hummingbird quadrocopter were preserved, the wireless communication, central electronics and onboard sensors were replaced to obtain better control over the onboard algorithms, and to have access to better-quality and higher-range rate gyro data. These changes allow for more aggressive maneuvers, faster turn rates, and generally better flight performance. With the new rate gyros, rotations of up to 2000 deg/s are possible around the body's principal axes of inertia. Detailed documentation of the changes are found in [11].

The quadrotor accepts three body angle rate commands and a collective thrust command at 50 Hz. An onboard 800 Hz feedback controller uses rate gyros to track the given commands. No feedback is currently done on the thrust command. The exact onboard controller design

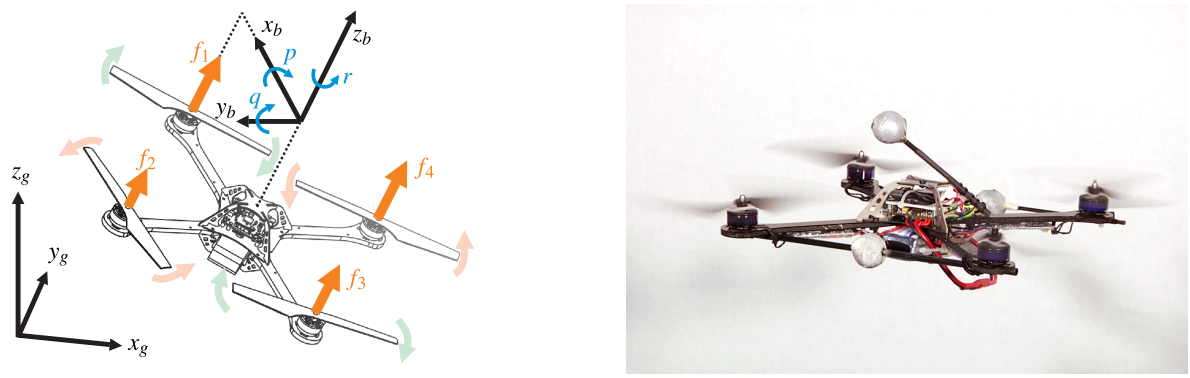



**Figure 3.3** Schematic drawing of the quadrocopter introducing local and global coordinate systems, propeller rotation directions, and motor numbering (left). The photo on the right shows one of the FMA quadrocopters in hover. Pictures are taken from [12].

is presented in [11]. Each vehicle is equipped with two radio systems: a one-way 2.4 GHz module used exclusively for controlling the vehicle (to constrain the amount of variable latency in the system), and a bidirectional 2.4 GHz transceiver with a different modulation for non-time-critical communication such as data feedback or onboard parameter reads/writes, see Fig. 3.2.

### Control

The Flying Machine Arena relies on a strongly cascaded control scheme. As mentioned before, minimal control is performed on board the vehicles: Only states that are directly measured on board are controlled; that is, only rotational rates are controlled using feedback from the gyroscopic sensors. The four inputs to the vehicle are usually provided by off-board controllers. For the experiments in this thesis, we usually use two different modes for controlling the vehicle.

In the first mode, the developed learning scheme provides the collective thrust command and the three body angle rates commands directly. That is, no feedback from the cameras is used during the execution of a trajectory. This control mode is partly used in Paper I.

In the second control mode, the developed learning schemes are built on top of an off-board trajectory-following controller (see e.g. Fig. 2.1). The trajectory-following controller takes desired vehicle positions as an input and closes the loop based on the camera information. The off-board controller calculates all four vehicle commands and sends them to the vehicle. This control mode is extensively use in Part B, but also in the iterative learning approach presented in Paper II.

More details about the testbed are found at `www.FlyingMachineArena.org`.

# 4

# Future Directions

This chapter provides a brief summary on the current status of both projects and comments on potential future research directions.

## 4.1 Part A. Iterative Learning

The optimization-based learning algorithm developed in Paper I and the improvements made in Paper II are a solid basis for further investigation. We have an effective and conceptually simple algorithm at hand that takes *a priori* knowledge about the system dynamics into account, and that is applicable to any dynamic system (including systems with underlying feedback loops) for which a numerical dynamics simulation is available. Because of the acausal learning action, which corrects for repetitive disturbances before they occur, the final tracking performance of the proposed learning scheme outperforms pure feedback control. We have shown that the algorithm is able to reliably learn arbitrary three-dimensional quadrocopter motions.

Future directions of this research may include:

**Transferring knowledge between different tasks** Traditionally, the focus of ILC has been on improving the performance of systems that execute a single, repeated operation [6]. When modifying the desired task, the system starts from zero, without any knowledge from the original maneuver. Since one can expect that the past experience contains useful information for a next task, especially if tasks are similar, a potential area for further research is to develop methods that allow us to propagate knowledge between different tasks. Having successfully learned one task, one idea is to extract relevant information from the collected data and incorporate it into the model to facilitate the learning of a second task. The potential of such an approach was highlighted by a previous work in the group, see [13]. Another idea is to learn a set of basic tasks, cf. [14]. By generating trajectories based on these basic elements, prior experience may be reused.

**Simultaneous system identification and learning** More sophisticated methods may be able to simultaneously identify system parameters and learn a desired task, improving the speed of convergence in the learning by building upon an increasingly accurate model. In particular, when applying a numerical identification routine as proposed in Paper II, an option is to re-identify the system around the most recent trajectory after a few iterations.

**Efficiency of trajectory generation** In view of the interactive slalom demonstration that we developed in Paper II (see Sec. 2.1), a trajectory-generation algorithm with a lower com-

putation time would be a major improvement. It would allow for a more intensive user experience because of the lower waiting time. In addition, the current trajectory-generation algorithm does not yet take all degrees of freedom into account in the optimization routine. This would clearly be desirable when aiming for computing trajectories that are as fast as possible.

**Learning in a multi-vehicle setup** We have considered a particular multi-agent framework in Paper III and Paper IV. First analytical results showed that there is only limited benefit from exchanging information when multiple agents learn the same task simultaneously. Different scenarios may be investigated, such as learning cooperative tasks where the motion of the agents requires coordination, see e.g. [5].

## 4.2 Part B. Rhythmic Flight Performances

The current status of this project is nicely summarized by the software tool that we recently developed. The software allows for computer-assisted choreography design via a graphical user interface and includes the following features: a library of motion primitives containing periodic and non-periodic, aerobatic motions; a scripting language that allows for easy parameterization and concatenation of motion primitives; the computation of collision-free transition motions from a set of initial quadrocopter states to a set of final states (a useful tool for connecting different motion primitives); an automatic feasibility check of the planned choreography; the possibility of simulating the performance prior to actual flight; and, an offline learning and online adaptation scheme for precise trajectory following of periodic motions (based on the results in Paper V and Paper VII).

Directions for future development may include:

**Automated generation of a choreography based on human rating** The idea is to create a system that automatically combines motion primitives in a meaningful way given the music structure. The decision process can be improved by exploiting human rating and evaluation.

**Interaction** A possible new research direction is real-time interaction. A quadrocopter may, for example, be told what to do by a human instructor in the space. This could possibly mimic the human process of testing and learning a choreography.

**Software improvements** The software can be improved in various aspects, in particular with respect to its usability. The scripting language may be replaced by graphical objects that are concatenated via drag-and-drop. In addition, a static 3D visualization of the trajectories would facilitate the design process. Once the structure of the project is finalized and most of the tools have been developed, the software design may be outsourced.

**Automated music analysis** In order to achieve a fully automated choreography generation, the music structure must be provided automatically as well. A first literature and software study was done. Now, relevant software must be tested and eventually included into the existing software framework. A collaboration with the Music Information Retrieval community may be sought.

**Improvements on motion control and feasibility algorithms** Recently, a nonlinear attitude controller was developed at our institute, which extends the envelope of flyable motions of the trajectory-following controller and improves its tracking performance. It is a logical next step to re-configure our setup and use this controller component. This step is easy to do thanks to the modular design of the project and software. In addition, the algorithm for evaluating a choreography's feasibility, as well as the feed-forward synchronization scheme would benefit from an extensive experimental testing and validation. We ran only preliminary tests that supported the theoretical analysis.

In brief, future work aims for further facilitating the choreography design and continuing the efforts to improve the actual flight and tracking performance. An ultimate goal is an automated choreography design, where flight performance is created fully automatically for any randomly selected song.

# References


[1] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill Science/Engineering/Math, 1997.

[2] H. Ahn, Y. Chen, and K. Moore, "Multi-agent coordination by iterative learning control: Centralized and decentralized strategies," in *Proceedings of the IEEE International Symposium on Intelligent Control (ISIC)*, 2011, pp. 394–399.

[3] D. Meng and Y. Jia, "Iterative learning approaches to design finite-time consensus protocols for multi-agent systems," *Systems & Control Letters*, vol. 61, no. 1, pp. 187–194, 2012.

[4] H.-S. Ahn, K. L. Moore, and Y. Chen, "Trajectory-keeping in satellite formation flying via robust periodic learning control," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 14, pp. 1655–1666, 2010.

[5] K. Barton, D. Hoelzle, A. Alleyne, and A. Johnson, "Cross-coupled iterative learning control of systems with dissimilar dynamics: design and implementation," *International Journal of Control*, vol. 84, no. 7, pp. 1223–1233, 2011.

[6] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[9] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[10] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 361–366.

[11] S. Lupashin, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[12] S. Lupashin and R. D'Andrea, "Adaptive fast open-loop maneuvers for quadrocopters," *Autonomous Robots*, vol. 33, pp. 89–102, 2012.

[13] O. Purwin and R. D'Andrea, "Performing and extending aggressive maneuvers using iterative learning control," *Robotics and Autonomous Systems*, vol. 59, no. 1, pp. 1–11, 2011.

[14] D. J. Hoelzle, A. G. Alleyne, and A. J. W. Johnson, "Basis task approach to iterative learning control with applications to micro-robotic deposition," *IEEE Transactions on Control Systems Technology*, vol. 19, pp. 1138–1148, 2011.

# Part A

# ITERATIVE LEARNING

# Paper I

# Optimization-Based Iterative Learning for Precise Quadrocopter Trajectory Tracking

Angela P. Schoellig · Fabian L. Mueller · Raffaello D'Andrea

**Abstract**

Current control systems regulate the behavior of dynamic systems by reacting to noise and unexpected disturbances as they occur. To improve the performance of such control systems, experience from iterative executions can be used to anticipate recurring disturbances and proactively compensate for them. This paper presents an algorithm that exploits data from previous repetitions in order to learn to precisely follow a predefined trajectory. We adapt the feed-forward input signal to the system with the goal of achieving high tracking performance – even under the presence of model errors and other recurring disturbances. The approach is based on a dynamics model that captures the essential features of the system and that explicitly takes system input and state constraints into account. We combine traditional optimal filtering methods with state-of-the-art optimization techniques in order to obtain an effective and computationally efficient learning strategy that updates the feed-forward input signal according to a customizable learning objective. It is possible to define a termination condition that stops an execution early if the deviation from the nominal trajectory exceeds a given bound. This allows for a safe learning that gradually extends the time horizon of the trajectory. We developed a framework for generating arbitrary flight trajectories and for applying the algorithm to highly maneuverable autonomous quadrotor vehicles in the ETH Flying Machine Arena testbed. Experimental results are discussed for selected trajectories and different learning algorithm parameters.

Published in *Autonomous Robots,* 2012. Submission includes multimedia attachment, also found at `www.tiny.cc/QuadroLearnsTrajectory`. DOI: 10.1007/s10514-012-9283-2.

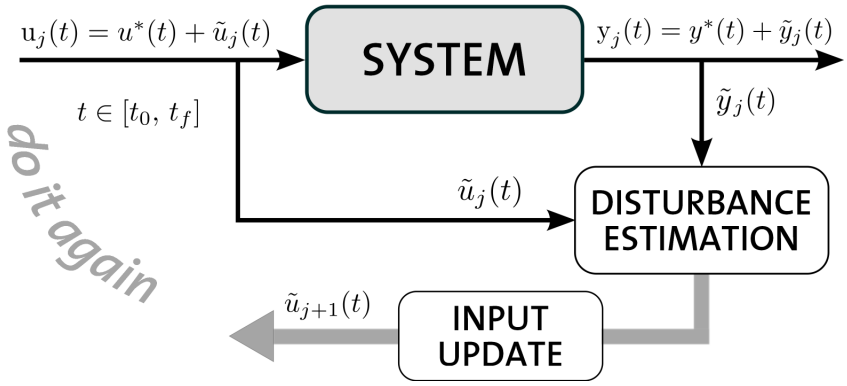



**Figure 1.** The general iterative learning framework considered in this paper: A complete trial $\mathrm{u}_j(t)$, $t \in [t_0, t_f]$ is performed. Based on the output error $\tilde{y}_j(t)$, a new input $\mathrm{u}_{j+1}(t)$ is calculated and applied during the next trial.

# 1. Introduction

## 1.1 Goal & Motivation

Over the last century, controls of dynamic systems have expanded from mechanical and analog-electronic controllers for limited subproblems (such as course stabilization of ships) to digital control of fully autonomous systems (such as unmanned aerial vehicles). This trend has been enabled by technical advancements in sensors, actuators, and digital computing components, as well as by significant developments in the theoretical foundations of control. Like their early counterparts, current control systems usually regulate the behavior of dynamic systems by reacting to noise and unexpected disturbances in the measured system output. Typically, they are based on a mathematical model of the system dynamics. The performance of this approach is limited by the accuracy of the dynamics model and the causality of the control action that is compensating only for disturbances as they occur. Unfavorable effects of these limitations are observed especially in regimes where feedback is not able to react in time and the dynamic behavior is difficult to identify or understand. To achieve high performance in such cases, we propose data-based control approaches that are able to store and interpret information from past executions, and infer the correct actions or control laws for future experiments.

The objective of this paper is to explore the field of data-based high performance control by developing algorithms that enable systems to precisely track predefined trajectories. We propose an iterative learning scheme with the goal of achieving high tracking performance – even under the presence of model errors, parameter uncertainties, and other recurring disturbances. Fig. 1 depicts the general idea of the algorithm. Data is collected through a repeated execution of the same task and the performance is improved from trial to trial by identifying recurring disturbances and adapting the feed-forward input signal accordingly. We leverage a model of the system's key dynamics to increase the efficiency of the learning and the speed of convergence. Thus, our approach applies to any underlying dynamic system for which a nominal model is available. Since the correcting action is executed only after a complete run

of the trajectory, the approach is not restricted by slow feedback rates or large system latencies. Furthermore, it is not limited to a causal action, which reacts only to disturbances after they occurred. Instead, recurring disturbances (mainly due to modeling errors) are anticipated and proactively compensated for before they occur. This approach is thus suitable for performing aggressive trajectories.

We apply the algorithm to quadrotor vehicles in the ETH Flying Machine Arena. Quadrotor vehicles offer exceptional agility in the rotational and translational degrees of freedom due to the large torques generated by the off-center mounted propellers and the high thrust-to-weight ratio. When operating these vehicles at high speeds, complex dynamic effects such as aerodynamics, battery behavior, and motor dynamics have a significant impact on the vehicle behavior. These effects are difficult to model but can be compensated for by an iterative execution.

Due to recent technological advances in aerial robotics, interest in using micro aerial vehicles for industrial applications, including exploration and surveillance, inspection and monitoring, and transportation and entertainment has grown. As such, precise trajectory tracking will become relevant for operations where the tracking performance of the system determines the quality of the experimental result. Iterative learning will be applicable if repetition is inherent to the required task. Examples include inspection of civil infrastructure (such as bridges, highways and dams), environmental monitoring (of forest, rivers, lakes, etc.) or filming a scene with a camera mounted on a robot. Common to all these examples is that 'measurements' must be taken along pre-computed paths.

We have developed a framework for the generation and iterative learning of flight trajectories for quadrocopters. Feasible flight trajectories are generated based on user input that defines the shape of the desired flight path in the vertical plane. We restrict ourselves to two-dimensional trajectories for the sake of simplicity (though all derivations generalize to 3D trajectories). Trajectory feasibility is considered with respect to the corresponding first-principles model of the vehicle and sensor/actuator constraints. Based on the same model, the proposed learning scheme is derived, which iteratively improves the trajectory tracking performance by adapting the feed-forward input signal. This adaptation does not change the underlying dynamics of the vehicle, and thus has an advantage over stiff and switching controllers, which may cause non-smooth motions and undesirable transients (especially in cases where the underlying model of the system is inaccurate).

## 1.2 Related Work

Research in aggressive flying and trajectory tracking of autonomous quadrocopters has made considerable progress over the last decade, and strategies for both generating reference trajectories and for designing effective and robust control algorithms have improved. Since quadrotor vehicles are inherently unstable nonlinear systems, and because they exhibit exceedingly complex behavior at high speeds, one approach is to extend classical control methods with sophisticated adaptation and learning schemes designed to cope with the complicated system dynamics, unavoidable model uncertainties and external disturbances.

Examples of classical control approaches used to track trajectories with quadrocopters are PID schemes [1], backstepping control techniques [2–7] and feedback linearization [8,9]. Other common strategies are trajectory linearization control [10], constrained finite-time optimal control [11], LQ optimal solutions [12] or Model Predictive Control [13]. Such control schemes fall in the area of causal controllers. Applying one of these approaches and performing the

same trajectory over and over again, results in the same tracking error in each trial (on average). These schemes are not designed to exploit past experience in order to improve future performances.

Other research on the control of flying vehicles has focused on learning schemes. In [14], where altitude control of quadrocopters is studied, integral sliding mode control and reinforcement learning techniques are compared. Neural networks and output feedback are used in [15] to learn the complete dynamics of the vehicle online. Work in [16] and [17] utilizes adaptive control to achieve both, adaption to unknown payloads and robustness to disturbances. While most of these approaches are concerned with near-hover operations, the goal of our approach is to track fast trajectories. As shown in Sec. 4, we generate reference trajectories that minimize execution time.

Other learning strategies have been developed for fast quadrocopter aerobatics. A method to learn high-speed quadrocopter multi-flips was introduced in [18], which uses a policy gradient method to iteratively learn parameterized flip primitives. Similar approaches are used in [19–21] to achieve various high-speed, high-performance maneuvers. When comparing those approaches to the approach presented in this paper, the main difference is the level of specificity of the desired reference trajectory. While we aim to follow a continuous trajectory, work in [18–21] compares the actual and desired states only at specific key frames.

The approach presented in this paper can be characterized as an iterative learning control (ILC) technique and is based on our previous work in [22]. ILC became a popular research topic beginning with [23], and has since proven to be a very powerful method for high performance reference tracking (a recent overview of ILC with an extensive bibliography is available in [24] and [25]). Yet methods from optimal control theory have only recently been applied to the design of ILC laws. Based on a so-called 'lifted' domain representation, cf. [26–28], LQG-type solutions have been proposed by [25, 29–32] for estimating the tracking error and minimizing a quadratic cost function. Work in [24, 30, 33, 34] has shown that ILC can be applied to systems with underlying feedback loops. The real-time feedback component is intended to reject non-repetitive noise while the ILC adjusts to the repetitive disturbance.

In practice ILC has been applied to repetitive tasks performed by stationary systems, such as wafer stages, chemical reactors, and industrial robots. Applications to autonomous vehicles are more rare. There is one example where ILC was applied to quadrocopters: [35] introduce a least-squares based learning rule to improve on horizontal point-to-point motions. The work presented in this paper can be viewed as an extension of the results in [35], and addresses the issues referred to as 'open questions' in [35]. We consider a larger class of motions – namely, arbitrary trajectories in the vertical plane – and a generalized two-step learning framework, which is discussed in more detail below. Input and state constraints of the system are explicitly incorporated in our learning rule.

### 1.3 Contribution to the Field of Feed-forward Based Trajectory Learning

The contribution of this paper to the field of feed-forward based trajectory learning is twofold:

First, we develop an algorithm that structures the trajectory learning problem around a disturbance estimation and input update step, see Fig. 1. Both steps rely on a nominal model of the underlying system. Estimation and input update are clearly separated, which allows for a flexible combination of different approaches for both steps. More important from a practical point of view, the clear separation allows for an intuitive tuning of the overall learning scheme.

In the first step, we design a time-varying Kalman filter, which estimates the model error along the trajectory. The estimated error serves as the input to the following control step. The Kalman filter explicitly takes noise characteristics into account, which can be adjusted to improve the convergence of the estimation and, thus, of the overall learning. In the second step, the control objective is formulated as a convex optimization problem [36]. Here, in contrast to least-squares approaches or LQG design [25, 29–32], input and state constraints can be explicitly incorporated. Different (nonlinear) performance objectives can be defined by choosing appropriate vector norms and adequate scaling and weighting of the error vector. Moreover, derivatives of the input can be included into the objective function to reduce jittering in the control inputs and, consequently, improve the robustness of the learning. The definition of a termination condition justifying the linearization of the system dynamics is unique in the area of ILC and results in a learning process that better meets the need for safe and efficient operation.

Second, the derived learning scheme is thoroughly applied to quadrocopters to achieve fast and accurate trajectory tracking. The paper presents an entire unified process, including the generation of feasible reference trajectories, the application to real quadrotor vehicles, and a detailed experimental study characterizing both the influence of different learning parameter settings as well as features of the experimental setup and the quadrotor vehicles.

The theoretic approach, as well as the application of the algorithm to real quadrotor vehicles, makes this work an original contribution to the field.

### 1.4 Outline

The paper is organized as follows:

In Sec. 2, we present the iterative learning algorithm in its general form. The learning scheme is introduced as a two- step process of first estimating the unknown repetitive disturbance (Sec. 2.2) and later compensating for it (Sec. 2.3). The approach is based on a dynamic model of the system (Sec. 2.1) and includes the unique feature of gradually increasing the trial horizon (Sec. 2.4).

In the second part of the paper, Sec. 3-7, we apply the algorithm to quadrotor vehicles, and develop a complete framework tailored towards generating and learning arbitrary flight trajectories in the vertical plane. A model of the quadrocopter dynamics and constraints is derived first, in Sec. 3, and a method for generating feasible reference trajectories is presented in Sec. 4. The experimental setup and implementation details are illustrated in Sec. 5. Finally, Sec. 6 presents the quadrocopter's learning behavior in actual experiments.[1] We conclude with a discussion on the limitations of the proposed approach in Sec. 7 and summarize the presented results in Sec. 8.

## 2. The Learning Algorithm

The basic idea of the proposed learning scheme is to use iterative experiments to teach a dynamic system how to precisely follow a trajectory. By exploiting the experience gained from previous trials, the system learns to anticipate recurring disturbances (that are mainly due to

[1] The accompanying video is found at `www.tiny.cc/QuadroLearnsTrajectory`.

modeling errors) and to compensate for them in a non-causal way. We execute a learning update after each trial by combining *a priori* knowledge about the system's dominating dynamics with real measurements from experiments.

The basic procedure is depicted in Fig. 1 and is described as follows: We assume that we can derive a model that captures the key dynamics of the underlying system. This model is used to calculate the nominal input and state trajectories. Moreover, by linearizing the system about the nominal trajectory and discretizing the resulting equations, we can derive a static map that describes the system dynamics during one trial (Sec. 2.1). The learning algorithm builds upon this lifted model when interpreting the data of one trial and updating the feed-forward input signal for the next trial. These two steps are clearly separated. We use a Kalman filter to interpret the measurement of the last trial and to incorporate the measurement into the current estimate of the disturbance (Sec. 2.2). The input update step takes the current disturbance estimate and returns a more adequate input for the next trial by solving a constrained optimization problem (Sec. 2.3). The input serves as a feed-forward reference signal to the underlying system. After each iteration the system is reset to the initial state. Safe and gradual learning is ensured by including a predefined termination condition that stops a trial whenever the actual trajectory diverges from the desired one by an unacceptable amount (Sec. 2.4).

Key concepts including the system representation, disturbance estimation, input update, and extending horizon learning are presented first. The main steps of the algorithm from an application perspective are summarized in Sec. 2.5, where we also highlight important prerequisites of the approach. Finally, in Sec. 2.6 we analyze the computational complexity of the approach.

### 2.1 Model of Dynamics and Lifted-domain Representation

We assume that we can derive a model that captures the key dynamics of the physical system under consideration. In the general case, the system dynamics are modeled by a set of time-varying nonlinear differential equations,

$$\begin{aligned} \dot{\mathrm{x}}(t) &= \mathrm{f}(\mathrm{x}(t),\mathrm{u}(t),t) \\ \mathrm{y}(t) &= \mathrm{g}(\mathrm{x}(t),t), \end{aligned} \tag{1}$$

where $\mathrm{u}(t) \in \mathbb{R}^{n_\mathrm{u}}$ denotes the system input, $\mathrm{x}(t) \in \mathbb{R}^{n_\mathrm{x}}$ the system state, and $\mathrm{y}(t) \in \mathbb{R}^{n_\mathrm{y}}$ the output. The vector fields f and g are assumed to be continuously differentiable in x and u. Constraints on the state $\mathrm{x}(t)$, the input $\mathrm{u}(t)$, and respective time derivatives are represented by

$$Z\,\mathrm{q}(t) \leq \mathrm{q}_{\max}, \tag{2}$$

where

$$\mathrm{q}(t) = \left[\mathrm{x}(t),\, \mathrm{u}(t),\, \dot{\mathrm{x}}(t),\, \dot{\mathrm{u}}(t),\, \ldots,\, \frac{d^m}{dt^m}\mathrm{x}(t),\, \frac{d^m}{dt^m}\mathrm{u}(t)\right] \tag{3}$$

and $\mathrm{q}_{\max} \in \mathbb{R}^{n_\mathrm{q}}$. The inequality is defined component-wise where $n_\mathrm{q}$ is the total number of constraints, and $Z$ is a constant matrix of appropriate dimensions. Eq. (2) allows the incorporation of constraints on any linear combination of $\mathrm{x}(t)$, $\mathrm{u}(t)$, and their time derivatives, for example,

$$\mathrm{x}(t) \leq \mathrm{q}_{x,\max} \qquad \text{and} \qquad a\,\mathrm{u}(t) + b\,\dot{\mathrm{u}}(t) \leq \mathrm{q}_{u,\max}, \quad a,b \in \mathbb{R}. \tag{4}$$

The goal of our learning algorithm is to track an *a priori* determined output trajectory $y^*(t)$ over a finite-time interval $t \in \mathscr{T} = \left[t_0, t_f\right]$, $t_f < \infty$. We assume that the desired trajectory $y^*(t)$, $t \in \mathscr{T}$, is feasible with respect to the nominal model (1), (2). That is, there exists a triple

$$(u^*(t), x^*(t), y^*(t)), \qquad t \in \mathscr{T}, \tag{5}$$

satisfying (1) and (2). For some applications, the desired output trajectory $y^*(t)$ may be known ahead of time. However, it may also be the result of an optimization problem as shown in Sec. 4.

Below, we derive a system representation of (1)-(5) that facilitates the derivation and implementation of the learning algorithm. First we assume that the motion of the system stays close to the generated reference trajectory (5) during the learning process. (Note that this can be enforced by the extending horizon feature introduced in Sec. 2.4.) We linearize the dynamics around the reference trajectory. Considering only small deviations $(\tilde{u}(t), \tilde{x}(t), \tilde{y}(t))$ from the desired trajectory (5),

$$\tilde{u}(t) = \mathrm{u}(t) - u^*(t), \qquad \tilde{x}(t) = \mathrm{x}(t) - x^*(t), \qquad \tilde{y}(t) = \mathrm{y}(t) - y^*(t), \tag{6}$$

the system's behavior (1) can be approximated by a first-order Taylor series expansion about the reference trajectory (5) (cf. [29]) resulting in the following linear, time-varying system

$$\begin{aligned} \dot{\tilde{x}}(t) &= A(t)\tilde{x}(t) + B(t)\tilde{u}(t) \\ \tilde{y}(t) &= C(t)\tilde{x}(t), \qquad\qquad t \in \mathscr{T}, \end{aligned} \tag{7}$$

where the time-dependent matrices $A(t), B(t), C(t)$ are the corresponding Jacobian matrices of the nonlinear functions f and g with respect to x and u. The input-output relationship as given by (7) is fundamental to the model-based learning scheme proposed subsequently. On the real system, however, inputs are sent at discrete times, and measurements are available only at fixed time intervals. To capture this fact, we derive a discrete-time representation of the plant dynamics (7), cf. [24, 25, 37], and references therein. Converting (7) to a discrete-time system results in the following linear, time-varying difference equations,

$$\tilde{x}(k+1) = A_D(k)\tilde{x}(k) + B_D(k)\tilde{u}(k)\tilde{y}(k+1) = C_D(k+1)\tilde{x}(k+1), \tag{8}$$

where $k \in \mathscr{K} = \{0, 1, \ldots, N-1\}$, $N < \infty$, denotes the discrete-time index and $N$ is the trial length in discrete-time steps, $N = t_f/\Delta t$ with $\Delta t$ being the sampling time and $t_f$ assumed to be a multiple of $\Delta t$. That is, the desired trajectory (5) is represented by an $N$-sample sequence

$$(u^*(k), x^*(k+1), y^*(k+1)), \;\; k \in sK, \tag{9}$$

with given initial state $x^*(0)$.

Other associated signals, e.g. (6), are discretized analogously. The constraints (2) are similarly transformed,

$$Z\,\tilde{q}(t) \leq \mathrm{q}_{\max} - Z\,q^*(t) := \mathrm{q}_{\max}(t) \tag{10}$$

where $\tilde{q}(t)$ is the deviation of $\mathrm{q}(t)$ from the corresponding nominal values $q^*(t)$ defined analogously to (6). Discretizing the above equation results in

$$Z\,\tilde{q}(k) \leq q_{\max}(k), \tag{11}$$

where the derivative components in $\tilde{q}(k)$ are replaced by a discrete approximation. For example, $\Delta\tilde{u}(k) = (\tilde{u}(k) - \tilde{u}(k-1))/\Delta t$ may be used as an approximation for the input derivative. The values of the vector $q_{\max}(k) \in \mathbb{R}^{n_\mathrm{q}}$ depend on the discretization method.

Introducing the lifted vector representation, cf. [38],

$$\begin{aligned} u &= [\,\tilde{u}(0), \tilde{u}(1), \ldots, \tilde{u}(N-1)\,]^T \quad \in \mathbb{R}^{Nn_\mathrm{u}} \\ x &= [\,\tilde{x}(1), \ldots, \tilde{x}(N)\,]^T \quad \in \mathbb{R}^{Nn_\mathrm{x}} \\ y &= [\,\tilde{y}(1), \ldots, \tilde{y}(N)\,]^T \quad \in \mathbb{R}^{Nn_\mathrm{y}}, \end{aligned} \tag{12}$$

the dynamics (8) of a complete trial are captured by a static mapping

$$\begin{aligned} x &= Fu + d^0 \\ y &= Gx, \end{aligned} \tag{13}$$

where the lifted matrix $F \in \mathbb{R}^{Nn_\mathrm{x} \times Nn_\mathrm{u}}$ is composed of the matrices $F_{(l,m)} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}}$, $1 \leq l, m \leq N$,

$$F = \begin{bmatrix} F_{(1,1)} & \cdots & F_{(1,N)} \\ \vdots & \ddots & \vdots \\ F_{(N,1)} & \cdots & F_{(N,N)} \end{bmatrix}, \tag{14}$$

with

$$F_{(l,m)} = \begin{cases} A_D(l-1)\ldots A_D(m)B_D(m-1) & \text{if } m < l \\ B_D(m-1) & \text{if } m = l \\ 0 & \text{if } m > l\,. \end{cases}$$

The matrix $G$ is block-diagonal and analogously defined by

$$G_{(l,m)} = \begin{cases} C_D(l) & \text{if } l = m \\ 0 & \text{otherwise} \end{cases}$$

where $G_{(l,m)} \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{x}}$. Vector $d^0$ contains the free response of the system (8) to the initial deviation $\tilde{x}(0) = \tilde{x}_0 \in \mathbb{R}^{n_\mathrm{x}}$,

$$d^0 = \left[ (A_D(0)\tilde{x}_0)^T,\, (A_D(1)A_D(0)\tilde{x}_0)^T, \ldots, \left(\prod_{i=0}^{N-1} A_D(i)\,\tilde{x}_0\right)^T \right]^T.$$

This lifting technique is well-suited for the analysis and synthesis of iterative learning schemes, where the system is assumed to operate in a repetitive mode, cf. [26–28, 31]. The static linear

mapping (13) captures the complete time-domain dynamics of a single trial by mapping the finite input time series $\tilde{u}(k)$, $k \in \mathscr{K}$, onto the corresponding output time series $\tilde{y}(k+1)$, $k \in \mathscr{K}$. The goal of the iterative learning scheme is to use data gathered during previous executions to improve the systems performance from iteration to iteration by updating the feed-forward signal $u$, cf. (12), after each trial. The dynamics of the learning, i.e., the dynamic behavior of a sequence of consecutive trials, can be described in the lifted domain by introducing a subscript $j$ indicating the $j$th execution of the desired task, $j \in \{1, 2, \dots\}$.

The evolution of the system over several iterations is modeled by

$$\begin{aligned} x_j &= Fu_j + d_j + N_\xi \xi_j \\ y_j &= Gx_j + N_\upsilon \upsilon_j, \end{aligned} \tag{15}$$

with

$$d_j = d_{j-1} + \omega_{j-1}. \tag{16}$$

Here, $j$ denotes the $j$th trial. The signals $\xi_j$ and $\upsilon_j$ account for process and measurement noise, respectively. These noise signals vary from iteration to iteration and are assumed to be trial-uncorrelated sequences of zero-mean Gaussian white noise. The vector $d_j$ can be interpreted as a repetitive disturbance component that is subject only to slight changes from iteration to iteration, cf. (16) with $\omega_j$ being another trial-uncorrelated sequence of zero-mean Gaussian white noise. The vector $d_j$ captures model errors along the trajectory, including repeating disturbances [39], and repeated nonzero initial conditions [40], which were previously represented by $d^0$, cf. (13). The zero-mean noise component of the initial condition is part of the random variable $\xi_j$.

In the model (15)-(16), the state deviation $x_j$ from the reference trajectory $x^*$, $x^* = [x^*(1), \dots, x^*(N)]^T$, is affected by two different noise sources: a trial-uncorrelated zero-mean component $\xi_j$, and a 'random walk' component $d_j$. This versatile noise model includes the stochasticity of the process noise $\xi_j$ and the repetitive nature of the modeling errors $d_j$, which can vary between trials due to the influence of $\omega_j$, see also [29, 32, 33]. In particular, the vector $d_j$ captures all non zero-mean noise effects along the desired trajectory $x^*$. It may also be interpreted as a vector representation of all unmodeled dynamics along the desired trajectory $x^*$. As a result, $d_j$ may depend on the applied input $\mathrm{u}(t) = u^*(t) + \tilde{u}(t)$, $t \in \mathscr{T}$. The ultimate goal of the subsequent derivations is to estimate and optimally compensate for the disturbance $d_j$ by updating the input trajectory appropriately.

To complete the lifted representation (15)–(16), the constraints (11) are transformed appropriately. Note that all entries in $\tilde{q}(k)$ can be expressed by linear combinations of the vectors $x$ and $u$ defined in (12). Introducing $q = [x, u]^T$ and stacking the bounds $q_{\max}(k)$ in a vector as

$$q_{\max} = [q_{\max}(0), q_{\max}(1), \dots, q_{\max}(N)]^T \in \mathbb{R}^{(N+1)n_{\mathrm{q}}}, \tag{17}$$

constraints (11) read as

$$Lq \leq q_{\max}, \tag{18}$$

where $L$ is a constant matrix of appropriate dimensions.

Subsequently, the representation of the model dynamics in the lifted domain by (15), (16), and (18) allows for the derivation and the execution of operations in the trial-time domain.

## 2.2 Disturbance Estimation

We consider our learning algorithm to be a two-step update law, cf. Fig. 1. First, we estimate the modeling error $d_j$ along the desired trajectory using optimal filtering techniques [41]. Then, in order to optimally compensate for the estimated vector $\widehat{d}_j$, we provide a new feed-forward input $u_{j+1} \in \mathbb{R}^{Nn_u}$.

We propose an iteration-domain Kalman filter that retains all available information from previous trials (namely the output signals $y_1, y_2, \ldots, y_j$) in order to estimate the current error $d_j$. Combining (15) and (16), we obtain a discrete-time system that fits into the standard Kalman filter approach, cf. [42]:

$$\begin{aligned} d_j &= d_{j-1} + \omega_{j-1} \\ y_j &= G d_j + GF\, u_j + \mu_j\,, \end{aligned} \tag{19}$$

where, consistent with the previous definitions, the noise term $\mu_j$, $\mu_j = GN_\xi\, \xi_j + N_\upsilon\, \upsilon_j$, is assumed to be zero-mean Gaussian white noise with covariance $M_j$: $\mu_j \sim \mathcal{N}(0, M_j)$. The noise characteristics of $\omega_j$ are given by $\omega_j \sim \mathcal{N}(0, \Omega_j)$. Both stochastic inputs, $\omega_j$ and $\mu_j$, are trial-uncorrelated and assumed to be independent; that is, for $i, j \in \{0, 1, 2, \ldots\}$,

$$\begin{aligned} &E\left[\omega_i \omega_j^T\right] = E\left[\mu_i \mu_j^T\right] = 0 \qquad \text{if } i \neq j \\ &E\left[\omega_i \mu_j^T\right] = 0 \qquad \forall\, i, j\,. \end{aligned} \tag{20}$$

$E[\cdot]$ denotes the expected value.

For the above system (19)-(20), the Kalman filter returns an unbiased disturbance estimate $\widehat{d}_j$ for $j \geq 1$ that minimizes the trace of the error covariance matrix

$$P_j = E[(d_j - \widehat{d}_j)\,(d_j - \widehat{d}_j)^T] \tag{21}$$

of trial $j$ taking measurements $y_m$, $1 \leq m \leq j$, into account.

Given initial values for $\widehat{d}_0$ and $P_0$, the Kalman filter update equations for the specific problem read as:

$$\left\{ \begin{array}{rcl} S_j &=& P_{j-1} + \Omega_{j-1} \\ K_j &=& S_j G^T \left(G S_j G^T + M_j\right)^{-1} \\ P_j &=& \left(I - K_j G\right) S_j\,, \end{array} \right. \tag{22}$$

where $I \in \mathbb{R}^{Nn_x \times Nn_x}$ represents the identity matrix. Based on the optimal Kalman gain $K_j$, and taking into account the previous estimate $\widehat{d}_{j-1}$ and the actual measurement $y_j$, the disturbance estimate $\widehat{d}_j$ is calculated by

$$\widehat{d}_j = \widehat{d}_{j-1} + K_j \left(y_j - G\widehat{d}_{j-1} - GF u_j\right). \tag{23}$$

Note that the matrices in (22) and, especially, the Kalman gains $K_j$ (necessary for an appropriate online update of the error estimate $\widehat{d}_j$) can be calculated prior to the experiment as long as we know the initial value $P_0$.

***Design parameters*** The performance of the estimation can be adjusted by four design parameters:

- The covariance matrix $\Omega_j$, $\omega_j \sim \mathcal{N}(0,\Omega_j)$, indicates the likely change of the disturbance $d_j$ from iteration to iteration. The vector $d_j$ captures the effect of unmodeled dynamics and, hence, may depend on $u_j$. The input $u_j$ changes significantly during the first iterations of the learning, but converges for an increasing number of trials. This may also be true for $d_j$." To account for these changes, one possible definition of the covariance $\Omega_j$ is

$$\Omega_j = \varepsilon_j I\,, \qquad \varepsilon_j > 0, \tag{24}$$

  where the scalar $\varepsilon_j$ is chosen to be larger during the first iterations to guarantee a fast initial adaptation of the disturbance estimate $\widehat{d_j}$, and chosen to be smaller as $j$ increases in order to avoid adapting to outliers and non-repetitive noise.

- The covariance matrix $M_j$, where $\mu_j \sim \mathcal{N}(0,M_j)$, combines the covariance of the process and measurement noise. It is possible to obtain a value for the covariance by carrying given sensor noise characteristics and the known process disturbances of the real system from the original model description (1) through to the lifted domain representation (15). However, modeling $M_j$ as

$$M_j = \eta_j I\,, \qquad \eta_j > 0, \tag{25}$$

  is often sufficient. The ratio between $\varepsilon_j$ and $\eta_j$ determines how much we trust the measurement vs. the process model, cf. (19). Often this ratio is varied by changing $\varepsilon_j$ only and keeping $\eta_j$ constant; that is, $\eta_j = \eta$ for all $j \in \{1,2,\dots\}$. Experimental results discussing the choice of $\eta_j$ and $\varepsilon_j$ are shown in Sec. 6.6.

- The initial value $\widehat{d_0}$ is another design parameter. Most of the time, $\widehat{d_0} = 0$ is a reasonable first guess.

- With the starting value $P_0 = E[(d_0 - \widehat{d_0})\,(d_0 - \widehat{d_0})^T]$, the initial error variance is specified. Choosing $P_0$ to be a diagonal matrix with large positive elements on the diagonal, results in larger changes of $\widehat{d_j}$ at the beginning of the learning.

Note that the entries of $d_j$ and $y_j$, and consequently of $\omega_j$ and $\mu_j$, represent different quantities with different units (e.g. position, velocity, etc. ), whose nominal values may differ by orders of magnitude. To account for this, it may be beneficial to introduce scaling matrices $S_\Omega$ and $S_M$, and define

$$\Omega_j = S_\Omega \left(\varepsilon_j I\right) S_\Omega^T, \qquad M_j = S_M \left(\eta_j\, I\right) S_M^T. \tag{26}$$

To summarize, the advantage of the above approach lies in its explicit incorporation of noise characteristics and its model-based update rule (23), which provides an optimal estimate $\widehat{d_j}$ in the context of the *a priori* tunable parameters (namely the covariances of the disturbances $\omega_j$ and $\mu_j$, and the initial values $P_0$ and $\widehat{d_0}$). The algorithm for the disturbance estimation takes all available information $y_1, y_2, \dots, y_j$ into account. In addition, the modeling error along the desired trajectory $\widehat{d_j}$ may lead to a better system understanding and may be re-used to update the dynamic model (1) or when learning a different reference trajectory.

Based on the disturbance estimate $\widehat{d_j}$, the feed-forward input signal can be adapted in order to compensate for the estimated disturbance, resulting in an improved performance in the next trial.

### 2.3 Input Update

The learning algorithm is completed by the subsequent learning update. Making use of the information provided by the estimator, cf. Sec. 2.2, we derive a nonlinear model-based update rule, which calculates a new input sequence $u_{j+1} \in \mathbb{R}^{Nn_\mathrm{u}}$ in response to the estimated disturbance $\widehat{d}_j$.

The objective of the update step is to find an input $u_{j+1}$, which optimally compensates for the identified disturbance $\widehat{d}_j$. In the context of (15), this means finding an input $u_{j+1}$ that minimizes the deviation from the nominal trajectory in the next trial. More precisely, we consider the expected value of $x_{j+1}$ given all past measurements,

$$E\left[x_{j+1} \,\middle|\, y_1, y_2, \ldots, y_j\right] = F u_{j+1} + \widehat{d}_j \,. \tag{27}$$

The constraints (18) are explicitly taken into account when solving for the optimal $u_{j+1}$. We approximate the future state $x_{j+1}$ in $q_{j+1}$, cf. (18), by (27), resulting in a constraint inequality that depends only on the decision variable $u_{j+1}$.

The update rule can be expressed by the following optimization problem:

$$\begin{cases} \min\limits_{u_{j+1}} \; \left\| S\left(F u_{j+1} + \widehat{d}_j\right) \right\|_\ell \; + \; \alpha \left\| D u_{j+1} \right\|_\ell \\ \text{subject to} \quad L_\mathrm{opt}\, u_{j+1} \leq q_\mathrm{max} \,, \end{cases} \tag{28}$$

where $\alpha \geq 0$ weights an additional penalty term, which was included into the objective function as a means of directly penalizing the input. Via the matrix $D$, the input itself or discrete approximations of its time derivatives can be penalized. This may be beneficial if one wants to enforce smoothness of the optimal input. In Sec. 6.6, an appropriate choice of $D$ and $\alpha$ is discussed in the context of the quadrocopter example. Note that high $\alpha$ values may corrupt or even destroy the learning performance since more emphasis is put on achieving a small (or smooth) input than on minimizing the error along the trajectory.

The matrix $S \in \mathbb{R}^{Nn_\mathrm{x} \times Nn_\mathrm{x}}$ in (28) allows the original error signal (27) to be scaled, and serves several objectives: equalizing the magnitude of the different physical quantities in the lifted domain, penalizing deviations of certain states more than others, or weighting specific parts of a trajectory (e.g. the first part).

The vector norm $\ell$, $\ell \in \{1, 2, \infty\}$, of the minimization (28) affects the convergence behavior and the result of the learning algorithm. For a vector $p = \left(p^{(1)}, p^{(2)}, \ldots, p^{(n_\mathrm{p})}\right) \in \mathbb{R}^{n_\mathrm{p}}$, the one norm ($\ell = 1$), the Euclidean norm ($\ell = 2$), and the maximum norm ($\ell = \infty$), are defined as

$$\|p\|_1 = \sum_{i=1}^{n_\mathrm{p}} |p^{(i)}|, \qquad \|p\|_2 = \sqrt{p^T p}, \qquad \|p\|_\infty = \max_{i \in \left\{1,2,\ldots,n_\mathrm{p}\right\}} |p^{(i)}|.$$

The update law defined by (28) can be transformed into a standard convex optimization problem. More precisely, we obtain a linear program for $\ell \in \{1, \infty\}$ and a quadratic program for $\ell = 2$, cf. [36]. Details on the transformation are provided in Sec. A.1.

Linear and quadratic programs can be solved very efficiently using existing software packages such as [43] (see also comments in Sec. 2.6). Further, if the optimization problem is

feasible (i.e. , if there exist $u_{j+1}$ that satisfy the constraints), then there exists a local minimum that is globally optimal. Furthermore, we formulated the update law as a convex optimization problem so that we could incorporate the input and state constraints explicitly. Such constraints are present in any existing real system and have a notable influence on the dynamic behavior of the system. In particular, when learning high performance maneuvers, constraints often represent the limiting factor for further improvement and should, as such, be taken explicitly into account. Relevant constraints of the quadrotor vehicles and their effects during the learning experiments are illustrated in Sec. 3 and following sections.

***Design parameters*** Four different design parameters allow for an adaptation of the optimization problem (28):

- The norm $\ell$ defines the overall performance objective of the learning algorithm. While $\ell = \infty$ minimizes the maximum deviation from the desired trajectory, the one norm and Euclidean norm minimize the "average error"; that is, they minimize the sum of the deviations along the reference trajectory, where the Euclidean norm weights large errors more than the one norm does. In Sec. 6.6, the learning performance of different norms is experimentally evaluated for the quadrotor tracking problem.
- For an intuitive design, the scaling matrix $S$ may be decomposed into three diagonal matrices,
$$S = T_W S_W S_x, \qquad T_W, S_W, S_x \in \mathbb{R}^{Nn_x \times Nn_x}. \tag{29}$$
First the state scaling matrix $S_x$ scales the lifted state vector such that all entries of the scaled vector representation $x^s$, $x^s = S_x x$, are within the same range of magnitude. The state-weighting matrix $S_W$ puts emphasis on specific states of the original system (1) by penalizing their deviations from the reference trajectory more than the deviations of the other states. The matrices $S_x$ and $S_W$ are usually defined via vectors $s_x$ and $s_W$ of length $n_x$, which are repeatedly placed along the corresponding matrix diagonal. Finally, the matrix $T_W$ allows us to weight particular parts of the trajectory more than others, i.e. to change the scaling along the trajectory.
- When aiming to penalize the input $u$ directly, the matrix $D$ is chosen as the identity matrix, where the lifted vector $u$ represents the deviation from the nominal input trajectory (12). Another option is to consider the rate of change of $u$ or its curvature by choosing $Du$ such that the $k$th component is given by
$$(Du)^{(k)} = \frac{\tilde{u}(k+1) - \tilde{u}(k)}{\Delta t} \qquad \text{or} \tag{30}$$
$$(Du)^{(k)} = \frac{\tilde{u}(k+2) - 2\tilde{u}(k+1) + \tilde{u}(k)}{(\Delta t)^2}, \tag{31}$$
representing the discrete approximation of the first or second derivative of $u$, respectively.
- The scalar $\alpha$ balances the influence of the state deviation component and the input component in the objective function (28). In Sec. 6.6 design criteria for $\alpha$ are discussed in the context of the quadrocopter tracking experiment.

In brief, the above parameters enable us to enforce a specific learning behavior, and thus meet individual performance criteria.

## 2.4 Extending Horizon

Now that we have familiarized the reader with the main ideas of the algorithm in question (namely, the disturbance estimation and input update), we wish to introduce a means of gradually extending its time horizon such that only small deviations from the desired trajectory are guaranteed.

We earlier assumed that each trial $u_j,\ j \in \{1, 2, \dots\}$, is performed over the full time horizon $\mathscr{T} = \left[t_0, t_f\right]$. However, all derivations in Sec. 2.2 and Sec. 2.3 build upon the lifted domain representation, which was the result of a linearization of the system dynamics about the nominal trajectory. The linearization can only be justified if the actual trajectory of the system stays close to the desired one. From a different perspective, it is important that the lifted matrices $F$ and $G$ are valid first-order approximations of the system dynamics as this guarantees a proper interpretation of the measurement (estimation step) and a correct input adaptation (update step). In this section, we introduce a *termination condition* that stops a learning trial whenever the deviation between the real and the desired trajectory grows too large. This guarantees that previous derivations are valid and lead to successful learning.

The general idea of the method is as follows: A new trial is started with an input $u_j \in \mathbb{R}^{Nn_\mathrm{u}}$. During the execution, the actual trajectory may begin to diverge from the reference trajectory; if the deviation exceeds a certain boundary (as specified by the termination condition), the trial is stopped. In this case the learning algorithm, cf. Sec. 2.2 and Sec. 2.3, considers only the first part of the execution (until the premature ending at $N_j < N$) and returns an updated input for the first part of the trajectory. The updated input segment is then extended by the last values of the reference input $u^*(k)$, cf. (9), and a new trial $(j+1)$ is executed. In general, the following trial $(j+1)$ performs better during the initial segment of the trajectory and is terminated at a later stage $N_{j+1} \geq N_j$. That is, *the time horizon of the learning algorithm is gradually extended.*

The termination condition is defined on the system's output deviation $\tilde{y}(k) \in \mathbb{R}^{n_\mathrm{y}}$, cf. (6),

$$h(\tilde{y}(k)) \geq \gamma(k), \tag{32}$$

where $h(\cdot)$ maps the system output at time $k$ to the relevant termination variables, whose critical values are defined by $\gamma(k) \in \mathbb{R}^{n_\gamma}$. If condition (32) is satisfied for some $k$, the trial $j$ is stopped and $N_j := k$.

Assuming that $N_j \geq N_{j-1}$, the input update is performed for the subproblem, $u_j \in \mathbb{R}^{N_j n_\mathrm{u}}$ and $y_j \in \mathbb{R}^{N_j n_\mathrm{y}}$, such that only the effective length of execution is considered. A current estimate of the modeling error $\widehat{d}_j \in \mathbb{R}^{N_j n_\mathrm{x}}$ is obtained from (22)-(23). In addition to the recent input and output, $u_j$ and $y_j$, the estimation takes advantage of the previous estimate $\widehat{d}_{j-1} \in \mathbb{R}^{N_{j-1} n_\mathrm{x}}$ and covariance $P_{j-1} \in \mathbb{R}^{N_{j-1} n_\mathrm{x} \times N_{j-1} n_\mathrm{x}}$. However, because of the extended time horizon $N_j$, these values must be extended for the current estimation using the initial conditions $P_0$ and $\widehat{d}_0$: with $l, m \in \left\{1, 2, \dots, N_j\right\}$,

$${}^{(c)}P_{j-1}^{(l,m)} = \begin{cases} P_{j-1}^{(l,m)} & \text{for } l,m \in \left\{1, \dots, N_{j-1}\right\} \\ P_0^{(l,m)} & \text{otherwise,} \end{cases} \tag{33}$$

where $P^{(l,m)} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ represents the $(l,m)$th entry in $P$ and ${}^{(c)}P$ denotes the adaptation to the

current length $N_j$; similarly,

$$ {}^{(c)}\widehat{d}_{j-1}^{(m)} = \begin{cases} \widehat{d}_{j-1}^{(m)} & \text{for} \quad m \in \{1,\dots,N_{j-1}\}, \\ \widehat{d}_{0}^{(m)} & \text{for} \quad m \in \{(N_{j-1}+1),\dots,N_j\}, \end{cases} \tag{34} $$

where $\widehat{d}^{(m)} \in \mathbb{R}^{n_\mathrm{x}}$. The matrices $G, F, \Omega_{j-1}$, and $M_j$ are adapted analogously. The resulting estimate $\widehat{d}_j$ from the estimation step is used in (28) to calculate the updated input ${}^{(f)}u_{j+1} \in \mathbb{R}^{N_j n_\mathrm{u}}$ for the first part $(f)$ of the trajectory. This input is continued by the last entries of the nominal input; that is,

$$ u_{j+1}^{(m)} = \begin{cases} {}^{(f)}u_{j+1}^{(m)} & \text{for} \quad m \in \{1,\dots,N_j\}, \\ \mathbf{0} & \text{for} \quad m \in \{N_j+1,\dots,N\}, \end{cases} \tag{35} $$

where $u^{(m)}, \mathbf{0} \in \mathbb{R}^{n_\mathrm{u}}$ and $\mathbf{0}$ is a zero vector. The input is applied to the system during the following trial. If the $(j+1)$th trial is stopped earlier again, the input $u_{j+1}$ is cropped accordingly, such that $u_{j+1} \in \mathbb{R}^{N_{j+1} n_\mathrm{u}}$. If $N_{j+1} < N_j$, the algorithm falls back to time horizon $N_{j+1}$ and all variables are cropped correspondingly. Measurement information for times $k$ larger than $N_{j+1}$ (obtained from earlier iterations) is discarded.

This method of extending the horizon not only guarantees the validity of the linearization in Sec. 2.1, but also responds to safety requirements during the learning process by guaranteeing only small deviations from the desired trajectory.

***Design parameters*** This part of the learning algorithm features the following design parameters:

- The termination function $h(\cdot)$ defines critical variables originating from either the nonlinearity of the system equations (1) or from the safety requirements. The choice of $h(\cdot)$ is very specific to the problem under consideration. One example is to put constraints on the states that introduce the nonlinearity to the system.
- The bound $\gamma(k)$ may vary along the trajectory. Its size is crucial for the convergence of the learning. When $\gamma(k)$ is too small, the learning might never reach the final length $N$, since the system is not able to achieve these small deviations everywhere along the trajectory; if $\gamma(k)$ is too large, it may lead the system into regions where the linearization approximation is not accurate enough.

Even though most papers on ILC are based on a linear system representation, the issue of staying in the region where the linearization is an acceptable approximation, has not been a focus of consideration, see e.g. [32]. The idea of gradually extending the trial time is novel to our approach.

Examples for reasonable termination conditions are given in Sec. 6.5 for the quadrocopter experiments. Sec. 6.5 also shows an example where $N_{j+1} < N_j$. Note, however, that the majority of experiments shown in Sec. 6 were performed without a termination condition and still showed learning convergence.

## 2.5 Summary of the Algorithm

The proposed learning algorithm requires a dynamic model of the physical system under consideration (see Fig. 1). This nominal model serves two main purposes: (i) given a desired trajectory, it is used to obtain an initial guess of the feed-forward input, and (ii) it provides the direction for feed-forward corrections in the input update step. Thus, the nominal model needs to approximate the system dynamics (in the proximity of the desired trajectory) to first order. Note that any physical system, including systems with underlying feedback control, can be considered as long as a nominal model is available. Ideally, the desired trajectory is feasible with respect to dynamics, input and state constraints of the physical system.

The learning algorithm consists of several preparation steps that are performed offline prior to experiment, and an iterative correction step executed online after each run of the experiment. We summarize the algorithm as follows:

***Prerequisites*** Derive a dynamics model of the system such that it captures key dynamic effects and relevant input and state constraints.

***Offline preparations***

a) Define a desired trajectory and find the corresponding nominal input based on the dynamics model.
b) Linearize the model about the nominal trajectory, discretize, and build lifted system representation.
c) Choose design parameters of the estimation and update step. Optional: define a termination condition.
d) Calculate the Kalman filter gains $K_j$.
e) Set $j = 1$ and apply the nominal input, i.e. $u_1 = 0$.

***Online refinement through experiments.***

a) Run experiment with $u_j$. Check if termination condition is satisfied.
b) Stop if the termination condition is satisfied or if the task is completed.
c) Update the disturbance estimate $\widehat{d}_j$ based on the measurement $y_j$.
d) Solve the optimization problem using $\widehat{d}_j$ and obtain the next input $u_{j+1}$.
e) Set $j = j+1$, go to (a).

## 2.6 Computational Complexity

Each of the steps of the learning algorithm highlighted above are of different computational complexity. Below, we provide a brief overview on the complexity of the key steps of the algorithm. We distinguish between offline preparation steps and calculations that are performed online after each iteration.

Recall that the length of the desired trajectory (number of discrete time steps) is denoted by $N$, where $N = t_f/\Delta t$ increases when extending the duration of the trajectory or when reducing the sampling time.

***Offline preparations*** The offline cost is dominated by the computation of the Kalman filter gains $K_j$ according to (22) for $j = 1, \ldots, J_{\max}$, where $J_{max}$ denotes the total number of iterations. The computational complexity of calculating the Kalman filter gains is of order $\mathcal{O}(J_{\max} N^3 (n_x^3 + n_y^3))$.

***Online refinement*** The online cost comprises two steps: updating the disturbance estimate according to (23) and solving the optimization problem (28). The disturbance estimate update requires $\mathcal{O}(N^2 n_x n_y)$ arithmetic operations. The optimization problem has the form of an inequality-constrained linear or quadratic program, see Sec. 2.3. Problems of this type can be solved by interior-point methods. The complexity is given by the total number of Newton steps required for the solution of the optimization problem multiplied by the cost of one Newton step. Under mild assumptions, [36] shows that the worst-case number of Newton steps is of order $\mathcal{O}(N_c \log N_c)$ (and $\mathcal{O}(\sqrt{N_c})$ for a particular choice of parameters), where $N_c$ is the total number of constraints. The cost of one Newton iteration is a polynomial function of the problem dimensions. For (28), the total number of constraints is of order $\mathcal{O}(N(n_c + n_x + n_u))$ for the 1- and $\infty-$norm, and of order $\mathcal{O}(N n_c)$ for the 2-norm (see Sec. A.1), where $n_c$ denotes the number of constrained quantities. For completeness, the number of decision variables in the optimization problem is $\mathcal{O}(N n_u)$ for the 2- and $\infty-$norm and $\mathcal{O}(N(2n_u + n_x))$ for the 1-norm. In practice, convex optimization problems are tractable for a large number of decision variables and constraints. As an example, cf. [36], a linear program with "hundreds of variables and thousands of constraints" can be solved "on a small desktop computer, in a matter of seconds." In Sec. 6.7, we provide specific computations times for the quadrocopter example.

## 3. Quadrocopter Dynamics and Constraints

We now apply the iterative learning algorithm to quadrotor vehicles, with the objective of precisely tracking trajectories that are defined in the vertical plane. Considering two-dimensional trajectories only allows for the use of a more concise quadrocopter model (for both, trajectory generation and learning), while the corresponding learning results still highlight the key characteristics of the proposed algorithm, cf. Sec. 6. The subsequent derivations generalize to 3D trajectories and are equally tractable for this class of problems, see Sec. 6.7.

A two-dimensional model of the quadrocopter dynamics is derived from first principles under the assumption that the out-of-plane dynamics, including vehicle yaw, are stabilized separately (Fig. 2). The equations of motion that define the evolution of horizontal position $y$, the vertical position $z$ and the quadrocopter's roll angle $\phi$ are

$$\ddot{z} = (f_a + f_b + f_c + f_d)\cos\phi - g \tag{36}$$

$$\ddot{y} = -(f_a + f_b + f_c + f_d)\sin\phi \tag{37}$$

$$I_{xx}\ddot{\phi} = m l (f_a - f_c), \tag{38}$$

where $m$ denotes the mass of the vehicle, $g$ represents the gravitational constant, $l$ is the distance from the center of mass of the vehicle to a propeller, $I_{xx}$ is the moment of inertia about the out-of-plane principal axis, and $f_a$ and $f_c$ are the thrust forces produced by the two in-plane rotors

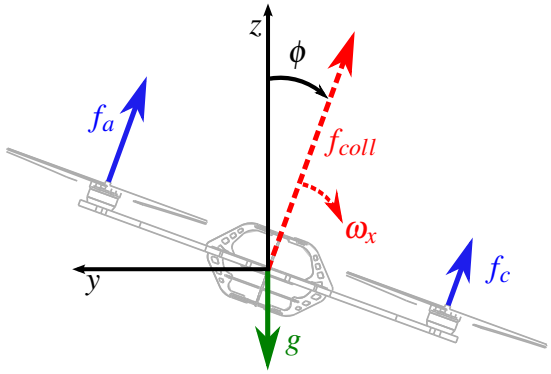



**Figure 2.** Schematic drawing of a quadrocopter moving in the vertical yz-plane with relevant coordinates ($y$, $z$, and $\phi$) and control inputs ($f_{coll}$ and $\omega_x$).

normalized by the mass of the vehicle (Fig. 3). The mass-normalized forces of the other two rotors, $f_b$ and $f_d$, are used to stabilize out-of-plane motion and are nominally set to the average of $f_a$ and $f_c$,

$$f_b = f_d = (f_a + f_c)/2. \tag{39}$$

In the two-dimensional scenario, the inputs to the quadrocopter are the collective acceleration produced by the four motors,

$$f_{coll} = f_a + f_b + f_c + f_d = 2(f_a + f_c), \tag{40}$$

and the roll rate $\omega_x$, cf. Fig. 2. The resulting dynamics are

$$\begin{aligned} \ddot{z} &= f_{coll}\cos\phi - g \\ \ddot{y} &= -f_{coll}\sin\phi \\ \dot{\phi} &= \omega_x, \end{aligned} \tag{41}$$

where $\mathrm{x} = (y, \dot{y}, z, \dot{z}, \phi)$ and $\mathrm{u} = (f_{coll}, \omega_x)$ in the framework of (1). Consequently, the feed-forward input corrections of the learning step are applied at the level of thrust and rate. In (41), we assume that the roll rate $\dot{\phi}$ can be controlled directly. In reality, an underlying high-bandwidth controller on board of the vehicle tracks the commanded rates using feedback from gyroscopes, cf. Sec. 5.2. Because the quadrocopters can achieve exceptionally high angular accelerations (typically on the order of several hundred rad/s$^2$), and thus can respond very quickly to changes in the desired rotational rate, this is a valid approximation for the learning algorithm and trajectory generation. We also assume that the collective thrust can be changed instantaneously. True thrust dynamics are as fast as the rotational dynamics, with propeller spin-up being faster than spin-down. We can – directly or indirectly – measure all five states; that is $\mathrm{y} = \mathrm{x}$ in (1).

The mass-normalized single motor thrusts $f_a$ and $f_c$ are related to the input u by the following equations:

$$f_a = \frac{1}{4} f_{coll} + \frac{I_{xx}}{2ml}\dot{\omega}_x, \qquad f_c = \frac{1}{4} f_{coll} - \frac{I_{xx}}{2ml}\dot{\omega}_x. \tag{42}$$

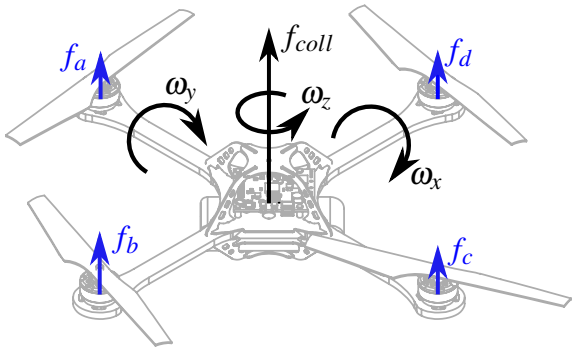



**Figure 3.** The control inputs of the quadrocopter are the body rates $\omega_x$, $\omega_y$, and $\omega_z$, and the collective thrust $f_{coll}$. These inputs are converted by an onboard controller into motor forces $f_i$, $i \in \{a,b,c,d\}$.

The first-principles model presented above neglects numerous aerodynamic effects, such as drag, blade flapping [44], or changes in the angle of attack of the propellers [45], which may have a significant effect on the dynamic behavior of the quadrocopter, especially at high speeds. These effects are difficult to model and are presumed to be compensated by the iterative learning scheme.

The system is subject to several constraints that result from both limited actuator action and limited range of sensor measurements. First, the thrust that each motor can provide is limited by

$$f_{min} \leq f_i \leq f_{max}, \qquad i \in \{a,b,c,d\}\,. \tag{43}$$

Second, due to the motor dynamics, the rate of change of the thrust is also limited:

$$\left|\dot{f}_i\right| \leq \dot{f}_{max}, \qquad i \in \{a,b,c,d\}\,. \tag{44}$$

Equation (43) imposes a constraint on the maximum feasible angular acceleration via (38),

$$\left|\ddot{\phi}\right| \leq \ddot{\phi}_{max}. \tag{45}$$

Furthermore, the onboard rate gyroscopes have a limited measurement range. Taking into account an additional safety margin for the onboard control, a sufficiently conservative constraint on the angular velocity is derived as:

$$\left|\dot{\phi}\right| \leq \dot{\phi}_{max}. \tag{46}$$

Note that the above constraints implicitly constrain the inputs $(f_{coll}, \omega_x)$ via (42). Later, in the input update of the learning algorithm, we consider upper and lower bounds on the input, its derivatives, and on $f_a$ and $f_c$. Note that this constraint definition contains some redundancy. In practice, computation time (cf. Sec. 6.7) can be reduced by constraining only $f_a, f_c$ and $\omega_x$. Derivative constraints are usually satisfied because of the time discretization of the input signal.

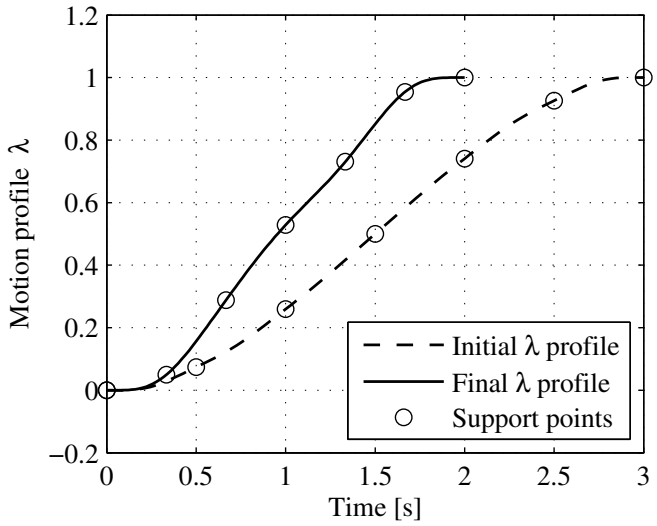



**Figure 4.** Initial and optimized motion profiles. The support points are distributed equally over time and their $\lambda$ values are the optimization variables.

## 4. Trajectory Generation for Quadrocopters

The goal of this section is to plan feasible quadrocopter trajectories (with respect to the constraints introduced in Sec. 3) that track arbitrary user-defined shapes in the vertical plane.

As described in Sec. 2, a feasible state trajectory with its corresponding nominal input is the starting point, and hence a prerequisite of the proposed learning scheme. In this section, we describe an approach for generating feasible state trajectories starting with minimal information on the geometry of the desired state evolution. Once the state trajectory is known as a function of time, the corresponding input is computed from the quadrocopter model in Sec. 3.

This approach is similar to [46–48], where the trajectory generation problem is split into two parts. First, the trajectory's geometry is defined using a set of basis functions such as splines or polynomials. The trajectory's geometry or shape, (hereafter referred to as 'path') does not contain any time information. In the second step, a motion profile is assigned to the path. This profile is chosen such that the resulting time-parameterized trajectory satisfies the feasibility constraints.

In the following, both the path and its motion profile are defined by splines (cf. [49–52]), and the trajectory generation is posed as a constrained optimization problem with the objective of minimizing the trajectory's end time.

The trajectory generation algorithm for two-dimensional quadrocopter maneuvers comprises the following steps:

a) Define the shape of the trajectory, referred to as 'path', by specifying $N_p$ points in the $yz$-plane,

$$\{p^{(1)}, \ldots, p^{(N_p)}\} \qquad \text{with} \quad p^{(i)} \in \mathbb{R}^2, \;\; i \in \{1, 2, \ldots, N_p\}. \tag{47}$$

b) Each point $p^{(i)}$ is assigned a chord-length parameter $\lambda^{(i)}$,

$$\lambda^{(i)} = \frac{i-1}{N_p-1}, \qquad i \in \{1,2,\ldots,N_p\}, \tag{48}$$

resulting in an ordered sequence $(\lambda^{(i)}, p^{(i)})$, which defines the shape of a spline $\mathscr{P}$. In other words, a continuous path in the $yz$-plane is obtained as a mapping from $\lambda$, $\lambda \in [0,1]$, to points in $\mathbb{R}^2$:

$$\mathscr{P}: \quad [0,1] \to \mathbb{R}^2, \tag{49}$$

where $\mathscr{P}(\lambda^{(i)}) = p^{(i)},\ i \in \{1,\ldots,N_p\}$. That is, the spline $\mathscr{P}$ passes through the previously defined points (47). The motion profile along the path is defined by $\lambda(t)$, a monotonically increasing function of time:

$$\lambda: \quad [0,t_f] \to [0,1], \tag{50}$$

where $t_f$ represents the end time of the trajectory. The function $\lambda(t)$ is itself a spline defined by $N_\lambda$ support points $(t^{(k)}, \sigma^{(k)})$, such that

$$\lambda(t^{(k)}) = \sigma^{(k)}, \qquad k \in \{1,\ldots,N_\lambda\}. \tag{51}$$

The first and the last time point are fixed,

$$\begin{aligned} (t^{(1)}, \sigma^{(1)}) &= (0,0) \\ (t^{(N_\lambda)}, \sigma^{(N_\lambda)}) &= (t_f, 1), \end{aligned} \tag{52}$$

and the time instances $t^{(k)},\ k \in \{1,\ldots,N_\lambda\}$, are equally distributed over $[0,t_f]$,

$$t^{(k)} = \frac{k-1}{N_\lambda - 1}\ t_f. \tag{53}$$

The $(N_\lambda - 2)$ interior support points,

$$\Sigma := \{\sigma^{(i)} \,|\, i = 2,\ldots,N_\lambda - 1\}, \tag{54}$$

define the curvature of the $\lambda$ profile and act as the decision variables in the optimization problem described in the next step, see Fig. 4. We can write $\lambda(t) = \lambda(t,\ t_f, \Sigma)$ to make the dependency on the parameters $t_f$ and $\Sigma$ explicit.

c) We find a feasible motion profile by solving the constrained minimization problem:

$$\begin{cases} \min\limits_{\{t_f,\Sigma\}} \ t_f & \\ \text{subject to} & \frac{\partial}{\partial t}\lambda(t,\ t_f,\Sigma) \geq 0, \quad t \in [0,t_f], \\ & \mathscr{P}(\lambda) \text{ feasible.} \end{cases} \tag{55}$$

The objective is to find interior support points $\Sigma$, see (54), such that the corresponding motion profile increases monotonically and yields feasible state trajectories. Feasibility for the special case of the quadrocopter is discussed at the end of this section. An important advantage of this approach is that the number of decision variables in the optimization problem is relatively small. It is equal to the number of interior support points of the motion profile $\lambda$ plus the final time $t_f$, i.e. in total, $N_\lambda - 1$. Moreover, the number of decision variables is independent of the number of path points $N_p$. Generally, the constraints in (55) are non-convex. Thus, the optimization problem lacks any useful characteristics that would guarantee global optimality. Consequently, a solution of (55) is only locally optimal and depends on the initial values of the decision variables $\{t_f, \Sigma\}$.

d) The solution of (55) yields a set of locally optimal interior support points $\Sigma^*$ and a locally optimal end time $t_f^*$. These values uniquely define the motion profile $\lambda(t) = \lambda(t,\, t_f^*, \Sigma^*)$, which, in turn, determines the state trajectories $\mathscr{P}(t) = (y(t), z(t)),\ t \in [0, t_f^*]$, cf. Fig. 4. The nominal inputs are computed via an inversion of the system dynamics (41). Note that time derivatives of $y$, $z$ can be computed analytically since the nominal state trajectories are given by splines; that is, by piece-wise polynomial functions.

In the remainder of this section, we derive constraints that guarantee the feasibility of the trajectories $\mathscr{P}(\lambda)$. First, constraints result from the assumption that the quadrocopter starts and ends in hover; that is, we derive conditions that ensure the continuity of the states and inputs at the start and end of the trajectory. We then consider constraints on the input and its first derivative.

a) *Continuity of the states at the start and end point.*
Hovering is characterized by

$$\dot{y} = \dot{z} = 0, \quad \ddot{y} = \ddot{z} = 0, \quad \phi = \dot{\phi} = \ddot{\phi} = 0. \tag{56}$$

Conditions (56) are satisfied if

$$\frac{\partial^i}{\partial t^i} \lambda(t) \bigg|_{t \in \{0, t_f\}} = 0, \quad i \in \{1,2,3,4\}. \tag{57}$$

b) *Continuity of the inputs at the start and end point.* The condition (57) implicitly guarantees the continuity of the inputs at the start and end of a trajectory; that is, if (57) holds so do the following input conditions:

$$f_{coll}(t)\,\big|_{t \in \{0, t_f\}} = g, \quad \omega_x(t)\,\big|_{t \in \{0, t_f\}} = 0, \tag{58}$$

where $g$ denotes the gravitational constant.

c) *Input constraints.* Input constraints are taken into account directly in the form (43)-(46).

The numeric values of the parameters used in the trajectory generation process are summarized in Tab. 1. In order to leave room for learning, we have chosen restrictive constraint bounds for the trajectory generation.

**Table 1.** Quadrocopter parameters used for the trajectory generation and learning.

| | Trajectory Generation | Learning |
|---|---|---|
| $l$ | 0.17 m | 0.17 m |
| $m$ | 0.468 kg | 0.468 kg |
| $I_{xx}$ | 0.0023 kg m$^2$ | 0.0023 kg m$^2$ |
| $f_{max}$ | 4.5 m/s$^2$ | 5.5 m/s$^2$ |
| $f_{min}$ | 0.4 m/s$^2$ | 0.25 m/s$^2$ |
| $\dot{f}_{max}$ | 27 m/s$^3$ | 51 m/s$^3$ |
| $\dot{\phi}_{max}$ | 22 rad/s | 25 rad/s |
| $\ddot{\phi}_{max}$ | 150 rad/s$^2$ | 200 rad/s$^2$ |

Considering (43)–(46) again, we observe that single thrust input trajectories $f_{a,c}(t)$ must be continuous, whereas their first derivatives $\partial/\partial t\, f_{a,c}(t)$ are allowed to have steps. According to (42), $f_{a,c}(t)$ are functions of $\partial/\partial t\, \omega_x(t)$, which in turn depends on the fourth derivative of the desired trajectory, $\partial^4/\partial t^4 y(t)$ and $\partial^4/\partial t^4 z(t)$. This statement is supported by (41) and is especially relevant in the inversion step of the previous algorithm. Thus, in order for $f_{a,c}(t)$ to be continuous, the fourth derivatives of $y(t)$ and $z(t)$ (with respect to time) must be continuous. This is guaranteed by describing the $\{y,z\}$-trajectories and the $\lambda$-profile by splines that are at least quintic, i.e. of order 5. However, we chose the $\lambda$-spline to be of order 9; this leaves us with 8 free parameters needed to satisfy the 8 constraints given by (57).

We implemented the algorithm in MATLAB, and solved the optimization problem with MATLAB's `fmincon` routine [53].

# 5. Experimental Setup

## 5.1 The Testbed

We have demonstrated the algorithm in question on custom quadrocopters operated in the ETH Flying Machine Arena (FMA), a dedicated testbed for motion control research. The setup is similar to [54, 55]: The space is equipped with an 8-camera motion capture system that, for any properly marked vehicle, provides millimeter-accurate position information and degree-precise attitude data at 200 Hz. The localization data is sent to a PC, which runs the control algorithms (including the iterative learning algorithm), and which in turn sends commands to the quadrocopters. The flying vehicles are based on the Ascending Technologies Hummingbird platform described in [56], with custom wireless communication and central onboard electronics. More details about the test environment can be found in [18] and on the FMA webpage[2].

[2] `www.FlyingMachineArena.org`

### 5.2 Quadrocopter Control

Each vehicle accepts four inputs: three angular rate commands $(\omega_x, \omega_y, \omega_z)$, see Fig. 3, and a mass-normalized collective thrust command $f_{coll}$. These inputs are usually provided by off-board controllers.

For the experiments, we use two different modes for controlling the vehicle. The first mode, referred to below as (C1), is used to stabilize the quadrocopter at the start and end position of the trajectory. It is also used to return the quadrocopter to its initial position before starting a new trial. In this mode, the off-board controller takes desired vehicle positions as an input and closes the loop based on the camera information. The off-board controller calculates all four commands and sends them to the vehicle.

To fly and learn the desired trajectory, we use a different control mode called (C2). In this mode, the iterative learning scheme provides the collective thrust command $f_{coll}$ and the roll rate command $\omega_x$. The inputs $\omega_y$ and $\omega_z$ are used to stabilize the vehicle in the vertical plane. These two inputs are computed by a separate off-board feedback controller that uses position and attitude information of the vehicle.

An onboard controller does high-rate feedback control on the angular rates $(\omega_x, \omega_y, \omega_z)$ using rate gyro information. No feedback is done on the thrust command.

In summary, for (C2) the out-of-plane dynamics are stabilized using camera information; the in-plane dynamics are driven by the feed-forward input signals of the iterative learning scheme, $f_{coll}$ and $\omega_x$. The onboard controller closes the loop on the roll rate input $\omega_x$ to guarantee that the commanded value is actually achieved.

### 5.3 Implementation of Learning Algorithm

In order to test the proposed learning scheme on the real vehicles, we developed a software framework that manages the learning process, and allows for efficient and reliable operation of the quadrocopters. The program manages the operations of flying the quadrocopter to a defined initial position, triggering the learning trajectory, and stabilizing the vehicle at the end of the trajectory. At the core of this setup lies the learning algorithm of Sec. 2. This is implemented in C++ using *boost uBLAS* libraries [57] for matrix operations and the CPLEX optimizer [43] to solve the convex optimization problem.

We designed the overall program as a state machine consisting of two main states, the *WAIT* and *RUN* mode, and a transition state called *AUTOSTART*, cf. Fig. 5. The general procedure is as follows: first, the desired trajectory and the corresponding nominal input are loaded from a *mat* file (generated by the algorithm presented in Sec. 4) and the settings and parameter values of the learning algorithm are read from an *XML* file. Second, based on the nominal model (Sec. 3), the lifted-domain representation and the Kalman gains are computed. After these preliminary steps, the system enters the *WAIT* mode, where the quadrocopter hovers at a given initial point with (C1), see Sec. 5.2. As soon as the user decides to start the experiment, the *AUTOSTART* mode is activated. The system switches to a more aggressive controller of type (C1) for more precise hover performance. The goal is to achieve accurate initial conditions for the learning trajectory. As soon as the quadrocopter satisfies a set of predefined start conditions, a new iteration is triggered. In the experiments presented herein, the start conditions are defined on the translational velocity of the quadrocopter, its attitude, and the rate of change of the

attitude:

$$\begin{aligned} |\dot{x}|, |\dot{y}|, |\dot{z}| &< 0.01 \text{ m/s} \\ |\psi|, |\theta|, |\phi| &< 0.05 \text{ rad} \\ |\dot{\psi}|, |\dot{\theta}|, |\dot{\phi}| &< 0.2 \text{ rad/s}, \end{aligned} \tag{59}$$

where $\psi, \theta, \phi$ are the yaw, pitch and roll angles that entirely define the vehicle attitude (in z-y-x Euler angle notation). The position is not considered because the quadrocopter motion is invariant with respect to the initial position. The actual initial position is simply subtracted from all following position measurements. Once the start conditions (59) are satisfied, the *RUN* mode is activated and the desired trajectory is performed with (C2) using the most recent feedforward inputs of the learning algorithm, see Sec. 5.2. A trajectory is either fully completed or is terminated prematurely if the termination condition of the extending horizon condition is activated and satisfied. After an iteration, the system enters the *WAIT* mode, which performs two tasks simultaneously: returning the quadrocopter to the initial position, and executing the online update step of the learning algorithm (which computes a new input trajectory to be applied in the next iteration).

The program allows the execution of an arbitrary number of iterations and stores all log data.

# 6. Results

This section shows the experimental results of the proposed learning scheme applied to quadrotor vehicles. We consider two different trajectories: a diagonal trajectory and an S-shaped trajectory. Both were generated by the method proposed in Sec. 4. With a set of default learning parameters introduced in Sec. 6.1, the trajectories are learned after five to six iterations and tracked with an accuracy as high as the stochastic noise level of the system allows (Sec. 6.3 and Sec. 6.4). In Sec. 6.5, we apply the extending horizon method to the diagonal trajectory using two different termination conditions. The influence of different learning parameters is shown in Sec. 6.6, where we evaluate the learning performance for different objective function norms, as well as for different input penalty terms and varying noise model parameters. In order to provide insight into the computational cost associated with the approach, we specify the computation times of the different algorithmic steps in Sec. 6.7.

A video of the experiments presented herein is available online[3], and as an electronic appendix to this article.

## 6.1 Default Learning Parameters

As highlighted in Sec. 2.2-2.4, the iterative learning scheme includes several design parameters. The default parameter values used in the subsequent experiments are summarized in Tab. 2.

The applied state scaling factors $s_x$ are empirical values that map the deviations of $\mathrm{x} = (y, \dot{y}, z, \dot{z}, \phi)$ to similar magnitudes. Here, the position was scaled by a factor of 2 and the angle by a factor of 5 as compared to the velocities. The state weighting $s_W$ in Tab. 2 penalizes

[3]The accompanying video is found at `www.tiny.cc/QuadroLearnsTrajectory`

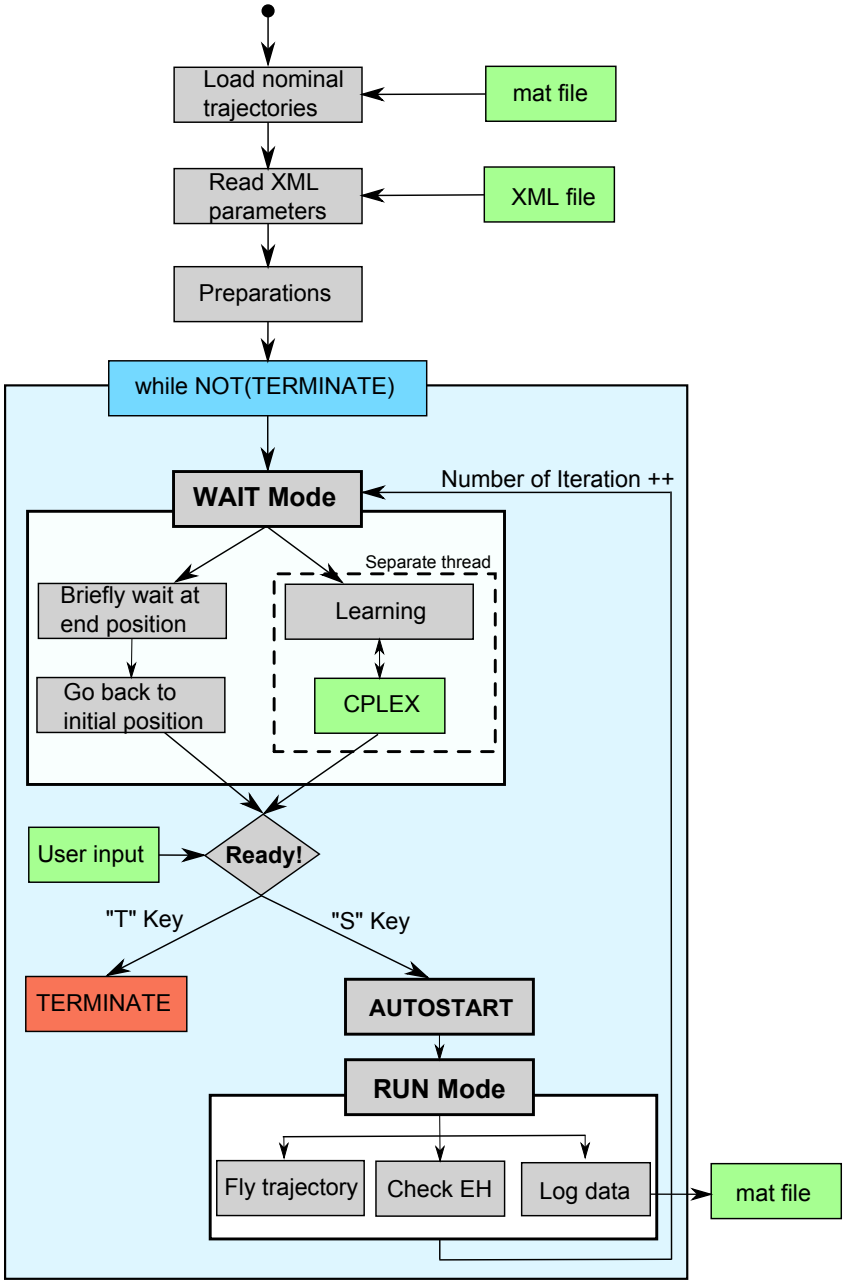



**Figure 5.** Flow diagram of the implemented learning procedure.

deviations of certain components in x more or less than others. We aim for a very precise position tracking. Achieving the desired velocities is less important, however, and angular errors are completely neglected in the input update rule (28). It is reasonable to prioritize between position and angular error. When an imprecise model is used in the trajectory generation, it may be impossible to follow the position trajectories while also tracking the nominal angle trajectory. We use the 2-norm as the default objective function norm. In the objective function, we use a penalty on the input's second derivative, according to (31), to obtain smooth inputs.

**Table 2.** Default learning parameters for the quadrocopter experiments (in SI units).

| | Value | Description |
|---|---|---|
| $\Delta t$ | 0.02 | sampling time in seconds |
| $\varepsilon_j$ | $[0.5, 0.3]$ | process noise variance |
| $\eta$ | 0.05 | measurement noise variance |
| $d_0$ | $0 \in \mathbb{R}^{Nn_x}$ | initial disturbance estimate |
| $P_0$ | $I$ | initial variance of disturbance |
| $\ell$ | 2 | norm of input update |
| $s_x$ | $[2,1,2,1,5]$ | state scaling factors |
| $s_W$ | $[1,0.1,1,0.1,0]$ | state weighting factors |
| $T_W$ | $I$ | trajectory weighting matrix |
| $\alpha$ | 0.08 | 1-norm input penalty factor |
| | $5e-5$ | 2-norm input penalty factor |
| | 0.05 | $\infty$-norm input penalty factor |

As described in Sec. 2.2, we keep the measurement variance constant ($\eta_j = \eta$). We choose a larger process noise covariance $\varepsilon_j$ during the first iterations and choose a smaller noise covariance as the number of iterations increases. The default value for the first five iterations is $\varepsilon_j = 0.5,\ j \in \{1,2,3,4,5\}$. For all subsequent iterations, $j > 5$, $\varepsilon_j = 0.3$ is used.

The choice of parameters is discussed in detail in Sec. 6.6, where experimental results are shown for various parameter combinations.

## 6.2 Learning Performance Measure

In order to quantitatively evaluate and compare the learning performance of different iterations and experiments, we introduce a weighted error function, which reflects the objective function (28) of the learning algorithm (not considering the input penalty term):

$$e_{w,j} = \left\| S_y y_j \right\|_\ell, \qquad \ell = \{1,2,\infty\}, \tag{60}$$

where $y_j$ denotes the lifted-domain output vector (12), whose entries are the *measured* deviations from the reference output trajectory. The matrix $S_y$ weights the output such that it best reflects the defined learning objective (28). For this experiment, all states $\mathrm{x} = (y, \dot{y}, z, \dot{z}, \phi)$ are measured (cf. Sec. 3) and $S_y = S$ is chosen, cf. (28). Note that $y_j$ denotes the lifted output vector, while $y$ is the horizontal position of the quadrocopter. The weighted error is obtained by scaling and weighting the measured output error by the same scaling and weighting matrices that were used in the objective function of the learning routine. In addition, the same norm $\ell$ as in the input update (28) is used in (60), reflecting the main objective of the learning scheme. In the following, we also refer to the weighted error (60) as 'weighted state error'.

This type of performance measure, however, depends on the norm $\ell$. In order to compare the learning performance for different choices of $\ell$ (Sec. 6.6), we introduce another intuitive

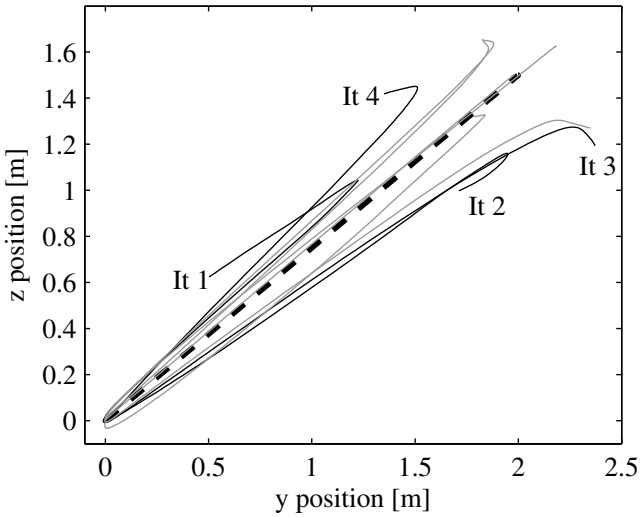



**Figure 6.** *Experiment 1:* Learning a diagonal trajectory. The quadrocopter position in the *yz*-plane is depicted for different iterations. The dashed black line shows the desired trajectory. The trajectories of iterations 1-4 are drawn in black, iterations 5-10 are shown in grey color.

performance measure for the quadrocopter experiments. We use the average position error along a trajectory:

$$e_{pos,j} = \frac{1}{N}\sum_{k=1}^{N}\sqrt{\Delta y(k)^2 + \Delta z(k)^2}, \tag{61}$$

where $\Delta y(k) \in \mathbb{R}$ denotes the deviation of the quadrocopter's horizontal position (from the desired trajectory) at the discrete time $k$ and, similarly, $\Delta z(k) \in \mathbb{R}$ is the vertical deviation. Since $e_{pos}$ is independent of the objective function's norm, it is used to compare the learning performance of the algorithm for different objective function norms in Sec. 6.6.

### 6.3 Experiment 1: Diagonal Trajectory

The desired trajectory of the first experiment is a diagonal motion in the *yz*-plane, see dashed black line in Fig. 6. Challenging for this and all subsequent motions is the coupling of the inputs. Both inputs, $f_{coll}$ and $\omega_x$, have an influence on both quadrocopter coordinates, the horizontal and vertical position. Second, the quadrocopter is required to hover at the beginning and the end of the trajectory. That is, the acceleration and de-acceleration phases at the beginning and end of the trajectory must be learned precisely.

In the first iteration, we apply the nominal input (obtained from the method presented in Sec. 4) to the quadrocopter. As depicted in Fig. 6, the resulting trajectory is far off. Despite the large initial discrepancy, the vehicle learns to track the reference trajectory over the next four iterations. Fig. 7 shows the corresponding evolution of the tracking error and Fig. 8 highlights the convergence of the feed-forward input corrections. Although the feed-forward input converges, the corresponding trajectories in Fig. 6 vary around the desired trajectory, resulting in non-zero tracking errors (Fig. 7). These variations reflect an important characteristic of feed-forward based learning approaches. The feed-forward input that is adapted during the

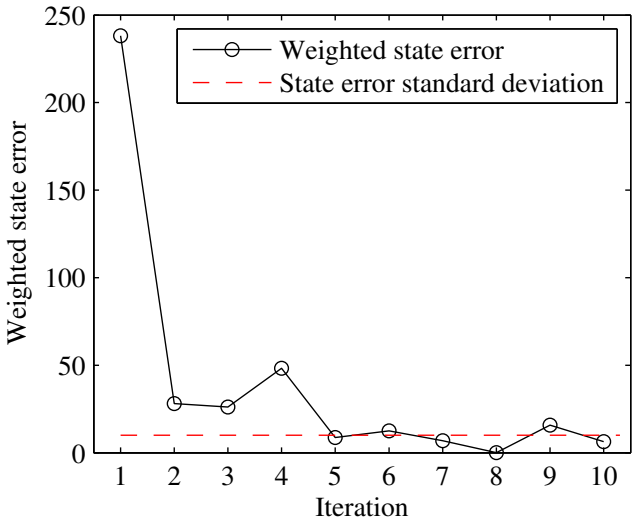



**Figure 7.** *Experiment 1:* Error convergence for the diagonal trajectory. The error is computed according to (60). The dashed line illustrates the standard deviation of the tracking error when applying the same diagonal-trajectory input to the vehicle and observing the variations in the performed trajectories. It can be viewed as a measure of the noise level in the experimental setup.

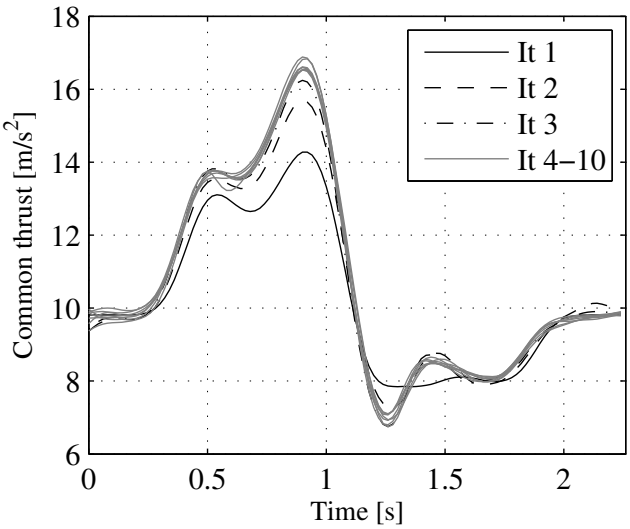



**Figure 8.** *Experiment 1:* The thrust input converges for an increasing number of executions of the diagonal trajectory. The roll rate input converges similarly.

learning process (see Fig. 8) is only able to compensate for repetitive disturbances while any non-repetitive noise directly affects the tracking performance. More precisely, the tracking accuracy is lower-bounded by the level of stochastic (i.e. non-repetitive) noise that acts on the system output.

In order to measure the non-repetitive noise level of our system, we repeatedly apply the

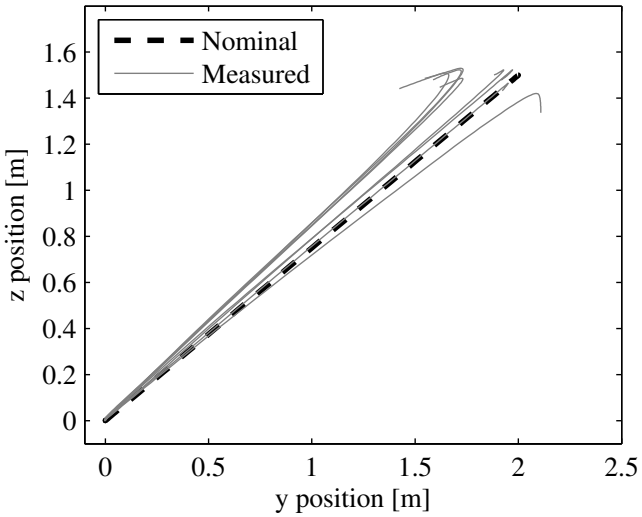



**Figure 9.** Quadrocopter state trajectories for the same feed-forward input (applied repeatedly to the vehicle). Note that the system output varies for identical inputs due to non-repetitive noise acting on the system.

same diagonal-trajectory input and observe the variations in the output. Fig. 9 shows the corresponding state trajectories. We calculate the respective tracking errors and compute their standard deviation. This value characterizes the system noise level and serves as a lower bound for the achievable tracking error, illustrated in Fig. 8 by the dashed horizontal line.

The tracking error in Fig. 8 reaches magnitudes that are in the range of the non-repetitive variability of the system. Consequently, the visible variations of the output trajectories (Fig. 6, grey solid lines) are due to non-repetitive noise and are comparable in size to Fig. 9. We conclude that the learning algorithm is able to effectively compensate for repetitive disturbances and achieves the best possible tracking performance for the given overall system setup.

Moreover, this experiment proves the robustness of the proposed learning algorithm to modeling errors (reflected by the large initial tracking error). Based on the simplified model of Sec. 3, the algorithm is able to learn the desired trajectory in a few iterations.

Statistical information for the proposed learning scheme (including mean and variance of the errors in Fig. 7) are presented in Sec. 6.6 and derived from performing the same learning experiment several times. In Sec. 7, we discuss possibilities to decrease the system noise level and consequently improve the tracking performance.

### 6.4 Experiment 2: S-shaped Trajectory

The proposed framework enables us to define and learn arbitrary trajectories in the vertical plane. In the second experiment, we consider an S-shaped trajectory, cf. Fig. 10 dashed line. The experimental results show the same system characteristics as discussed in Sec. 6.3 and are summarized in Fig. 10-12: Starting from a poor initial performance, the tracking accuracy improves quickly, but is bounded by the system's inherent stochastic variability. Moreover, despite the two outliers at iteration 5 and 10, the feed-forward corrections converge (Fig. 12) to values that best compensates for the occurring repetitive disturbances. The Kalman filter,

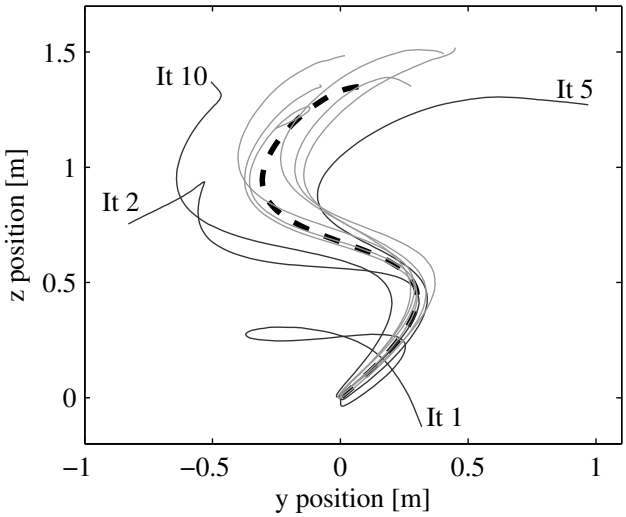



**Figure 10.** *Experiment 2:* Learning an S-shaped trajectory. The quadrocopter position in the *yz*-plane is depicted for different iterations. The black dashed line shows the desired trajectory. The trajectories of iterations 1,2,5,10 are drawn in black, iterations 3,4,6-9 are shown in grey color.

which provides the disturbance estimate, handles outliers effectively by averaging them out rather than over-adapting. The outliers may be caused by the more challenging motor actuation required for the S-shaped trajectory, which implies larger changes and change rates in the single motor thrusts. In brief, the experiment highlights the algorithm's robustness to outliers caused by non-repetitive noise.

## 6.5 Experiment 3: Extending Horizon

An additional component of the presented learning algorithm is the extending horizon feature. This feature terminates a trial as soon as a predefined termination condition is satisfied, cf. Sec. 2.4, and is a means of guaranteeing that the actual motion stays close to the desired trajectory. The following types of termination conditions (TC) are tested in experiments:

- *TC1:* Terminate the current trial if

$$|\Delta z| \geq 0.1\,\mathrm{m}. \tag{62}$$

- *TC2:* Terminate the current trial if

$$\sqrt{(\Delta y)^2 + (\Delta z)^2} \geq 0.2\,\mathrm{m}. \tag{63}$$

The same diagonal reference trajectory as in Sec. 6.3 is considered again. Fig. 13 and Fig. 15 show learning results using (62) and (63), respectively. In both examples, the horizon is gradually extended until, in iteration 5, the entire trajectory is flown within the prescribed bound. Fig. 14 and Fig. 16 show the corresponding error trajectories. Note that in the second

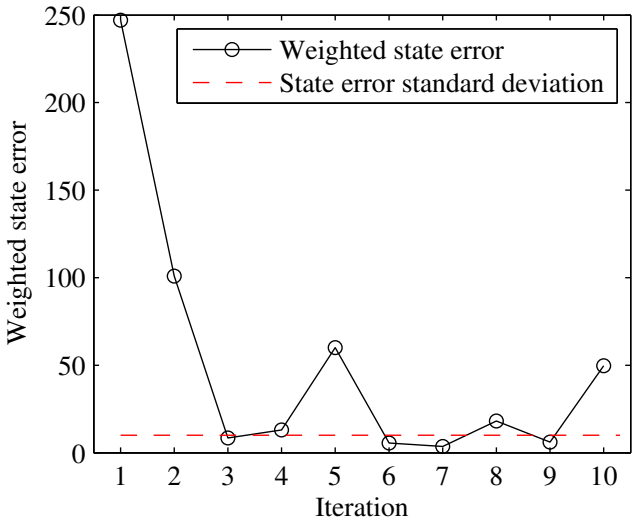



**Figure 11.** *Experiment 2:* Error convergence for the S-shaped trajectory. The error is computed according to (60). The dashed line illustrates the standard deviation of the tracking error when applying the same S-shaped trajectory input to the vehicle and observing the variations in the performed trajectories. It can be viewed as a measure of the noise level in the experimental setup.

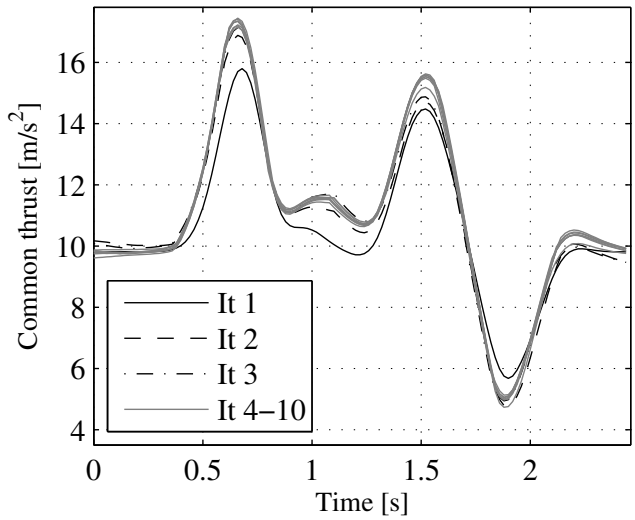



**Figure 12.** *Experiment 2:* The input trajectories converge for an increasing number of executions of the S-shaped trajectory. The roll rate input converges similarly.

experiment (Fig. 15 and Fig. 16), the time horizon of iteration 3 is shorter than the one of iteration 2. In this case, the online learning update is performed only for the first part of the trajectory, until the termination time of iteration 3, and all measurement information for larger times (from earlier iterations) is discarded. For the particular bounds (62) and (63), extending horizon learning takes a similar amount of iterations to converge as learning without a termina-

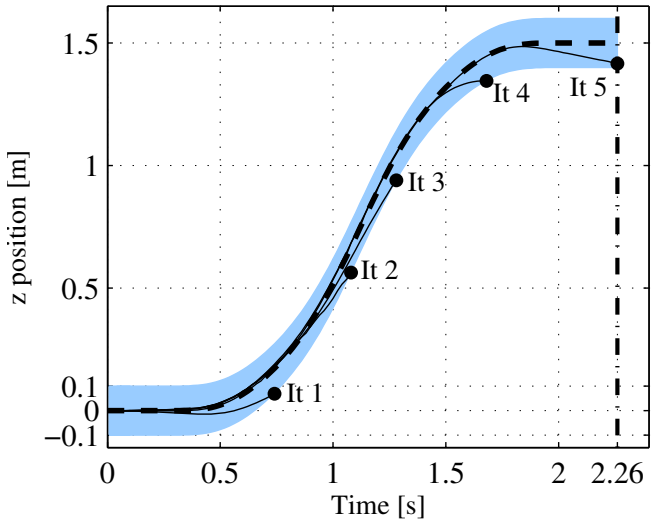



**Figure 13.** *Experiment 3, TC1:* Learning of the diagonal trajectory with extending horizon and termination condition: $|\Delta z| \geq 0.1\,\text{m}$. The learning horizon is gradually increased from iteration to iteration. The black dashed line depicts the desired trajectory of the vertical quadrocopter position over time, the shaded area represents the extending horizon bound, and the black dots show the end of an executed trajectory.

tion condition, cf. Sec. 6.3. If smaller bounds are used, however, the learning may take many more iterations or may not even converge. This is particularly the case if the bounds are smaller than the noise level of the system, cf. Sec. 6.3.

## 6.6 Influence of Different Learning Parameters

We now discuss the influence of different parameter settings on the learning performance. We use the diagonal trajectory as desired trajectory throughout this section, allowing us to compare the subsequent experiments with previous results in Sec. 6.3 and 6.5. We also give more insight into the typical characteristics of the proposed learning scheme, and attempt to justify the choice of the default parameters (Tab. 2).

Unless otherwise stated, the parameter values given in Tab. 2 apply.

***Choice of input penalty factor*** $\alpha$ When solving the optimization problem (28) with $\alpha = 0$, the feed-forward corrections we obtain are highly non-smooth and jitter within the allowed bounds (43)-(46). This effect is observed even in noise-free simulations. One reason may be the discretization that is inherent in the lifted model used in the optimization problem. Fig. 17 shows the input trajectories that are obtained in experiments after one execution of the learning step with $\alpha = 0$. Comparing Fig. 8 ($\alpha = 5e^{-5}$) to Fig. 17 illustrates the positive effect of adding a penalty on the second derivative of the input (31). Smooth inputs are particularly beneficial because they increase the repeatability of the experiment, and thus the effectiveness of the learning. The system noise level is usually much higher when driving a system by fast changing inputs.

For the 2-norm optimization problem, the smallest $\alpha$ value still yielding smooth inputs is $\alpha = 5e^{-5}$ (cf. Tab. 2). For larger values, the learning performance may be corrupted as more

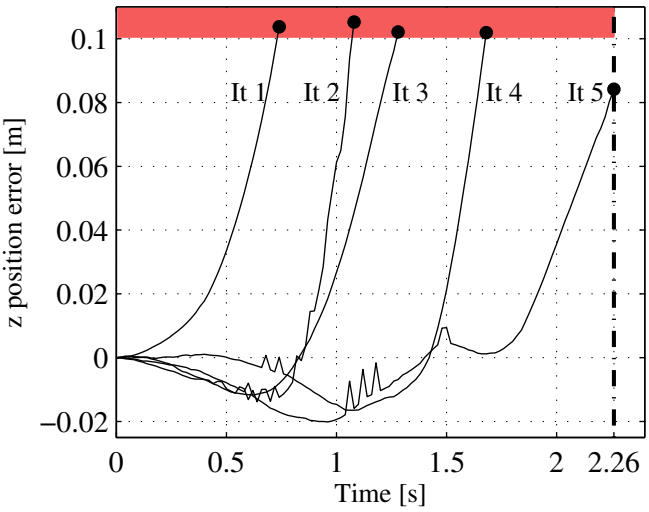



**Figure 14.** *Experiment 3, TC1:* Learning of the diagonal trajectory with extending horizon and termination condition: $|\Delta z| \geq 0.1\,\mathrm{m}$. The solid black lines illustrate the $z$-position error and the black dots show the end of an iteration. The shaded area represents the termination condition (62). The vertical dashed-dotted line marks the end of the desired trajectory.

emphasis is put on minimizing the input's second derivative than on learning to track the desired trajectory. Smaller values of $\alpha$ result in non-smooth inputs, which have a fatal effect on the learning performance. Fig. 18 shows the statistic error convergence for different values of $\alpha$.

***Choice of input penalty matrix*** $D$ The matrix $D$ is used to penalize either the input or its approximate first or second derivative, cf. (28) and Sec. 2.3. When performing and learning the diagonal trajectory with the quadrocopter, it is possible to obtain smooth inputs with either of those choices. Interestingly, all three types of input penalty terms result in similar learning performances, as illustrated in Fig. 19. It is, however, reasonable to penalize the input's second derivative since jittering corresponds to large absolute values of the second derivative.

***Choice of input update norm*** For the objective function (28), three different norms $\ell \in \{1, 2, \infty\}$ can be chosen. The diagonal trajectory is learned successfully by all three input update rules as illustrated in Fig. 20. In order to compare the learning performance of different norms, we used the average position error of the quadrocopter defined by (61). All three norms show fast error convergence.

***Choice of noise covariances*** The performance of the disturbance estimation has a large influence on the learning behavior. Lying at the heart of the Kalman equations (22), the noise model (19) is characterized mainly by the process variance $\varepsilon_j$ and the measurement variance $\eta$, cf. (24) and (25). Both are assumed to be constant over iterations, i.e. $\varepsilon_j = \varepsilon$. The ratio $\varepsilon/\eta$ expresses the belief in the Kalman filter measurement model versus the process model (19). Since the Kalman filter uses a very simple process model and the motion capture system

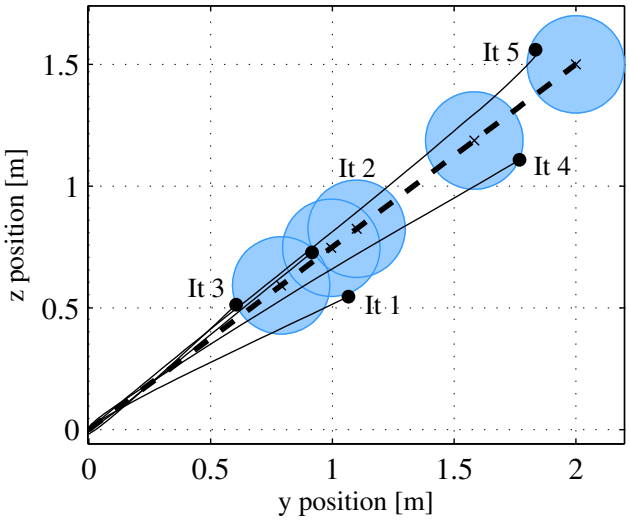



**Figure 15.** *Experiment 3, TC2:* Learning of the diagonal trajectory with extending horizon and termination condition: $\sqrt{(\Delta y)^2 + (\Delta z)^2} \geq 0.2\,\text{m}$. The black dashed line shows the desired trajectory in the $yz$-plane, the shaded disks represent the extending horizon bound, and the black dots show the end of an executed trajectory.

provides accurate measurement data, it is obvious to choose $\varepsilon/\eta \gg 1$. The larger the ratio $\varepsilon/\eta$, the more emphasis is placed on the measurements. Moreover, a large value of $\varepsilon$ allows the estimated disturbance to change rapidly. Therefore, a large ratio $\varepsilon/\eta$ should result in a fast convergence of the disturbance estimate and, as a consequence, of the tracking error. As expected, the larger the ratio $\varepsilon/\eta$, the faster the error converges, cf. Fig. 21.

### 6.7 Computation Times

The objective of this section is to relate the theoretic complexity analysis (Sec. 2.6) to real computation times and develop an intuition for the computational cost of the algorithm. As an example, we use the diagonal trajectory of Sec. 6.3. The trajectory comprises $N = 113$ discrete steps ($t_f = 2.26\,\text{s}$ and $\Delta t = 0.02\,\text{s}$). We recall that the number of inputs is $n_u = 2$, the number of states is $n_x = 5$, and the number of constrained quantities is $n_c = 6$, where for each constrained quantity an upper and lower bound is defined. We run the algorithm on a standard desktop PC (Windows 7, 64-bit; quad-core processor with 2.8 GHz; 4 GB RAM) for $J_{max} = 10$ iterations and solve the optimization problem with CPLEX Version 11.2. As summarized in Tab. 3, the computation times are in the range of seconds, even for solving the optimization problem with hundreds of decision variables and thousands of constraints.

## 7. Advantages & Limitations

The experiments in the previous section demonstrated the effectiveness of the proposed learning. The tracking error was reduced to a level defined by the non-repetitive noise in the system,

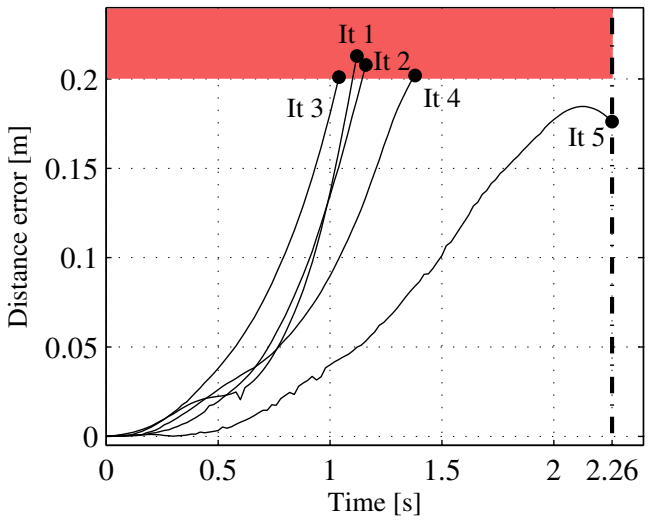



**Figure 16.** *Experiment 3, TC2:* Learning of the diagonal trajectory with extending horizon and termination condition: : $\sqrt{(\Delta y)^2 + (\Delta z)^2} \geq 0.2\,\text{m}$. The black solid lines illustrate the distance error evolution and the black dots show the end of an iteration. The shaded area represents the termination condition (63). The vertical dashed-dotted line marks the end of the desired trajectory.

**Table 3.** Computation times in seconds for the diagonal trajectory.

| | 1-norm | 2-norm | ∞-norm |
|---|---|---|---|
| ***Offline preparation*** | | | |
| Kalman gains | 1.5 | 1.5 | 1.5 |
| ***Online refinement*** | | | |
| Estimate update | 0.028 | 0.028 | 0.028 |
| Input update / Optimization | 1.34 | 0.17 | 0.87 |
| Number of decision variables | 1018 | 226 | 228 |
| Number of constraints | 2938 | 1356 | 2938 |

while any repetitive disturbances were compensated for by proper acausal input corrections. The learning algorithm proved to be robust to the choice of learning parameters, cf. Sec. 6.6. The separation of disturbance estimation and input update allowed for an intuitive interpretation and tuning of the learning algorithm parameters.

Limitations of the approach are caused by: (i) the prerequisites that must be fulfilled for applying the proposed learning approach, and (ii) the requested quality of the final tracking performance. Prerequisites of the approach are a model of the system dynamics and a method for generating feasible trajectories given the model and constraints. Refer to Sec. 2.5 for more

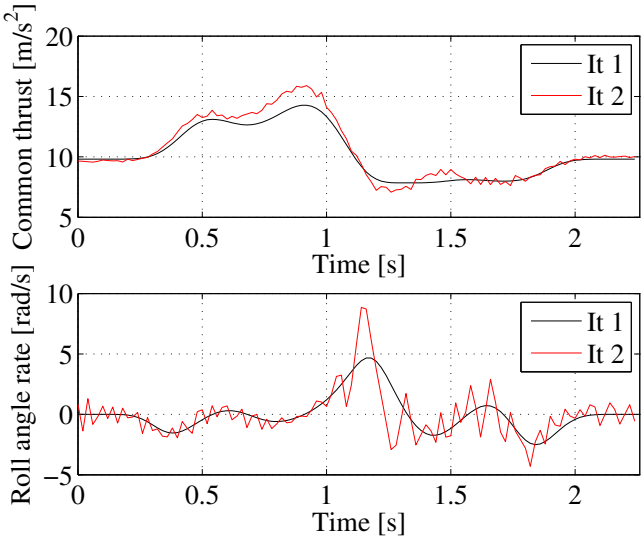



**Figure 17.** *Iteration 1:* The nominal input trajectory. *Iteration 2:* Input trajectory obtained from the input update (28) after the first iteration. No input penalty term was active, i.e. $\alpha = 0$ in (28). The jittering in the inputs is significant.

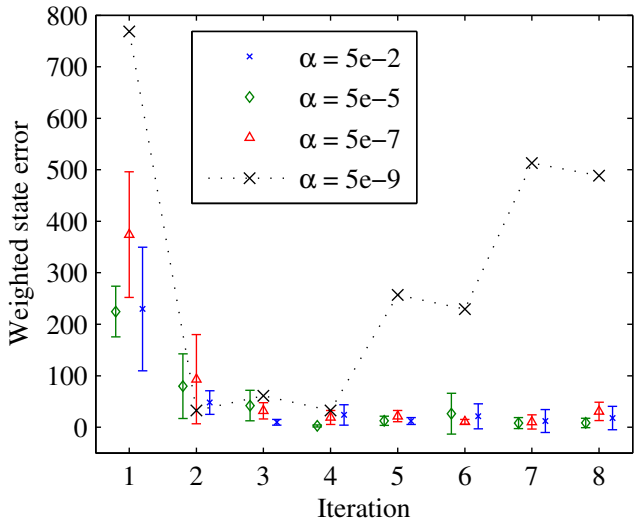



**Figure 18.** Learning performance for different input penalty factors $\alpha$. The error is computed according to (60). The figure shows the average error and its standard deviation obtained from five independent learning experiments. Using very small $\alpha$ values leads to jittering in the input which corrupts the learning performance, see dotted line.

details.

The tracking performance that can be achieved with the proposed method depends on the level of noise that corrupts the system output. Non-repetitive noise cannot be compensated for by the proposed feed-forward learning strategy and directly affects the tracking performance. A measure of the system noise level is the variance of the system output when repeatedly applying

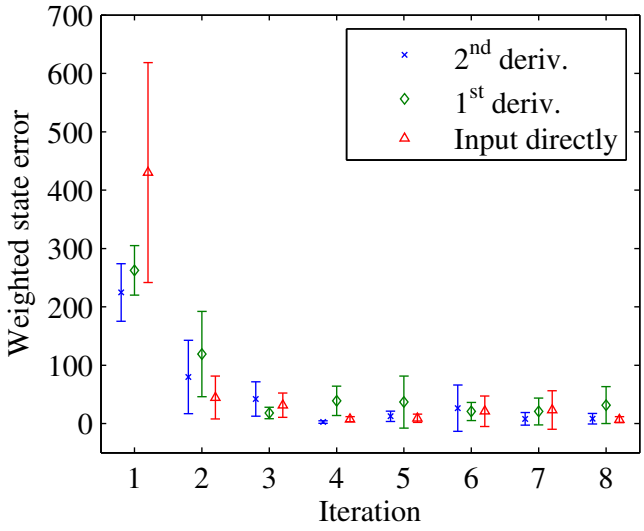



**Figure 19.** Learning performance for different choices of input penalty matrices $D$ in (28). The error is computed according to (60). For each iteration, we show the average error and its standard deviation obtained from five independent learning experiments. Penalizing the second derivative of the input results in small average and standard deviation values.

the same input. In order to increase the repeatability of the system and decrease the effect of non-repetitive noise, one may choose to introduce underlying feedback loops, cf. [24,30,33,34].

For the quadrocopter example, we defined the system inputs on the level of thrusts and rates, which implied a high level of non-repetitive noise resulting mainly from imperfect initial conditions, unpredictable motor dynamics, and environmental noise like wind gusts during flight. Experiments in the FMA have shown that the measured trajectories of a quadrocopter differ significantly, even if the same input is applied, cf. Fig. 9. The variability of the observed trajectories increases towards the end of the trajectory, which is caused by the fact that the entire trajectory is flown without feedback on position or attitude. Angle errors at the beginning of the trajectory, for instance, influence all subsequent angle values and inevitably add up over an iteration. Thus, achieving an accurate initial state is crucial. To this end, we introduced start conditions (see *AUTOSTART* mode in Sec. 5.3), which allow the quadrocopter to start an iteration only if the initial state lies in the bounds (59). These start conditions significantly improve the repeatability of the system, cf. Fig. 22 and Fig. 23, and are essential for the experimental investigations in Sec. 6.

The quadrocopter experiment can be viewed as a proof-of-concept example, where thrust and roll rate serve as inputs for two reasons: (i) for the sake of simplicity (the nominal model is straight-forward to derive), and (ii) to show the limiting case where no global measurements from the cameras are used during a trajectory execution. To obtain a more versatile learning scheme for the quadrocopters and to lower the effect of non-repetitive noise on the system output, a next step may be to close feedback loops on position and attitude. The input and corresponding feed-forward corrections may then be defined on the level of position and attitude. Experiments in the FMA have shown that for this setup, the trajectory variations, when repeatedly applying the same input, lie within one to two centimeters.

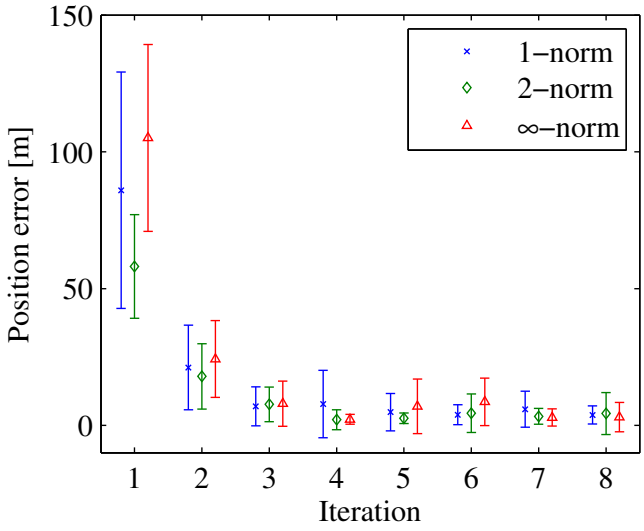



**Figure 20.** Learning performance for different objective function norms $\ell$. Shown is the position error as defined by (61). For each iteration, we show the average error and its standard deviation obtained from ten independent learning experiments. All norms show reasonable convergence behavior. We observe fast convergence and small standard deviations for the 1- and 2-norm with increasing number of iterations.

In practice, we expect the proposed learning scheme to be used for quadrotor vehicles that have access to (possibly rare) position and/or attitude measurements (for example from GPS), and that use this information for feedback. As mentioned in Sec. 1.1, feed-forward adaptation is especially beneficial if feedback is available with low rate only. In this case, feedback is not able to react fast enough to allow for precise tracking but can still be used as correcting action in a feed-forward based approach. Such a setup may be used for practical applications including the ones stated in Sec. 1.1.

## 8. Conclusions

This paper presents an optimization-based iterative learning approach for trajectory tracking. Optimality is achieved in both the estimation of the recurring disturbance and the following input update step, which optimally compensates for the disturbance with an updated feed-forward input signal. While the first method is borrowed from classical control theory, the latter originates from mathematical optimization theory and uses a computationally efficient state-of-the art convex optimization solver. Input and state constraints are explicitly taken into account. Depending on the problem under consideration, the overall learning behavior can be controlled by changing the noise characteristics in the estimation step or the optimization objective, or by assigning different weights on different states and different parts of the trajectory.

The approach was successfully applied to quadrotor vehicles. It has been shown that trajectory tracking can be achieved by pure feed-forward adaptation of the input signal. The accuracy of the tracking is limited by the level of non-repetitive noise. In future research, the

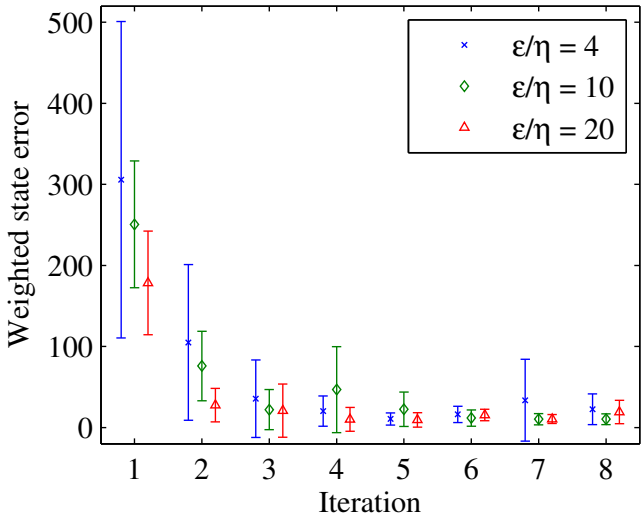



**Figure 21.** Learning performance for different noise settings. The values $\varepsilon$ and $\eta$ are constant over iterations. The error is computed according to (60). For each iteration, we show the average error and its standard deviation obtained from five independent learning experiments. The larger the ratio $\varepsilon/\eta$, the faster the error convergence and the smaller the standard deviation.

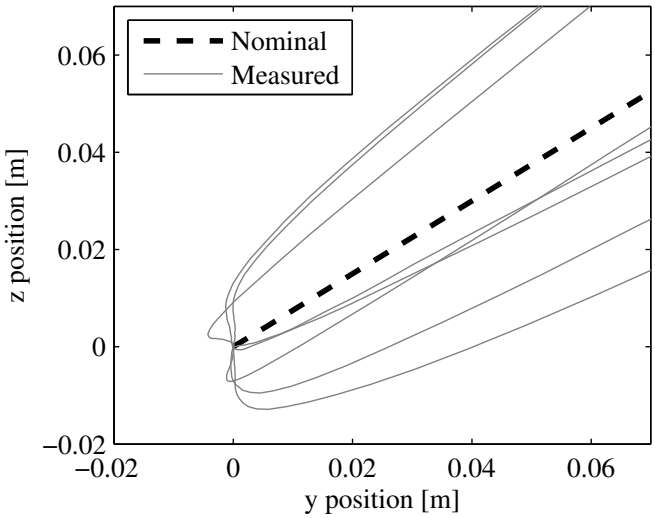



**Figure 22.** Measured quadrocopter position at the start of the diagonal trajectory when not using an *AUTOSTART* mode, cf. Sec. 5.3. The entire trajectory is influenced by the initial state.

final tracking accuracy can be increased by incorporating feedback from position and/or attitude measurements (refer to Sec. 7). Furthermore, the approach is equally feasible for trajectories in three dimensions, especially when one considers that none of the calculations must be done in real time. In the 3D case, the number of inputs and constrains doubles and the number of states increases from five to nine.

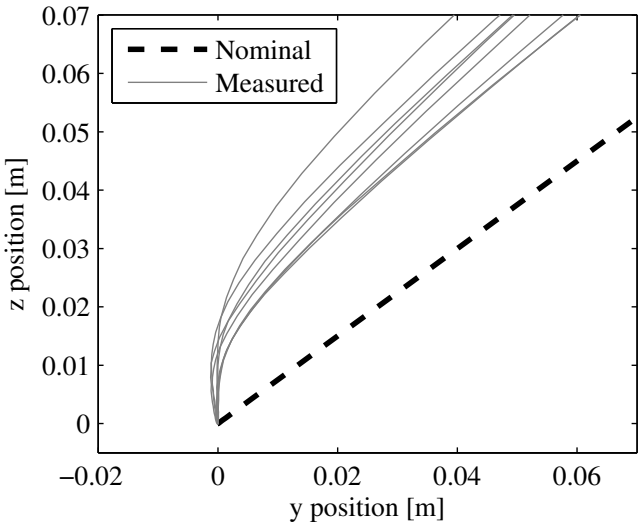



**Figure 23.** Measured quadrocopter position at the start of the diagonal trajectory for more restrictive start tolerance values enforced by the *AUTOSTART* mode, cf. Fig. 5. This start procedure increases the repeatability of the system.

# A. Appendix

## A.1 Input Update as Convex Optimization Problem

The update law defined in (28) can be transformed into a convex optimization problem of the form:

$$\min_{z} \left( \frac{1}{2} z^T V z + v^T z \right) \qquad \text{subject to} \quad Wz \leq w, \tag{64}$$

where $z$ represents the vector of decision variables and $V$ is a symmetric positive semi-definite matrix. The number of constraints is defined by the length of $w$. We call (64) a 'linear program' if $V$ is zero and a 'quadratic program' otherwise, cf. [36].

For simplicity, we consider $\alpha = 0$ in the following derivations. However, similar arguments can be made for arbitrary $\alpha > 0$. In case of norms $\|\cdot\|_\ell$, $\ell \in \{1, \infty\}$, which are inherently nonlinear, non-quadratic functions, (28) is re-formulated by extending the original vector of decision variables $u_{j+1}$ and adding additional inequality constraints. Thus, in case of the 1-norm, the objective function in (28) is replaced by

$$\min_{u_{j+1},\, e} \mathbb{I}^T e \qquad \text{subject to} \quad -e \leq S\left(F u_{j+1} + \widehat{d}_j\right) \leq e\,, \tag{65}$$

where $e \in \mathbb{R}^{Nn_\mathrm{x}}$ and $\mathbb{I}$ represent a vector of ones, $\mathbb{I} = [1, 1, 1, \ldots]^T \in \mathbb{R}^{Nn_\mathrm{x}}$. Similarly, for the maximum norm, the extended equation reads as

$$\min_{u_{j+1},\, e} e \qquad \text{subject to} \quad -e\,\mathbb{I} \leq S\left(F u_{j+1} + \widehat{d}_j\right) \leq e\,\mathbb{I} \tag{66}$$

with $e \in \mathbb{R}$. In both cases, the constraints in (28) must still be satisfied. The 2-norm results in the following objective function:

$$\min_{u_{j+1}} \left(Fu_{j+1} + \widehat{d}_j\right)^T S^T S \left(Fu_{j+1} + \widehat{d}_j\right). \tag{67}$$

## Acknowledgements

This research was funded in part by the Swiss National Science Foundation (SNSF). The authors thank the anonymous reviewers for their thoughtful comments.

## References

[1] Q.-L. Zhou, Y. Zhang, Y.-H. Qu, and C.-A. Rabbath, "Dead reckoning and Kalman filter design for trajectory tracking of a quadrotor UAV," in *Proceedings of the IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA)*, 2010, pp. 119–124.

[2] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 2247–2252.

[3] A. Mokhtari and A. Benallegue, "Dynamic feedback controller of Euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2004, pp. 2359–2366.

[4] Z. Zuo, "Trajectory tracking control design with command-filtered compensation for a quadrotor," *IET Control Theory & Applications*, vol. 4, no. 11, pp. 2343–2355, 2010.

[5] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3255–3260.

[6] D. Lee, T. Burg, D. Dawson, D. Shu, B. Xian, and E. Tatlicioglu, "Robust tracking control of an underactuated quadrotor aerial-robot based on a parametric uncertain model," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2009, pp. 3187–3192.

[7] G. Raffo, M. Ortega, and F. Rubio, "Backstepping/nonlinear $H_\infty$ control for path tracking of a quadrotor unmanned aerial vehicle," in *Proceedings of the American Control Conference (ACC)*, 2008, pp. 3356–3361.

[8] Q.-L. Zhou, Y. Zhang, C.-A. Rabbath, and D. Theilliol, "Design of feedback linearization control and reconfigurable control allocation with application to a quadrotor UAV," in *Proceedings of the Conference on Fault-Tolerant Systems (SysTol)*, 2010, pp. 371–376.

[9] S. Al-Hiddabi, "Quadrotor control using feedback linearization with dynamic extension," in *Proceedings of the International Symposium on Mechatronics and its Applications (ISMA)*, 2009, pp. 1–3.

[10] B. Zhu and W. Huo, "Trajectory linearization control for a quadrotor helicopter," in *Proceedings of the IEEE International Conference on Control and Automation (ICCA)*, 2010, pp. 34 –39.

[11] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Constrained-control of a quadrotor helicopter for trajectory tracking under wind-gust disturbances," in *Proceedings of the IEEE Mediterranean Electrotechnical Conference (MELECON)*, 2010, pp. 1411 –1416.

[12] P. Bauer, B. Kulcsar, and J. Bokor, "Discrete time minimax tracking control with state and disturbance estimation II: Time-varying reference and disturbance signals," in *Proceedings of the Mediterranean Conference Control and Automation (MED)*, 2009, pp. 486 –491.

[13] C. L. Castillo, W. Moreno, and K. P. Valavanis, "Unmanned helicopter waypoint trajectory tracking using model predictive control," in *Proceedings of the Mediterranean Conference on Control Automation*, 2007, pp. 1–8.

[14] S. L. Waslander, G. M. Hoffmann, J. S. Jang, and C. Tomlin, "Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 3712–3717.

[15] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor UAV using neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 50–66, 2010.

[16] C. Nicol, C. J. B. Macnab, and A. Ramirez-Serrano, "Robust adaptive control of a quadrotor helicopter," *Mechatronics*, vol. 21, no. 6, pp. 927–938, 2011.

[17] C. Diao, B. Xian, Q. Yin, W. Zeng, H. Li, and Y. Yang, "A nonlinear adaptive control approach for quadrotor UAVs," in *Proceedings of the Control Conference (ASCC)*, 2011, pp. 223–228.

[18] S. Lupashin, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[19] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Proceedings of the International Symposium on Experimental Robotics*, 2010.

[20] S. Lupashin and R. D'Andrea, "Adaptive open-loop aerobatic maneuvers for quadrocopters," in *Proceedings of the IFAC World Congress*, vol. 18, no. 1, 2011, pp. 2600–2606.

[21] R. Ritz, M. Hehn, S. Lupashin, and R. D'Andrea, "Quadrotor performance benchmarking using optimal control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 5179–5186.

[22] A. P. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European Control Conference (ECC)*, 2009, pp. 1505–1510.

[23] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[24] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[25] H.-S. Ahn, K. L. Moore, and Y. Chen, *Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems (Communications and Control Engineering)*, 1st ed. Springer, 2007.

[26] M. Q. Phan and R. W. Longman, "A mathematical theory of learning control for linear discrete multivariable systems," in *Proceedings of the AIAA/AAS Astrodynamics Conference*, 1988, pp. 740–746.

[27] K. L. Moore, "Multi-loop control approach to designing iterative learning controllers," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 1, 1998, pp. 666–671.

[28] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control using optimal feedback and feedforward actions," *International Journal of Control*, vol. 65, no. 2, pp. 277–293, 1996.

[29] K. S. Lee, J. Lee, I. Chin, J. Choi, and J. H. Lee, "Control of wafer temperature uniformity in rapid thermal processing using an optimal iterative learning control technique," *Industrial and Engineering Chemistry Research*, vol. 40, no. 7, pp. 1661–1672, 2001.

[30] M. Cho, Y. Lee, S. Joo, and K. S. Lee, "Semi-empirical model-based multivariable iterative learning control of an RTP system," *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 3, pp. 430–439, 2005.

[31] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 5, 2001, pp. 4463–4468.

[32] J. K. Rice and M. Verhaegen, "A structured matrix approach to efficient calculation of LQG repetitive learning controllers in the lifted setting," *International Journal of Control*, vol. 83, no. 6, pp. 1265–1276, 2010.

[33] I. Chin, S. J. Qin, K. S. Lee, and M. Cho, "A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection," *Automatica*, vol. 40, no. 11, pp. 1913–1922, 2004.

[34] K. Barton, S. Mishra, and E. Xargay, "Robust iterative learning control: L1 adaptive feedback control in an ILC framework," in *Proceedings of the American Control Conference (ACC)*, 2011, pp. 3663 –3668.

[35] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1731–1736.

[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[37] Y. Chen and C. Wen, *Iterative Learning Control: Convergence, Robustness and Applications (Lecture Notes in Control and Information Sciences)*, 1st ed. Springer, 1999.

[38] B. Bamieh, J. B. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[39] M. Norrloef and S. Gunnarsson, "Time and frequency domain convergence properties in iterative learning control," *International Journal of Control*, vol. 75, no. 14, pp. 1114–1126, 2002.

[40] R. W. Longman, "Iterative learning control and repetitive control for engineering practice," *International Journal of Control*, vol. 73, no. 10, pp. 930–954, 2000.

[41] B. D. O. Anderson and J. B. Moore, *Optimal Filtering (Dover Books on Engineering)*. Dover Publications, 2005.

[42] C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications (Springer Series in Information Sciences)*. Springer, 1998.

[43] *IBM ILOG CPLEX Optimizer*, last accessed feb 08, 2012. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[44] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a quad-rotor robot," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2006.

[45] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3277–3282.

[46] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[47] I. Cowling, O. Yakimenko, J. Whidborne, and A. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *Proceedings of the European Control Conference (ECC)*, 2007, pp. 1–8.

[48] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Proceedings of the Mediterranean Conference on Control and Automation*, 2008, pp. 1258–1263.

[49] *Curve Fitting Toolbox Splines and MATLAB Splines*, last accessed Feb 08, 2012. [Online]. Available: http://www.mathworks.com/help/toolbox/curvefit/f2-10452.html

[50] C. Zhang, N. Wang, and J. Chen, "Trajectory generation for aircraft based on differential flatness and spline theory," in *Proceedings of the International Conference on Information Networking and Automation (ICINA)*, vol. 1, 2010, pp. V1–110–V1–114.

[51] S.-H. Lin and F.-L. Lian, "Study of feasible trajectory generation algorithms for control of planar mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2005, pp. 121–126.

[52] A. Piazzi and C. Guarino Lo Bianco, "Quintic $G^2$-splines for trajectory planning of autonomous vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2000, pp. 198–203.

[53] *The Mathworks Optimization Toolbox*, last accessed Feb 08, 2012. [Online]. Available: http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html

[54] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[55] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP Multiple Micro UAV Testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[56] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 361–366.

[57] *Boost–Basic Linear Algebra Library*, last accessed at 08 Feb 2012. [Online]. Available: http://www.boost.org/doc/libs/1_46_1/libs/numeric/ublas/doc/index.htm

# Paper II

# Iterative Learning of Feed-Forward Corrections for High-Performance Tracking

Fabian L. Mueller · Angela P. Schoellig · Raffaello D'Andrea

**Abstract**

We revisit a recently developed iterative learning algorithm that enables systems to learn from a repeated operation with the goal of achieving high tracking performance of a given trajectory. The learning scheme is based on a coarse dynamics model of the system and uses past measurements to iteratively adapt the feed-forward input signal to the system. The novelty of this work is an identification routine that uses a numerical simulation of the system dynamics to extract the required model information. This allows the learning algorithm to be applied to *any* dynamic system for which a dynamics simulation is available (including systems with underlying feedback loops). The proposed learning algorithm is applied to a quadrocopter system that is guided by a trajectory-following controller. With the identification routine, we are able to extend our previous learning results to three-dimensional quadrocopter motions and achieve significantly higher tracking accuracy due to the underlying feedback control, which accounts for non-repetitive noise.

Published in *Proc. of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2012. DOI 10.1109/IROS.2012.6385647.

## 1. Introduction

Current control systems usually regulate the behavior of dynamic systems by *reacting* to noise and unexpected disturbances. Typically, they are based on a mathematical model of the system dynamics. The performance of this approach is limited by the accuracy of the dynamics model and the causality of the control action that is compensating only for disturbances as they occur. Unfavorable effects of these limitations are observed especially when operating systems in regimes where feedback is not able to react in time and the dynamic behavior is difficult to identify. To achieve high tracking performance in such cases, we propose data-based control approaches that are able to store and interpret information from past executions, and infer the correct actions for future experiments.

We build upon the iterative learning scheme previously presented [1, 2], which relies on a coarse dynamics model of the system under consideration when interpreting past measurements and updating the feed-forward input after each iteration. In contrast to [1, 2], where the system model was derived analytically (e.g. from first principles), this work introduces a numerical identification routine that extracts the required model information from a dynamics simulation. The novel identification routine allows us to apply the learning algorithm to (complex) systems where no analytical model is available, but where a numerical simulation is. This generalized approach comes at a slightly higher computational cost due to the required numerical model extraction. The resulting learning framework is conceptually simple and allows for an acausal correction, which anticipates and compensates for recurring disturbances before they occur. The latter distinguishes our approach from other iterative approaches, which iteratively update the feedback law (cf. [3] and references therein) or adapt the reference input online [4].

The proposed learning algorithm is applied to quadrotor vehicles in the ETH Flying Machine Arena (FMA), cf. [5]. Quadrocopters offer exceptional agility, and complex effects such as aero- and motor-dynamics have a significant impact on the vehicle behavior. These effects are difficult to model but can be compensated for by iterative learning.

The approach presented in this paper can be characterized as an iterative learning control (ILC) technique. ILC became a popular research topic beginning with [6], and has since proven to be a powerful method for high-performance reference tracking. A recent overview of ILC is available in [7] and [8]. Work in [7, 9–11] has shown that ILC can be applied to systems with underlying feedback loops, and [12] first applied ILC to quadrocopter trajectory tracking.

Below we present the learning algorithm, which is introduced as a two-step process of first estimating the unknown repetitive disturbance (Sec. 2.2) and later compensating for it (Sec. 2.3). The numerical system identification routine is presented in Sec. 3. In Sec. 6, we apply the derived learning framework to quadrocopters and present the learning performance in actual experiments. A video of the quadrocopter results is found at `www.tiny.cc/SlalomLearning`. We conclude with a discussion of the approach in Sec. 7 and summarize the results in Sec. 8.

## 2. Iterative Learning

The goal of the proposed learning scheme is to use iterative experiments to teach a dynamic system how to precisely follow a desired trajectory, which is defined by a sequence of output

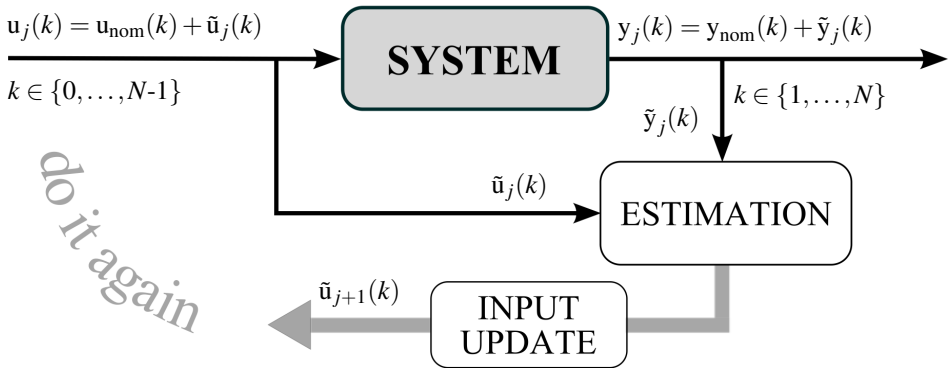



**Figure 1.** The general iterative learning framework considered in this paper: A complete trial $u_j(k), k \in \{0,1,\dots,N-1\}$ is performed at iteration $j$. Based on the output deviation $\tilde{y}_j(k)$, a new input $u_{j+1}(k)$ is calculated and applied during the next trial.

values $y^*(k)$, $k \in \{1,2,\dots,N\}$, with $N < \infty$ being the trial length in discrete time steps. To improve the tracking performance over iterations, the learning algorithm adapts the feed-forward input values after each experiment, where $u_j(k)$, $k \in \{0,1,\dots,N-1\}$, denotes the input of the $j$th experiment. Fig. 1 shows the basic setup. We use discrete-time representations of signals, which may be obtained by sampling the corresponding continuous-time signals.

The learning algorithm requires knowledge of the system's key dynamics around the desired trajectory, cf. Sec. 2.1. The algorithm builds upon this knowledge when exploiting the data from past trials and updating the feed-forward input signal for the next trial. A Kalman filter interprets the measurement of the last trial and incorporates it into the current estimate of the modeling error (Sec. 2.2). The input update step takes the current estimate and returns a more adequate input for the next trial by solving an optimization problem (Sec. 2.3).

The proposed learning scheme can be applied to dynamic systems with underlying feedback loops. In Sec. 4 we show that the tracking accuracy of a quadrocopter guided by a trajectory-following controller can be improved through iterative experiments, cf. Fig. 3.

## 2.1 System Representation

Our learning scheme is a model-based approach in the sense that it incorporates knowledge about the key dynamics of the physical system under consideration. In particular, we consider a mapping $\mathscr{D}$,

$$\mathscr{D} : U \to (Y, C), \tag{1}$$

that relates the input series $\overline{u} = (u(0),\dots,u(N-1)) \in U \subset \mathbb{R}^{Nn_u}$ to the output sequence $\overline{y} = (y(1),\dots,y(N)) \in Y \subset \mathbb{R}^{Nn_y}$ and to the constrained sequence $\overline{c} = (c(1),\dots,c(N)) \in C \subset \mathbb{R}^{Nn_c}$. The vector $c(\cdot) \in \mathbb{R}^{n_c}$ includes all quantities that are subject to constraints. The mapping (1) is not required to reproduce the real system's behavior in the minutest detail but is expected to approximate the dominating dynamics to first order. In Sec. 4.3 we show that a first-principles model of the quadrocopter dynamics is sufficient to derive the mapping (1).

The learning algorithm presupposes an initial guess for the input, hereafter referred to as the *nominal* input $\overline{u}_{nom} \in U$, yielding $(\overline{y}_{nom}, \overline{c}_{nom}) = \mathscr{D}(\overline{u}_{nom})$. We choose $\overline{u}_{nom}$ such that the

resulting $\overline{\mathrm{y}}_{\mathrm{nom}}$ is close to the desired output sequence $\overline{\mathrm{y}}^* = (\mathrm{y}^*(1), \ldots, \mathrm{y}^*(N))$. This is not required but represents a good starting point for the learning process. Subsequently, we consider deviations from the nominal trajectories,

$$\begin{aligned} \tilde{\mathrm{u}}(k) &= \mathrm{u}(k) - \mathrm{u}_{\mathrm{nom}}(k)\,,\ \tilde{\mathrm{y}}(k) = \mathrm{y}(k) - \mathrm{y}_{\mathrm{nom}}(k) \\ \tilde{\mathrm{c}}(k) &= \mathrm{c}(k) - \mathrm{c}_{\mathrm{nom}}(k), \end{aligned} \tag{2}$$

and introduce the lifted vector representation, cf. [13]:

$$\begin{aligned} \mathrm{u} &= [\,\tilde{\mathrm{u}}(0), \tilde{\mathrm{u}}(1), \ldots, \tilde{\mathrm{u}}(N-1)\,]^T \quad \in \mathbb{R}^{Nn_{\mathrm{u}}} \\ \mathrm{y} &= [\,\tilde{\mathrm{y}}(1), \ldots, \tilde{\mathrm{y}}(N)\,]^T \quad \in \mathbb{R}^{Nn_{\mathrm{y}}} \\ \mathrm{c} &= [\,\tilde{\mathrm{c}}(1), \ldots, \tilde{\mathrm{c}}(N)\,]^T \quad \in \mathbb{R}^{Nn_{\mathrm{c}}}. \end{aligned} \tag{3}$$

Note that we use u, y, c (in contrast to $\overline{\mathrm{u}}$, $\overline{\mathrm{y}}$, $\overline{\mathrm{c}}$) to represent *deviations* from the nominal input, output and constraint sequence (namely, $\overline{\mathrm{u}}_{\mathrm{nom}}$, $\overline{\mathrm{y}}_{\mathrm{nom}}$ and $\overline{\mathrm{c}}_{\mathrm{nom}}$).

The key assumption of the learning algorithm is that static linear mappings

$$\mathrm{y} = F\,\mathrm{u}\,, \quad \mathrm{c} = L\,\mathrm{u}\,, \tag{4}$$

with $F \in \mathbb{R}^{Nn_{\mathrm{y}} \times Nn_{\mathrm{u}}}$ and $L \in \mathbb{R}^{Nn_{\mathrm{c}} \times Nn_{\mathrm{u}}}$ can be derived from (1), which capture the main dynamics of the real system along the nominal trajectories $(\overline{\mathrm{u}}_{\mathrm{nom}}, \overline{\mathrm{y}}_{\mathrm{nom}}, \overline{\mathrm{c}}_{\mathrm{nom}})$ by relating the input *deviation* time series $\tilde{\mathrm{u}}(k)$, $k \in \{0, 1, \ldots, N-1\}$, to the corresponding time series of output and constrained quantities *deviations*, $\tilde{\mathrm{y}}(k)$, $\tilde{\mathrm{c}}(k)$, $k \in \{1, 2, \ldots, N\}$.

The mapping (4) is motivated by the fact that any nonlinear dynamics model of the form

$$\dot{\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t), t), \quad \mathrm{y}(t) = \mathrm{g}(\mathrm{x}(t), t), \tag{5}$$

can be written as (4) when time-discretized and linearized around the nominal trajectory, cf. [1]. In Sec. 3 we present an algorithm that identifies the matrices $F$ and $L$ from a dynamics simulation of (1). That is, we extract (4) from a numerical simulation and do not require an explicit analytical representation of (1).

The two steps of the proposed learning algorithm, the disturbance estimation and the input update, fundamentally rely on (4) and are explained in the following.

## 2.2 Disturbance Estimation

The purpose of the estimation step is to estimate a correction vector $\mathrm{d} \in \mathbb{R}^{Nn_{\mathrm{y}}}$ that is added to the first mapping in (4) with the goal of improving the mapping's accuracy. That is, the input-output relation in (4) now reads as

$$\mathrm{y} = F\,\mathrm{u} + \mathrm{d}. \tag{6}$$

The evolution of the learning over a sequence of consecutive trials is modeled by

$$\begin{aligned} \mathrm{d}_{j+1} &= \mathrm{d}_j + \omega_j \\ \mathrm{y}_j &= F\,\mathrm{u}_j + \mathrm{d}_j + \mu_j\,, \end{aligned} \tag{7}$$

where the subscript $j$ indicates the $j$th execution of the desired task, $j \in \{0, 1, \dots\}$. The vector $\mathrm{d}_j$ can be interpreted as repetitive disturbance along the trajectory, which is primarily caused by unmodeled dynamics. The disturbance vector is subject only to slight random changes $\omega_j$ from iteration to iteration. We account for process and measurement noise by adding the random variable $\mu_j$. Both stochastic variables, $\omega_j \sim \mathcal{N}(0, \varepsilon_j I)$ and $\mu_j \sim \mathcal{N}(0, \eta_j I)$, are trial-uncorrelated sequences of zero-mean Gaussian white noise and, moreover, assumed to be independent. The scalars $\mu_j$, $\eta_j$ represent the corresponding variances and $I$ denotes the identity matrix.

We use an iteration-domain Kalman filter, which retains all available information from previous trials (namely the measured output deviations $\{\mathrm{y}_0, \mathrm{y}_1, \dots, \mathrm{y}_j\}$) and calculates an updated estimate $\widehat{\mathrm{d}}_{j+1}$ of the disturbance vector after each iteration based on the relationship (7). Given initial values for the disturbance estimate and its covariance matrix, $\widehat{\mathrm{d}}_0$ and $P_0$, respectively, the disturbance estimate is updated according to

$$\widehat{\mathrm{d}}_{j+1} = \widehat{\mathrm{d}}_j + K_j \left( \mathrm{y}_j - F\,\mathrm{u}_j - \widehat{\mathrm{d}}_j \right), \tag{8}$$

where $K_j$ is the optimal Kalman gain, cf. [1].

### 2.3 Input Update

The learning algorithm is completed by the subsequent input update step. Making use of the information provided by the estimator, cf. Sec. 2.2, we derive a model-based update rule that calculates a new input sequence $\overline{\mathrm{u}}_{j+1} \in \mathbb{R}^{Nn_\mathrm{u}}$ in response to the estimated disturbance $\widehat{\mathrm{d}}_{j+1}$. The goal of the learning algorithm is to find an input *deviation* vector $\mathrm{u}_{j+1}$, such that the system output of the next iteration $\overline{\mathrm{y}}_{j+1}$ follows the desired trajectory $\overline{\mathrm{y}}^*$ as closely as possible. In the context of (2) and (3), this is equivalent to finding input corrections $\mathrm{u}_{j+1}$ that minimize

$$\| \overline{\mathrm{y}}_\mathrm{nom} + \mathrm{y}_{j+1} - \overline{\mathrm{y}}^* \| = \| \check{\mathrm{y}} + \mathrm{y}_{j+1} \|, \tag{9}$$

where $\check{\mathrm{y}}$ is defined as $\check{\mathrm{y}} = \overline{\mathrm{y}}_\mathrm{nom} - \overline{\mathrm{y}}^*$. Since $\mathrm{y}_{j+1}$ is unknown, we use its expected value instead,

$$E\left[ \mathrm{y}_{j+1} \,\middle|\, \mathrm{y}_0, \mathrm{y}_1, \dots, \mathrm{y}_j \right] = F\,\mathrm{u}_{j+1} + \widehat{\mathrm{d}}_{j+1}. \tag{10}$$

The complete optimization problem that constitutes the update step is given as:

$$\begin{aligned} \min_{\mathrm{u}_{j+1}} \; & \left\| S \left( \check{\mathrm{y}} + F\mathrm{u}_{j+1} + \widehat{\mathrm{d}}_{j+1} \right) \right\|_\ell + \alpha \left\| D\,\mathrm{u}_{j+1} \right\|_\ell \\ \text{s.t.} \quad & L\,\mathrm{u}_{j+1} \preceq \mathrm{c}_\mathrm{max}, \end{aligned} \tag{11}$$

where the first term accounts for (9) and $\alpha \geq 0$ weights an additional penalty term that was included into the objective function as a means of directly penalizing the input deviation ($D = I$) or, depending on the choice of $D$, approximations of its derivatives. The original error signal can be scaled and weighted via the diagonal matrix $S \in \mathbb{R}^{Nn_\mathrm{y} \times Nn_\mathrm{y}}$. The vector norm $\ell$, $\ell \in \{1, 2, \infty\}$, in (11) affects the convergence behavior and the result of the learning algorithm. Further, constraints are taken explicitly into account. The lifted vector $\mathrm{c}_\mathrm{max}$ denotes the maximum allowed *deviations* from nominal values $\overline{\mathrm{c}}_\mathrm{nom}$.

The update law (11) is a convex optimization problem, cf. [14], which can be solved very efficiently by existing software tools such as [15]. Moreover, if the optimization problem is feasible, then there exists a local minimum that is globally optimal.

## 3. System Identification

Both the estimation step (Sec. 2.2) and the input update step (Sec. 2.3) rely on information about the system dynamics provided by the mapping (4). The goal of the system identification routine is to obtain the mapping matrices $F$ and $L$ from a numerical simulation of $\mathscr{D}$, cf. (1). Using the nominal input $\bar{\mathrm{u}}_{\mathrm{nom}}$, the simulation provides $(\bar{\mathrm{y}}_{\mathrm{nom}}, \bar{\mathrm{c}}_{\mathrm{nom}}) = \mathscr{D}(\bar{\mathrm{u}}_{\mathrm{nom}})$. The idea of the proposed routine is to identify $F$ and $L$ by running a sequence of simulations with inputs $\bar{\mathrm{u}}$ that differ from the nominal input in exactly one of their (scalar) elements, that is

$$\bar{\mathrm{u}} = \bar{\mathrm{u}}_{\mathrm{nom}} + \Delta u \cdot \mathrm{e}, \quad \Delta u \ll 1\,, \tag{12}$$

where e is a vector of the form $(0,0,\dots,1,0,0,\dots)$ containing exactly one non-zero element. We denote the $i$th element of the lifted input vector by the superscript $(i)$ and write $\bar{\mathrm{u}}^{(i)} \in \mathbb{R}$. The identification routine is summarized in Algorithm 1. By observing the changes in $\bar{\mathrm{y}}$ and $\bar{\mathrm{c}}$ caused by the change in the input according to (12), the matrices $F$ and $L$ are computed, allowing us to use (4) for predictions. However, these predictions are valid only around the nominal trajectory $\bar{\mathrm{y}}_{\mathrm{nom}}$. Thus, if $\bar{\mathrm{y}}_{\mathrm{nom}}$ is far from $\bar{\mathrm{y}}^*$, it might be necessary to *re*-identify the system after some iterations around the current trajectories $(\bar{\mathrm{u}}_j, \bar{\mathrm{y}}_j, \bar{\mathrm{c}}_j)$, $j > 0$, in order to ensure accurate predictions. Note that the mapping $\mathscr{D}$ must be continuous in $\bar{\mathrm{u}}$ to obtain useful approximations for $F$ and $L$.

**Algorithm 1** Identification routine

**Require:** Nominal input $\bar{\mathrm{u}}_{\mathrm{nom}}$ and mapping $\mathscr{D}$.

/∗∗∗ **Preliminary Step** ∗∗∗/
Compute $(\bar{\mathrm{y}}_{\mathrm{nom}}, \bar{\mathrm{c}}_{\mathrm{nom}}) = \mathscr{D}(\bar{\mathrm{u}}_{\mathrm{nom}})$.
/∗∗∗ **Identification Loop** ∗∗∗/
Allocate: $F \in \mathbb{R}^{Nn_{\mathrm{y}} \times Nn_{\mathrm{u}}}$, $L \in \mathbb{R}^{Nn_{\mathrm{c}} \times Nn_{\mathrm{u}}}$
Choose input increment $\Delta u \ll 1$, $\Delta u \in \mathbb{R}$.
**for** $i = 1 : Nn_{\mathrm{u}}$ **do**
  Define simulation inputs: $\bar{\mathrm{u}}_1 = \bar{\mathrm{u}}_2 = \bar{\mathrm{u}}_{\mathrm{nom}}$
  Change $i$th element: $\bar{\mathrm{u}}_1^{(i)} = \bar{\mathrm{u}}_1^{(i)} + \Delta u$
  Change $i$th element: $\bar{\mathrm{u}}_2^{(i)} = \bar{\mathrm{u}}_2^{(i)} - \Delta u$
  **Simulation** $i.1$**:** apply $\bar{\mathrm{u}}_1$, store $(\bar{\mathrm{y}}_1, \bar{\mathrm{c}}_1) = \mathscr{D}(\bar{\mathrm{u}}_1)$
  **Simulation** $i.2$**:** apply $\bar{\mathrm{u}}_2$, store $(\bar{\mathrm{y}}_2, \bar{\mathrm{c}}_2) = \mathscr{D}(\bar{\mathrm{u}}_2)$
  Compute $i$th column of $F$ and $L$:

$$\begin{aligned} F(:,i) &= (\bar{\mathrm{y}}_2 - \bar{\mathrm{y}}_1)/(2\Delta u) \\ L(:,i) &= (\bar{\mathrm{c}}_2 - \bar{\mathrm{c}}_1)/(2\Delta u) \end{aligned} \tag{13}$$

**end for**
**return** Lifted-domain mapping matrices $F$ and $L$.

# 4. Experimental Setup

The iterative learning algorithm is applied to quadrocopter vehicles with the objective of precisely tracking trajectories in the three-dimensional space. The quadrocopters are operated in the ETH Flying Machine Arena (FMA), a dedicated testbed for motion control research. Similar to [16, 17], a motion capture system is used that provides precise position and attitude measurements of the vehicles. The localization data is sent to a PC that runs the control algorithms (including the iterative learning algorithm) and sends commands back to the quadrocopters. More details about the test environment can be found in [18] and on the FMA web page, cf. [5].

## 4.1 Quadrocopter Control

The quadrotor vehicles are controlled by an onboard controller (OBC) and by a trajectory-following controller (TFC) that runs off board. The OBC accepts four inputs, the commanded collective thrust $f_{\text{coll,cmd}}$ and rotational body rates $(\omega_{x,\text{cmd}}, \omega_{y,\text{cmd}}, \omega_{z,\text{cmd}})$, and computes desired motor forces $f_{i,\text{cmd}}$ $i \in \{a,b,c,d\}$ using feedback from rate gyros, cf. Fig. 2. The thrust and body rate commands are provided by the TFC, which takes desired position and yaw angle (and, optionally, corresponding derivatives) as an input, cf. Fig. 3. The TFC closes the loop using position and attitude measurements provided by the motion capture camera system. Refer to [19] for a detailed description of the TFC design.

The TFC is commonly used for trajectory tracking in the FMA. Applications include standard routines such as take-off and landing, as well as research projects like the 'Music in Motion' project, which builds upon the TFC and creates multi-vehicle performances that are designed and synchronized to music (see [5] for more information). We typically operate the TFC in two different modes: (C1) using desired quadrocopter position and yaw angle as inputs (later called *no feed-forward*), and (C2) additionally feeding velocity and acceleration values as inputs (called *with feed-forward*).

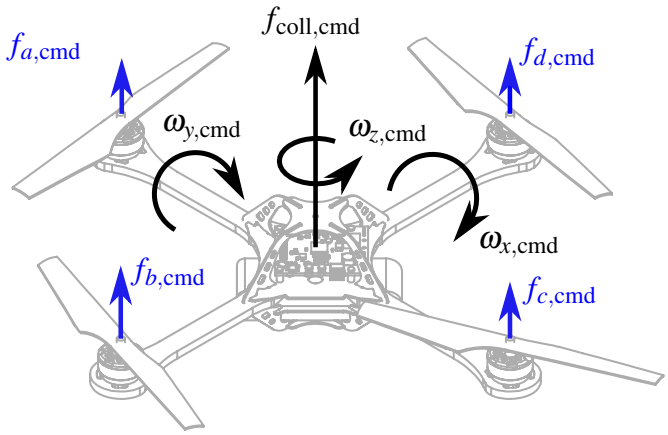



**Figure 2.** The control inputs of the quadrocopter are the body rates $\omega_{x,\text{cmd}}$, $\omega_{y,\text{cmd}}$, and $\omega_{z,\text{cmd}}$ and the collective thrust $f_{\text{coll,cmd}}$. These inputs are converted by an onboard controller into motor forces $f_{i,\text{cmd}}$, $i \in \{a,b,c,d\}$.

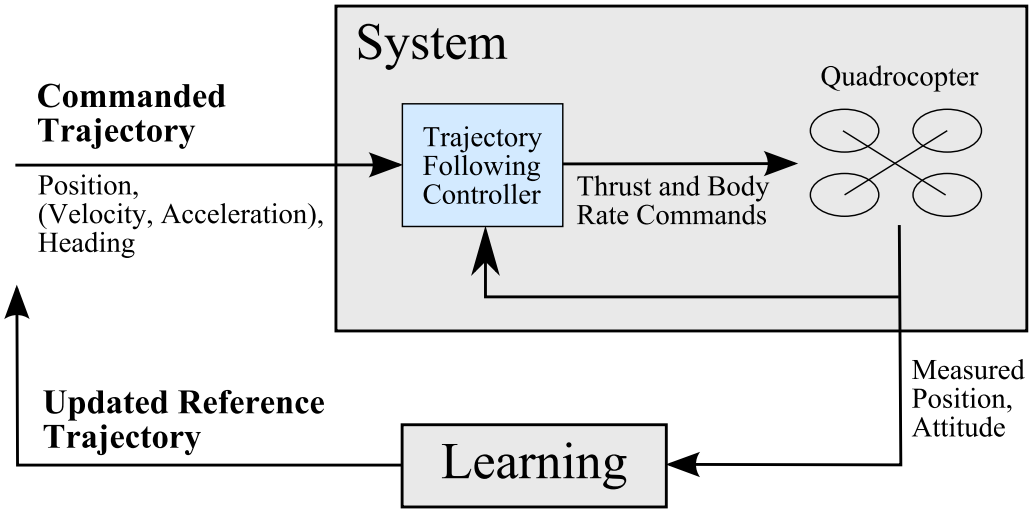



**Figure 3.** We build the iterative learning scheme around a quadrocopter that is controlled by a trajectory following controller.

### 4.2 Applying ILC to Quadrocopters

The iterative learning scheme of Sec. 2 is applied to the quadrocopter system on the level of position and yaw angle input. That is, the input and output in the framework of (1) are

$$\bar{\mathrm{u}} = [x_{\mathrm{cmd}}, y_{\mathrm{cmd}}, z_{\mathrm{cmd}}, \psi_{\mathrm{cmd}}]^T \tag{14}$$

$$\bar{\mathrm{y}} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi]^T, \tag{15}$$

where the quadrocopter's position and velocity are denoted by $(x, y, z)$ and $(\dot{x}, \dot{y}, \dot{z})$, respectively, and $(\phi, \theta, \psi)$ are the vehicle's roll, pitch and yaw angle.

The vehicles are subject to several constraints that result from both limited actuator action and limited range of sensor measurements. First, the thrust that each motor can provide is limited by $f_{\min} \leq f_i \leq f_{\max}$, $i \in \{a, b, c, d\}$. Second, due to the motor dynamics, the rate of change of the thrust is also limited: $\left|\dot{f}_i\right| \leq \dot{f}_{\max}$, $i \in \{a, b, c, d\}$. Furthermore, the onboard rate gyroscopes have a limited measurement range, $|\omega_x|, \left|\omega_y\right|, |\omega_z| \leq \Omega_{\max}$. The constrained quantities are summarized by $\mathrm{c}(k) = (f(k), \Delta f(k), \Omega(k))$ with $f(k) = (f_a(k), \ldots, f_d(k))$, $\Delta f = (\Delta f_a(k), \ldots, \Delta f_d(k))$ and $\Omega(k) = (\omega_x(k), \omega_y(k), \omega_z(k))$, where $\Delta f_i$ represents a numerical approximation of the time derivative of $f_i$. The constraints are taken explicitly into account in the optimization problem (11).

### 4.3 Quadrocopter Dynamics

The simulated quadrocopter dynamics that are at the heart of the identification routine (cf. Sec. 3) neglect all aerodynamic effects, motor dynamics or battery behavior, and aim to reproduce the fundamental dynamic effects only.

The continuous-time equations governing the quadrocopter's dynamics (first-principles model) are described in [19], and the dynamic behavior of

$$\Omega(t) = (\omega_x(t), \omega_y(t), \omega_z(t))$$

is modeled as a first-order system. Given $\Omega$ and $\dot{\Omega}$ at each time step, the motor forces $(f_a, f_b, f_c, f_d)$ are obtained from the rotational quadrocopter dynamics by solving a linear system of equations, see e.g. [18]. Parameters such as the vehicle mass are found in [2].

# 5. Results

This section presents experimental results of the proposed learning scheme applied to quadrocopters. We focus on a particular S-shaped trajectory, for which we show that our learning algorithm significantly improves the tracking performance. While demonstrating the key characteristics of the approach for a single trajectory, the learning scheme has proven to be equally effective for arbitrary 3D trajectories, see video at `www.tiny.cc/SlalomLearning` where various slalom trajectories are learned.

## 5.1 Performance of Trajectory Following Controller (TFC)

We first use the standard TFC described in Sec. 4.1 to track the desired S-shaped trajectory (see dashed line in Fig. 4). The resulting tracking performance of the TFC is depicted in Fig. 4 for both control modes, (C1) and (C2). The input to the TFC is the desired trajectory itself (and corresponding derivatives). The TFC's tracking performance is unsatisfactory for both modes. When repeatedly executing the trajectory, we observe a large repetitive tracking error component and small variations between subsequent executions due to non-repetitive disturbances. Consequently, the overall system behavior, including quadrocopter and TFC, is highly repetitive: repeated executions with the same input result in almost identical output trajectories. Iterative learning schemes are well-suited for such systems, since they are able to learn how to compensate for repetitive error components and thus are able to significantly improve the tracking accuracy. Note that we chose a particularly aggressive S-shaped motion (see velocity values in Fig. 4) to highlight the limitations of pure feedback control.

## 5.2 Tracking Performance Using Iterative Learning

Fig. 10 shows the output trajectories when applying the proposed iterative learning algorithm. After two learning steps, the quadrocopter follows the desired trajectory closely. The tracking error convergence is depicted in Fig. 6, where we consider the weighted error

$$e_{w,j} := \left\| S\left(\check{\mathrm{y}} + \mathrm{y}_j\right) \right\|_2 \tag{16}$$

as a performance measure reflecting the learning objective in (11). Starting from an initial performance determined by the TFC performance, the learning scheme successfully compensates for repetitive errors along the trajectory and, in 5 to 6 iterations, reduces the tracking error (16) to values in the range of the stochastic (i.e. non-repetitive) noise level. The dashed red line in Fig. 6 depicts the standard deviation of the tracking error when applying the same input to the system and observing the variations of the performed trajectories. It can be viewed as a measure of the noise level in the system and represents a lower bound of the achievable tracking accuracy. For an intuitive interpretation, the *average* position tracking error along the trajectory after successful learning is in the range of 2–3 cm. This is also the accuracy that we achieve when hovering the vehicle at a given point.

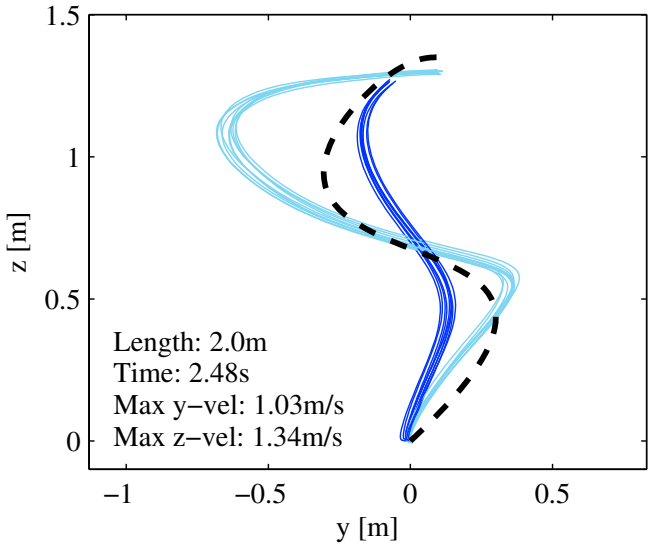



**Figure 4.** Tracking an S-shaped trajectory with the trajectory following controller (TFC). The quadrocopter position in the *yz*-plane is depicted for two different control modes: (C1) the TFC uses the desired position and yaw angle as inputs (dark blue), and (C2) the TFC uses additional velocity and acceleration feed-forward terms as inputs (light blue). The dashed black line shows the desired trajectory. A repeated operation shows that a large proportion of the tracking error is repetitive and only small non-repetitive effects are visible.

As discussed in Sec. 2, the learning scheme iteratively updates the reference input signal sent to the system. Fig. 12 depicts the *y*-position input signals of the initial trial and of iteration 7–9. The input trajectories converge over iterations until their variability is in the range of the stochastic noise level. The final input trajectories, together with the estimated disturbance vector, comprise the knowledge that we gain from the iterative learning process. Fig. 12 shows that the learned reference signal commands the left/right motion sooner (compensating for the delay in the system) and with larger amplitudes (counteracting the system's inherent attenuation).

## 6. Computational Complexity

**Identification Routine.** The number of simulations required to identify the mappings (4) is linear in $N$ and $n_\mathrm{u}$. According to Algorithm 1, we need exactly $2Nn_\mathrm{u}$ simulations. Obviously, the computation time of a single simulation depends on the complexity and implementation of (1). The identification of the trajectory of Sec. 6 takes 93 s on a standard desktop PC (Windows 7, 64-bit; quad-core @ 2.8 GHz, 4 GB RAM).

**Learning Algorithm.** A detailed complexity analysis of the learning algorithm can be found in [2]. The biggest computational cost is associated with the solution of (11), which takes 30 s in the example presented.

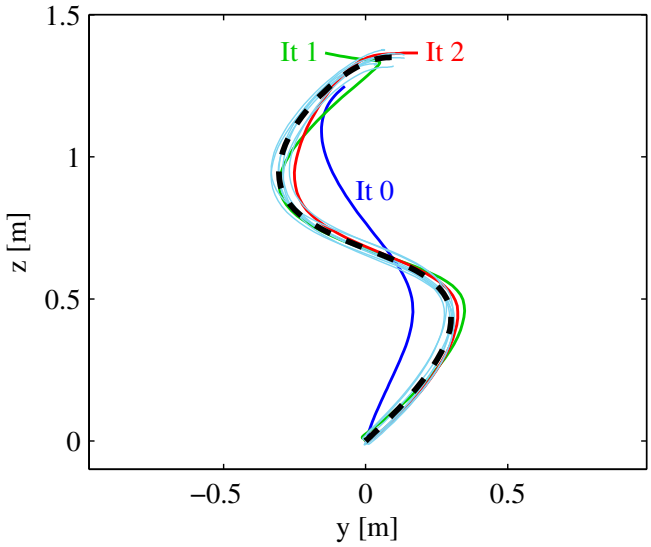



**Figure 5.** Learning an S-shaped trajectory. The quadrocopter position in the *yz*-plane is depicted for different iterations. The dashed black line shows the desired trajectory. The trajectories of iterations 0–2 are drawn in different colors, iterations 3–9 are shown in light blue color.

## 7. Advantages & Limitations

The experimental results presented in the previous section showed that the learning algorithm significantly improves the system's tracking performance. We achieve a high tracking accuracy due to the fact that the learning scheme embraces the quadrocopter including underlying feedback, cf. Fig. 3. This setup results in a highly repetitive system, where the feedback controller compensates for most of the non-repetitive noise. The remaining non-repetitive noise acting on the overall closed-loop system defines a lower bound on the achievable tracking accuracy. Compared to our previous work [2], this bound has been significantly reduced with this setup.

The tracking error convergence speed depends on the prediction quality of the mapping (6), which is based on the identified system dynamics, cf. Sec. 3. However, experimental results suggest that the learning algorithm is very robust to inaccurate mappings. Moreover, after a few executions of the learning step, the system may be *re*-identified around $(\overline{\mathrm{u}}_j, \overline{\mathrm{x}}_j, \overline{\mathrm{y}}_j)$ for some $j > 0$ in order to obtain a more accurate mapping (4). Further, an inaccurate constraint mapping $\mathrm{c} = L\,\mathrm{u}$, cf. (4), may corrupt the learning by either allowing infeasible values or being too restrictive. This can be overcome by estimating an additive correction vector $\mathrm{d_c}$ similar to d in (6); that is, $\mathrm{c} = L\,\mathrm{u} + \mathrm{d_c}$.

## 8. Conclusions

This paper provided a conceptually simple and computationally efficient learning framework for trajectory tracking. The approach is a generalization of our previous work. No analytical system model is required; instead, the algorithm is applicable to any dynamic system, for which a numerical dynamics simulation is available. Because of the acausal learning action

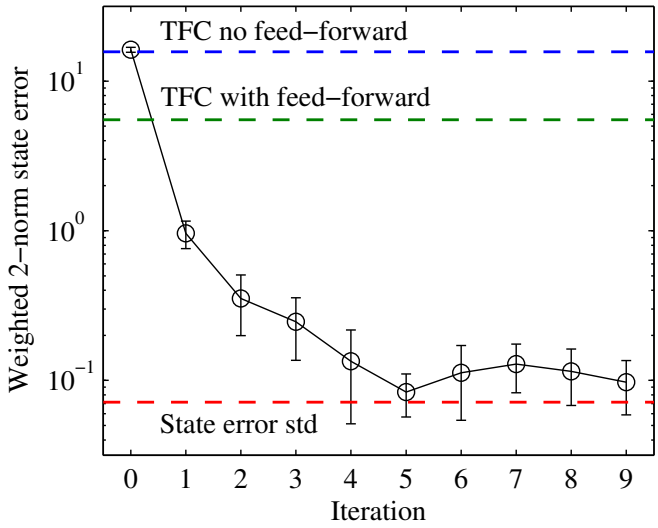



**Figure 6.** Error convergence for the S-shaped trajectory. The error is computed according to (16). The 10 independent learning experiments were performed. The circles and bars show the average error and standard deviation, respectively. The dashed blue and green lines show the average tracking error of the trajectory following controller with and without feed-forward terms. The dashed red line represents the standard deviation of the tracking error when applying the same input repeatedly. It can be viewed as a measure of the noise level in the experimental setup.

that corrects for repetitive disturbances *before* they occur, the final tracking performance of the proposed learning scheme outperforms pure feedback control. By basing the learning approach on a coarse dynamics model of the system, we achieve fast convergence, usually in around 5 iterations. The novelty of this paper is that the dynamics model was derived from a numerical simulation, and that this identification routine, combined with the learning algorithm, were experimentally evaluated on quadrotor vehicles guided by an underlying trajectory-following controller.

## References

[1] A. P. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European Control Conference (ECC)*, 2009, pp. 1505–1510.

[2] A. P. Schoellig, F. L. Mueller, and R. D'Andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, pp. 103–127, 2012.

[3] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference (ACC)*, vol. 1, 2005, pp. 300–306.

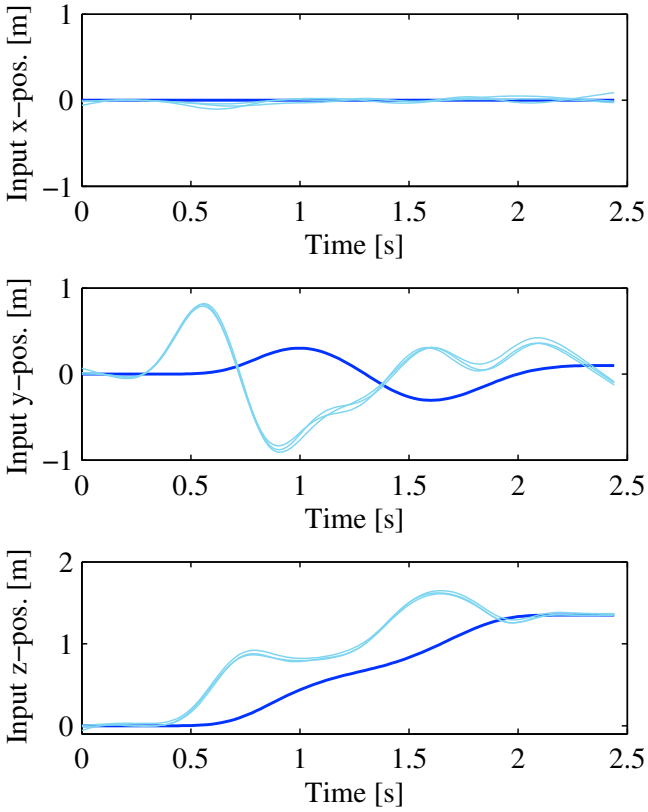



**Figure 7.** Learned *y*-position inputs for the S-shaped trajectory. The initial input corresponds to the desired output trajectory and is drawn in dark blue. The converged learning inputs (iterations 7–9) are shown in light blue color.

[4] M. Kawato, "Feedback-error-learning neural network for supervised motor learning," *Advanced Neural Computers*, vol. 6, no. 3, pp. 365–372, 1990.

[5] "The Flying Machine Arena," www.flyingmachinearena.org, last accessed March 07, 2012.

[6] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[7] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[8] H.-S. Ahn, K. L. Moore, and Y. Chen, *Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems (Communications and Control Engineering)*, 1st ed. Springer, 2007.

[9] I. Chin, S. J. Qin, K. S. Lee, and M. Cho, "A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection," *Automatica*, vol. 40, no. 11, pp. 1913–1922, 2004.

[10] M. Cho, Y. Lee, S. Joo, and K. S. Lee, "Semi-empirical model-based multivariable iterative learning control of an RTP system," *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 3, pp. 430–439, 2005.

[11] K. Barton, S. Mishra, and E. Xargay, "Robust iterative learning control: L1 adaptive feedback control in an ILC framework," in *Proceedings of the American Control Conference (ACC)*, 2011, pp. 3663–3668.

[12] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1731–1736.

[13] B. Bamieh, J. B. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[15] "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, last accessed February 08, 2012.

[16] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[17] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro UAV testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[18] S. Lupashin and R. D'Andrea, "Adaptive fast open-loop maneuvers for quadrocopters," *Autonomous Robots*, vol. 33, pp. 89–102, 2012.

[19] A. P. Schoellig, C. Wiltsche, and R. D'Andrea, "Feed-forward parameter identification for precise periodic quadrocopter motions," in *Proceedings of the American Control Conference (ACC)*, 2012, pp. 4313–4318.

Paper III

# Limited Benefit of Joint Estimation in Multi-Agent Iterative Learning

Angela P. Schoellig · Javier Alonso-Mora · Raffaello D'Andrea

**Abstract**

This paper studies iterative learning in a multi-agent framework, wherein a group of agents simultaneously and repeatedly perform the same task. Assuming similarity between the agents, we investigate whether exchanging information between the agents improves an individual's learning performance. That is, does an individual agent benefit from the experience of the other agents? We consider the multi-agent iterative learning problem as a two-step process of: first, estimating the repetitive disturbance of each agent; and second, correcting for it. We present a comparison of an agent's disturbance estimate in the case of (I) independent estimation, where each agent has access only to its own measurement, and (II) joint estimation, where information of all agents is globally accessible. When the agents are identical and noise comes from measurement only, joint estimation yields a noticeable improvement in performance. However, when process noise is encountered or when the agents have an individual disturbance component, the benefit of joint estimation is negligible.

Published in *Asian Journal of Control,* 2012. DOI: 10.1002/asjc.398.

## 1. Introduction

Exploiting previous experience when repeatedly executing the same task is a logical way to improve future performance in the presence of repetitive, unmodeled disturbances. Iterative learning control (ILC), as first proposed in [1], achieves precise tracking behavior by effectively incorporating past control information (such as applied input signals and measured outputs) when calculating the feedforward control action used in the next iteration, cf. [2, 3]. One way of viewing ILC is as a two-step process of estimation and control: first identifying the unknown repetitive disturbance and later compensating for it [4–9]. LQG-type solutions have been proposed in [10–13], which estimate the tracking error and, based on this result, calculate a new input trajectory by minimizing a quadratic cost function.

While ILC has proven to be successful in a variety of industrial applications (including chemical process control, rotary systems and robotics), we have yet to identify if – and how – ILC schemes can be generalized when facing homogeneous groups of agents or assemblies of similar units (for example, robot arms in an industrial environment, or a fleet of mobile robots in a warehouse [14, 15]). In other words, how can we cope with uncertainties in a multi-agent framework? Is there a benefit of exchanging information between the agents? What kind of information sharing makes sense? Cooperative iterative learning schemes were previously proposed in [16]. Recently, ILC was applied to multi-agent systems, cf. [17], with the goal of achieving formation control. While it has been established that the joint performance of all agents is fundamental to the formation problem, this paper focuses on the potential for individual agents to improve their performance when conducting a task alongside a group of similar agents conducting the same task. Preliminary results were first published in [18]. Analogous questions were previously studied in the context of reinforcement learning, see [19].

The results of our research show that the passing of information between agents has limited benefit for a large class of problems. This conclusion is based upon a comparison of independent learning with a cooperative scheme, where information of all agents is globally accessible to every agent. Similarity between the agents is assured by assuming that they have the same nominal dynamics and share a common iteration-independent disturbance; however, they differ in an additional individual disturbance component that is also constant across iterations. We introduce iteration-dependent noise terms that account for measurement and process noise, and obtain results for two limit cases: (i) pure process noise, and (ii) pure measurement noise. The benefits of information sharing are negligible in (i). For (ii), we observe a greater improvement in performance when a high similarity between the agents is guaranteed. From this point, we are able to deduce the properties of the general mixed-noise case. In short: Individual agents in an ILC framework do not, in most cases, benefit significantly from information sharing when simultaneously learning the same task.

The paper is organized as follows: Section 2 formalizes the multi-agent iterative learning problem and reduces it to a comparison of independent versus joint estimation. Section 3 compares both scenarios and presents the core result of the paper in terms of an upper bound on the performance improvement due to joint estimation. Several numerical examples are presented in Section 4 visualizing the derived analytical results. The work is summarized in Section 5, whereas proofs are partly presented in the appendix, Section A.

# 2. Problem Statement

## 2.1 Motivation

We begin by considering a group of $N$ agents that simultaneously and repeatedly perform the same task. A common way of describing an agent's dynamics during a single run is the so-called lifted system representation [20–22]. For each agent $i \in \mathscr{I} = \{1,2,\dots,N\}$, the input-state relationship is modeled by a static matrix equation,

$$x^i = F^i u^i + d^i\,, \tag{1}$$

which maps a given discrete-time input signal $u^i = \left[u^i(0), u^i(1), \dots, u^i(T)\right]^{\mathrm{T}} \in \mathbb{R}^{(T+1)n_{\mathrm{u}}}$ to the corresponding lifted states $x^i \in \mathbb{R}^{(T+1)n_{\mathrm{x}}}$. In this context, $(T+1)$ samples represent a single iteration and $n_{\mathrm{u}}$ and $n_{\mathrm{x}}$ denote the dimension of the input and state, respectively. The vectors $x^i$ and $u^i$ are defined as the deviation from the desired task trajectory and the corresponding nominal input, see for example [9]. The vector $d^i$ represents an exogenous disturbance constant across iterations, which captures model errors along the trajectory as well as repeating disturbances and nonzero initial conditions [3,23,24]. We include a non-repetitive noise signal $\xi^i_j$ in model (1) to account for process noise, which varies from trial to trial. Introducing the iteration index $j \in \{1,2,\dots\}$, the state in the $j$th trial is given by

$$x^i_j = F^i u^i_j + d^i + \xi^i_j\,, \tag{2}$$

where $\xi^i_j$ is assumed to be zero-mean Gaussian white noise. The vector $d^i$ is viewed as an agent-dependent, normally-distributed random signal.

The agents' output $y^i_j$ is corrupted by measurement noise and similarly represented in the lifted domain,

$$y^i_j = G^i x^i_j + \mu^i_j\,, \tag{3}$$

where $\mu^i_j$ is zero-mean Gaussian white noise. Similar to $x^i_j$ and $u^i_j$, the output $y^i_j$ is interpreted as the deviation from the nominal output trajectory.

Note that (2) and (3) might be the result of linearizing the agent dynamics about a desired task trajectory. Refer to [9, 25] for a more detailed derivation.

In the above context, the goal of the iterative learning algorithm is to make the state $x^i_j$ (that is, the deviation from the desired task trajectory) small or, more precisely, to reduce $x^i_j$ with an increasing number of iterations $j$. The performance of each individual agent is gradually improved by taking into account all information on previous iterations when estimating the disturbance vector $d^i$. As the accuracy of the disturbance estimate increases, a more appropriate open-loop input is determined, thereby compensating for the deficiencies in the modeled dynamics represented by the matrix $F^i$. From $x^i_j$ conclusions can be drawn as to the performance of execution $j$.

We now consider a homogeneous fleet of agents with the same nominal dynamics:

$$F^i = F, \quad G^i = I, \qquad \forall\, i \in \mathscr{I}\,, \tag{4}$$

where $I$ denotes the identity matrix. That is, the state is assumed to be measured directly. Differences between the agents are captured in the disturbance vector $d^i$, which is composed of a common part $d^0$ that is identical for all agents, and an individual part $d^{i,ind}$,

$$d^i = d^0 + d^{i,ind}, \qquad \forall i \in \mathscr{I}. \tag{5}$$

In this context, the question arises: Does an individual agent benefit from sharing information with its companions? To what degree can the disturbance estimate $d^i$ be improved by taking into account the measurements of the other agents?

### 2.2 Simplified Model

Our main objective and central problem is to identify the disturbance $d^i$ for each agent $i$ in the presence of both process and measurement noise. Based on the disturbance estimate, it is possible to find the correcting input $u^i_j$ that best compensates for the repetitive disturbance using a problem-specific optimization criterion, see for example [9]. Importantly, the correcting input $u^i_j$ applied in each iteration is known. Focusing on the estimation problem, we consider a condensed form of the above multi-agent system representation (2)-(3),

$$x^i_j = d^i + \xi^i_j \tag{6}$$

$$y^i_j = x^i_j + \mu^i_j, \tag{7}$$

which features the key noise and disturbance characteristics, but omits the known part $Fu^i_j$ without loss of generality. Equations (6) and (7) are summarized by

$$y^i_j = d^i + \upsilon^i_j, \tag{8}$$

where $\upsilon^i_j = \xi^i_j + \mu^i_j$ captures both process and measurement noise.

Moreover, assuming independence of the single entries in the vectors $d^i$ and $\upsilon^i_j$ and identical noise characteristics, the problem reduces to the scalar case,

$$y^i_j = d^0 + d^{i,ind} + \upsilon^i_j, \tag{9}$$

where all variables are scalar-valued. The probability distributions are given by

$$\begin{aligned} d^0 &\sim \mathscr{N}(0, \alpha) \\ d^{i,ind} &\sim \mathscr{N}(0, \beta) \\ \upsilon^i_j &\sim \mathscr{N}(0, 1), \qquad \alpha, \beta \geq 0, \end{aligned} \tag{10}$$

where all quantities, $\upsilon^i_j$, $i \in \mathscr{I}$, $j \in \{1, 2, \ldots\}$, $d^{i,ind}$, $i \in \mathscr{I}$, and $d^0$, are assumed to be mutually independent. The notation $\mathscr{N}(0, \alpha)$ represents a normal distribution with mean 0 and variance $\alpha$. Note that in (10), the variance of the individual disturbance $d^{i,ind}$ is assumed to be identical for all agents $i \in \mathscr{I}$. Without loss of generality, the variances are normalized such that the variance of $\upsilon^i_j$ is 1. For the variances of the process and measurement noise, this results in

$$\begin{aligned} \xi^i_j &\sim \mathscr{N}(0, \lambda) \\ \mu^i_j &\sim \mathscr{N}(0, 1-\lambda), \qquad 0 \leq \lambda \leq 1, \end{aligned} \tag{11}$$

assuming independence between $\xi_j^i$ and $\mu_j^i$. A value $\lambda = 1$ represents the case of encountering only process noise, whereas $\lambda = 0$ reflects the case where the noise is due to measurement only.

## 2.3 Independent vs. Joint Estimation

As the number of trials and measurements increases, more information about the system is collected, allowing an increasingly accurate estimate of the agents' constant noise terms $d^i,\ i \in \mathscr{I}$. Two limiting approaches might be taken when solving the estimation problem: (I) independent estimation, and (II) joint estimation.

In the case of independent estimation (I), each agent $i$ individually estimates its disturbance $d^i$ taking only its own measurements $y_j^i,\ j \in \{1, 2, \dots\}$, into account. That is, information on the individually obtained measurements is not exchanged between the agents.

In the joint case (II), the acquired measurement data of each agent is made available to all other agents; that is, every agent receives the information of all other agents' measurements. Based on this global knowledge, we can design a joint estimation scheme that exploits the measurements of all agents and provides estimates $d^i$ for every agent $i \in \mathscr{I}$. A vector $D$, reflecting the estimation objective in this case, is defined as: $D = [d^0, d^1, \dots, d^N]^{\mathrm{T}} \in \mathbb{R}^{(N+1)}$. The measurements of all agents in the $j$th trial are combined in $Y_j = [y_j^1, y_j^2, \dots, y_j^N]^{\mathrm{T}}$ and, analogously, the noise vector $V_j = [\upsilon_j^1, \upsilon_j^2, \dots, \upsilon_j^N]^{\mathrm{T}}$ is introduced. Based on this representation, the joint estimation problem can be formulated as a Kalman filter problem, cf. [26, 27]:

$$\begin{aligned} D_j &= D_{j-1} \qquad\qquad \forall\, j \geq 1, \\ Y_j &= H D_j + V_j, \end{aligned} \tag{12}$$

where $H = [\mathbf{0}, I]$ is a matrix with zeros in the first column concatenated with an identity matrix of appropriate dimensions. The Kalman filter returns an unbiased state estimate $\widehat{D}_j$ for $j \geq 1$ that minimizes the error covariance matrix

$$P_j = E\left[(D_j - \widehat{D}_j)(D_j - \widehat{D}_j)^{\mathrm{T}}\right], \tag{13}$$

of trial $j$, taking measurements $Y_m,\ 1 \leq m \leq j$, into account. $E[\cdot]$ denotes the expected value. The initial values are obtained from (10); in particular,

$$\widehat{D}_0 = [0, 0, \dots, 0]^T \tag{14}$$

and the initial covariance matrix $P_0 = [p_0^{(k,l)}],\ k,l \in \mathscr{K} = \{0, 1, \dots, N\}$, is

$$P_0 = E\left[D_0 D_0^{\mathrm{T}}\right] \tag{15}$$

with

$$p_0^{(k,l)} = E\left[d^k d^l\right] = E\left[\left(d^0 + d^{k,ind}\right)\left(d^0 + d^{l,ind}\right)\right],$$

where $d^{0,ind} = 0$. Recalling the mutual independence of $d^0$ and $d^{i,ind}$ for all $i \in \mathscr{I}$, the initial covariance is given by

$$p_0^{(k,l)} = \begin{cases} \alpha + \beta & \text{for} \quad k = l \geq 1 \\ \alpha & \text{otherwise}. \end{cases} \tag{16}$$

The variances $\alpha$ and $\beta$, which reflect the original noise characteristics (10), serve as a initial values. Note that the above derivations do not place further assumptions or restrictions on how information is shared between agents: the information $y_j^i$ of each agent is available to every other agent. In other words, we are investigating the ideal case of centralized, joint estimation within an optimal filtering context.

Equally important is that the independent estimation problem (I) is just a special case of the cooperative framework (II) with $N = 1$.

In both cases, (I) and (II), the variance of an individual's disturbance estimate at iteration $j$ is given by

$$E\left[(d^i - \widehat{d}^i_j)^2\right] = p_j^{(i,i)} = p_j^{(1,1)}, \qquad \forall\, i \in \mathscr{I}, \tag{17}$$

where $\widehat{D}_j = [\widehat{d}^i_j]$, $i \in \mathscr{I}$, and $P_j = [p_j^{(k,l)}]$, $k,l \in \mathscr{K}$. The variance is identical for all agents, since for each agent the same assumptions on the dynamics (9) and the initial noise characteristics (10) are made. The variance of an individual's disturbance (17) indicates the quality of the disturbance estimate. In the general case, (2)-(3), this value influences the effectiveness of the disturbance compensation, since the input update rule of the ILC algorithm is based on the current estimate $\widehat{d}^i_j$; for example by a relation as follows, see [9]:

$$u^i_{j+1} = \arg\min_u \left\| F^i u + \widehat{d}^i_j \right\|. \tag{18}$$

Below, we distinguish between the individual disturbance variance $p_j^{(1,1)}$ in case of joint and independent estimation, where the latter is given when evaluating $p_j^{(1,1)}$ for $N = 1$, i.e.

$$p_j^{(1,1)}\Big|_{N=1}. \tag{19}$$

Thus, the initial question can be reformulated: To what degree does joint estimation benefit the individual learning of an agent?

## 3. Result

We compared the learning performance based on (I) independent and (II) joint estimation, via the variance of the state $x^i_j$ given all past measurements. This value indicates the accuracy of the tracking behavior in each iteration $j$. We investigated the benefits of information sharing and used, as our basis for the investigation, two limiting cases of (8): (i) encountering pure process noise, and (ii) dealing with measurement noise only. From these benchmark examples, we were able to deduce properties for the general mixed-noise case in Section 4 and draw conclusions about the advantages of passing information in an ILC framework.

In order to compare the independent estimation result (I) with the joint estimation result (II), we derived an analytical expression for $p_j^{(1,1)}$.

PROPOSITION 1 The error variance of an agent's disturbance $p_j^{(1,1)}$ can be expressed in terms of the initial variances $\alpha$ and $\beta$, the number of agents $N$, and the iteration $j$,

$$p_j^{(1,1)} = \frac{\alpha + \beta + j\beta^2 + jN\alpha\beta}{(1+j\beta)(1+j\beta+jN\alpha)}. \tag{20}$$

The result is obtained by solving the Kalman filter equations for (12) with initial conditions (14) and (16).

■

A detailed proof is found in the appendix, Section A.

Next, we use the relation (20) to derive an upper bound on the performance improvement due to joint estimation. Two limiting cases are considered: (i) pure process noise and (ii) pure measurement noise.

### 3.1 Pure Process Noise

We assumed perfect measurements, i.e. $\mu_j^i = 0$ in (7) and $\lambda = 1$ in (11). The noise $v_j^i$ is interpreted as pure process noise, $v_j^i = \xi_j^i$. The performance of independent (I) vs. joint (II) estimation is analyzed through the variance of the state estimate. As mentioned in Section 2.1, the goal of ILC is to reduce the value $x_j^i$. This is achieved best if the variance in the estimate of $x_j^i$ is small. That is, the variance of the state estimate can be used as a measure of learning performance. Given (6) and (10), the best estimate of the state $\widehat{x}_j^i$ at iteration $j$ is equal to the current disturbance estimate $\widehat{d}^i{}_j$,

$$\widehat{x}_j^i = \widehat{d}^i{}_j, \tag{21}$$

since the noise $\xi_j^i$ has zero mean. Recalling the noise characteristics (10) and the previous assumption of mutual independence between $d^i$ and $v_j^i$, we obtain the variance of state estimate from the sum of the variance of the estimate $\widehat{d}^i{}_j$ and the variance of $\xi_j^i$. That is, with (17) and (11),

$$E\left[(x_j^i - \widehat{x}_j^i)^2\right] = E\left[(d^i + \xi_j^i - \widehat{d}^i{}_j)^2\right] = p_j^{(1,1)} + \lambda\,, \tag{22}$$

where $\lambda = 1$ in the pure process noise case. We introduce the performance index (for the pure process noise case) as the ratio of the state variance in the independent case vs. the joint case,

$$R^{\text{proc}} = \frac{p_j^{(1,1)}|_{N=1} + 1}{p_j^{(1,1)} \qquad + 1}, \tag{23}$$

using the notation of (17).

The following theorem can be stated:

THEOREM 1 The bounds on the performance improvement due to joint estimation (vs. independent estimation) are given by

$$1 \le R^{\text{proc}} \le \frac{1+j}{j}, \qquad \forall\, \alpha, \beta, N, j, \tag{24}$$

where the best performance improvement occurs when $N \to \infty$, $\alpha \to \infty$, and $\beta = 0$. In this case, $R^{\text{proc}} = (1+j)/j$. ■

Interpretation of the result:

- The performance improvement due to joint estimation has an upper bound which is valid for all possible combinations of $\alpha$, $\beta$, $N$, and $j$.
- Joint estimation is most beneficial if the agents' common disturbance component dominates and the individual noise component is negligible compared to the process noise; this corresponds to a large common noise variance $\alpha$ and a small individual component $\beta \ll 1$.
- The largest possible improvement in performance is a factor of 2, which is obtained only in the first iteration. With more iterations, the performance index rapidly decays to 1. In other words, the more often the agents perform a task, the less beneficial the exchange of information.
- Intuitively, the result shows that if the agents are different, the measurements of the other agents do not provide significant information for an individual's performance improvement. If the agents are almost identical, however, 'averaging' the measurements of the agents via a joint estimation still has no 'visible' effect, since the process noise directly corrupts the value of interest, $x^i_j$, see (6).

Moreover, independent estimation and learning (I) is robust to uncertainties in the initial noise assumptions (10). Note that the variance of an individual's disturbance in the independent case depends solely on the sum $(\alpha+\beta)$, cf. (20) with $N = 1$. In other words, the assumption on how the disturbance $d^i$ is decomposed in $d^0$ and $d^{i,ind}$, does not enter the result. It does, however, affect the joint estimation.

To conclude, there is little benefit of sharing information in the case of pure process noise.

*Proof.* Based on the closed-form representation in (20), Theorem 1 is proven by introducing $R^{\text{proc}}$ as a function of $j$, $\alpha$, $\beta$, and $N$,

$$R^{\text{proc}}_j(\alpha,\beta,N) = \frac{p^{(1,1)}_j(\alpha,\beta,1)+1}{p^{(1,1)}_j(\alpha,\beta,N)+1}. \tag{25}$$

Recalling the properties

$$\alpha,\beta \geq 0 \quad \text{and} \quad j,N \in \{1,2,\dots\}, \tag{26}$$

we note that $p^{(1,1)}_j(\alpha,\beta,N) \geq 0$ for all possible arguments. By taking partial derivatives of $R^{\text{proc}}_j(\alpha,\beta,N)$, it can be shown that

$$\frac{\partial R^{\text{proc}}_j(\alpha,\beta,N)}{\partial N} \geq 0 \tag{27}$$

and $R^{\text{proc}}_j(\alpha,\beta,N)$ is bounded by

$$R^{\text{proc}}_j(\alpha,\beta,\infty) := \lim_{N\to\infty} R^{\text{proc}}_j(\alpha,\beta,N) \tag{28}$$

with

$$R_j^{\text{proc}}(\alpha,\beta,\infty) = \frac{1+\frac{\alpha+\beta}{1+j(\alpha+\beta)}}{1+\frac{\beta}{1+j\beta}}.$$

Secondly, it is shown that

$$\frac{\partial R_j^{\text{proc}}(\alpha,\beta,\infty)}{\partial \alpha} \geq 0 \tag{29}$$

with

$$R_j^{\text{proc}}(\infty,\beta,\infty) := \lim_{\alpha\to\infty} R_j^{\text{proc}}(\alpha,\beta,\infty) = \frac{1+\frac{1}{j}}{1+\frac{\beta}{1+j\beta}},$$

that is $R_j^{\text{proc}}(\alpha,\beta,N) \leq R_j^{\text{proc}}(\infty,\beta,\infty)$. Finally, with

$$\frac{\partial R_j^{\text{proc}}(\infty,\beta,\infty)}{\partial \beta} \geq 0, \tag{30}$$

and

$$R_j^{\text{proc}}(\infty,0,\infty) = 1+\frac{1}{j},$$

statement (24) is proven,

$$R_j^{\text{proc}}(\alpha,\beta,N) \leq R_j^{\text{proc}}(\infty,0,\infty)$$

for all $\alpha$, $\beta$, $N$, $j$. The lower bound is obtained for $N=1$, cf. (27). Matlab and Mathematica files for reproducing the results below are available at `www.idsc.ethz.ch/Downloads/multiagentILC`. □

### 3.2 Pure Measurement Noise

We studied the system properties under the assumption of zero process noise, i.e. $\xi_j^i = 0$ in (6) and $\lambda = 0$ in (11), and interpreted $v_j^i$ as pure measurement noise, $v_j^i = \mu_j^i$. Following the derivation (22), the ratio of the state variances (for the pure measurement noise case) is given by

$$R^{\text{meas}} = \frac{p_j^{(1,1)}\big|_{N=1}}{p_j^{(1,1)}}. \tag{31}$$

The following theorem can be stated:

THEOREM 2 The bounds on the performance improvement due to joint estimation (vs. independent estimation) are given by

$$1 \leq R^{\text{meas}} \leq N, \qquad \forall\, \alpha,\beta,N,j, \tag{32}$$

where the best performance improvement occurs when $\alpha\to\infty$ and $\beta = 0$, for all $N$, $j$. In this case, $R^{\text{meas}} = N$. ■

Interpretation of the result:

- Again, an upper bound of the performance index is found which is valid for all possible combinations of $\alpha, \beta, N$, and $j$. However, the upper bound does not depend on the number of iterations.
- Joint estimation is most beneficial if the agents' common disturbance component dominates and the individual noise component is negligible compared to the measurement noise; this corresponds to a large common noise variance $\alpha$ and a negligible individual component $\beta \ll 1$.The largest possible improvement in performance is a factor of $N$.
- Intuitively, the result shows that if the agents are very similar ($\beta \ll 1$), joint estimation has a 'visible' effect. The measurement noise is 'averaged out' and, moreover, does not corrupt the performance result, $x_j^i$, directly, see (7). A significant improvement in the individual's performance can be achieved.

Joint estimation is beneficial when considering a large group of almost identical agents, where the individual disturbance is small compared to the measurement noise.

*Proof.* The proof of Theorem 2 proceeds similarly as the proof in Section 3.1. With (20), the performance index $R^{\text{meas}}$ is given as a function of $j$, $\alpha$, $\beta$, and $N$,

$$R_j^{\text{meas}}(\alpha,\beta,N) = \frac{p_j^{(1,1)}(\alpha,\beta,1)}{p_j^{(1,1)}(\alpha,\beta,N)}. \tag{33}$$

Partial derivatives are directly computed, where

$$\frac{\partial R^{\text{meas}}}{\partial \beta} \leq 0, \qquad \frac{\partial R^{\text{meas}}}{\partial \alpha} \geq 0, \qquad \frac{\partial R^{\text{meas}}}{\partial N} \geq 0, \tag{34}$$

with (26). In addition, the limiting property for $\beta = 0$ is

$$R_j^{\text{meas}}(\alpha,0,N) = \frac{1+\alpha jN}{1+\alpha j}$$

and

$$\lim_{\alpha\to\infty} R_j^{\text{meas}}(\alpha,0,N) = N, \qquad \forall\, j, N.$$

The lower bound is obtained for $N = 1$, cf. (34). Matlab and Mathematica files for reproducing the results are available at `www.idsc.ethz.ch/Downloads/multiagentILC`. □

## 4. Numerical Examples

Characteristic features of the performance indices, $R^{\text{proc}}$ and $R^{\text{meas}}$, are highlighted by showing selected numerical examples. In the subsequent considerations, the general mixed-noise case

is included and put in context with the results for (i) pure process and (ii) pure measurement noise.

The performance index for the mixed-noise case with $0 \leq \lambda \leq 1$, cf. (11), is derived analogously to (22) and (23):

$$R^{\text{mix}} = \frac{p_j^{(1,1)}\big|_{N=1} + \lambda}{p_j^{(1,1)} \qquad + \lambda}. \tag{35}$$

A comparison of the three performance indices, $R^{\text{proc}}$, $R^{\text{mix}}$, and $R^{\text{meas}}$, shows that

$$R^{\text{proc}} \leq R^{\text{mix}} \leq R^{\text{meas}}, \qquad \forall\, \alpha, \beta, N, j, \tag{36}$$

since $\lambda \in [0,1]$ and from (20),

$$p_j^{(1,1)} \leq p_j^{(1,1)}\big|_{N=1}, \qquad \forall\, \alpha, \beta, N, j. \tag{37}$$

An intuitive explanation for (36) is that the process noise $\xi_j^i$ directly corrupts the value of interest, state $x_j^i$, which represents the deviation from the desired trajectory and is aimed to be small. The measurement noise $\mu_j^i$ acts on the output $y_j^i$. In this case, multiple agents are beneficial in order to average out the measurement noise.

In Fig. 1 to 3, the evolution of the performance indices, $R^{\text{proc}}$, $R^{\text{mix}}$, and $R^{\text{meas}}$, is shown for different pairs of $\alpha$ and $\beta$. A group of $N = 10$ agents is considered and $\lambda = 0.1$ is chosen for the mixed-noise case. Even for this small value of $\lambda$, we observe a noticeable degradation of the performance index $R^{\text{mix}}$ (compared to $R^{\text{meas}}$). Note that the scaling of the vertical axis changes in the plots presented. The figures highlight the relationship (36). For the limiting case $j \to \infty$, $\beta > 0$, we observe

$$\lim_{j\to\infty} R^{\text{mix}} = 1. \tag{38}$$

Note that the mixed-noise case includes the special cases of pure process and pure measurement noise. In fact, if the number of iterations increases, the benefits of joint estimation become negligible. This result can be derived analytically from (20) and (35). As stated in Theorem 2, for the limiting case $\beta = 0$,

$$\lim_{j\to\infty} R^{\text{meas}} = N. \tag{39}$$

Thus, we observe that joint estimation yields a significant improvement in performance only if the agents are identical, $\beta = 0$, and the system dynamics are not corrupted by process noise, see Fig. 4.1(a). Moreover, Fig. 1 shows how the behavior of $R^{\text{proc}}$, $R^{\text{mix}}$, and $R^{\text{meas}}$ change, if the individual disturbance component $\beta$ is gradually increased from zero for a given common disturbance $\alpha$. The relation

$$\frac{\partial R^{\text{meas}}}{\partial j} > 0 \qquad \Leftrightarrow \qquad \beta(\beta + \alpha N) < \frac{1}{j^2}, \tag{40}$$

derived from (20) and (31), provides insight in the evolution of the performance index $R^{\text{meas}}$. Fig. 1 shows that the performance indices, which represent the performance improvement due to joint estimation, decrease with larger values $\beta$. The relation,

$$\frac{\partial R^{\text{mix}}}{\partial \beta} \leq 1, \tag{41}$$

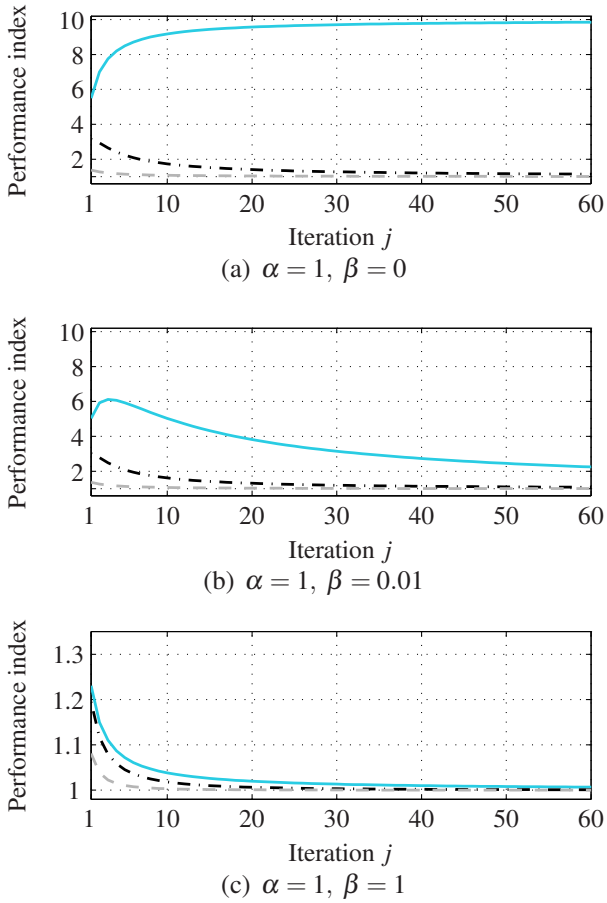


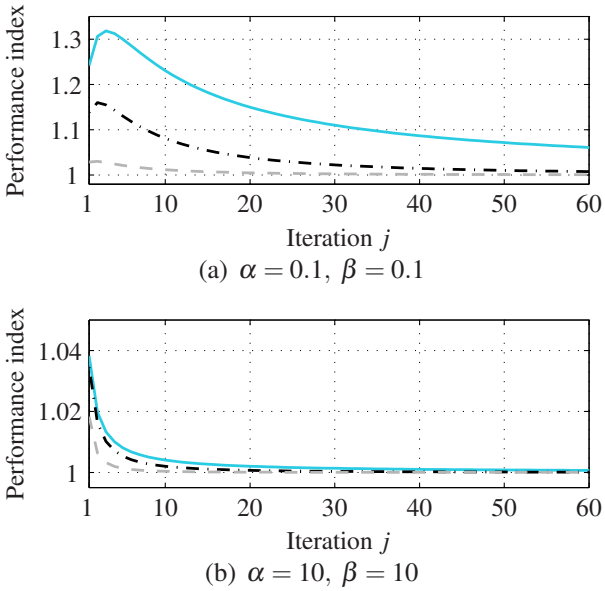

**Figure 1.** Evolution of the performance indices for $N = 10$ agents: pure measurement noise $R^{\text{meas}}$ (solid line), mixed noise case $R^{\text{mix}}$ with $\lambda = 0.1$ (dashed-dotted line), and pure process noise $R^{\text{proc}}$ (dashed line).

is derived from (20) and (35), where the mixed-noise case includes the special cases of pure process and pure measurement noise. The variances of the common and individual disturbance component, $\alpha$ and $\beta$, must be interpreted in relation to the variance of the noise $v_j^i$ which is normalized to 1, see (10). Fig. 4.1(c) shows the evolution of the performance indices if the disturbance variances have the same value as the noise variance. If both disturbance variances, $\alpha$ and $\beta$, are smaller than the noise variance, a behavior as shown in Fig. 4.2(a) is obtained. Fig. 4.2(b) shows both disturbance variances being larger than the noise variance.

In Fig. 3, the two cases, $\alpha \ll 1$, $\beta \gg 1$ and $\beta \ll 1$, $\alpha \gg 1$, are depicted. Fig. 4.3(a) underlines the fact that, in the general case,

$$\lim_{\alpha \to 0} R^{\text{mix}} = 1\,, \qquad \lim_{\beta \to \infty} R^{\text{mix}} = 1\,. \tag{42}$$

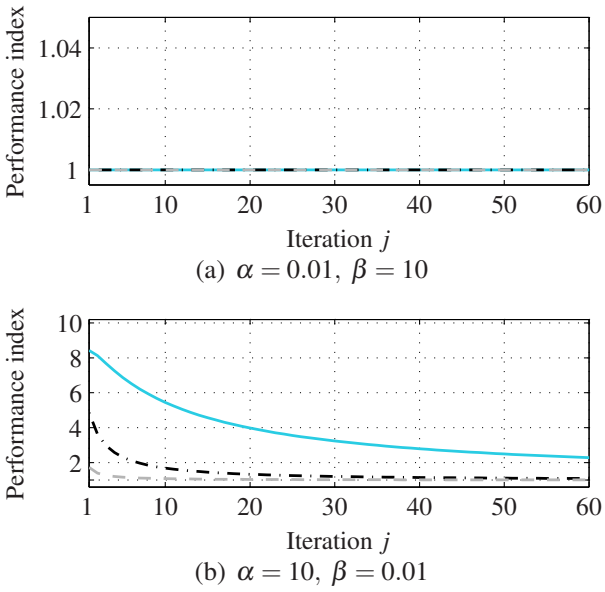
In contrast, information exchange and joint estimation is most beneficial when $\alpha \to \infty$ and $\beta = 0$, cf. Theorem 1 and 2. Fig. 4.3(b) shows a corresponding setting.




(a) $\alpha = 0.1,\ \beta = 0.1$

(b) $\alpha = 10,\ \beta = 10$

**Figure 2.** Evolution of the performance indices for $N = 10$ agents: pure measurement noise $R^{\text{meas}}$ (solid line), mixed noise case $R^{\text{mix}}$ with $\lambda = 0.1$ (dashed-dotted line), and pure process noise $R^{\text{proc}}$ (dashed line).




(a) $\alpha = 0.01,\ \beta = 10$

(b) $\alpha = 10,\ \beta = 0.01$

**Figure 3.** Evolution of the performance indices for $N = 10$ agents: pure measurement noise $R^{\text{meas}}$ (solid line), mixed noise case $R^{\text{mix}}$ with $\lambda = 0.1$ (dashed-dotted line), and pure process noise $R^{\text{proc}}$ (dashed line).

## 5. Conclusions

In this paper we considered a group of agents which share the same dynamics and a common iteration-independent disturbance, but differ in an additional individual error component. In the

context of having these agents learn to perform an identical task, we asked: How beneficial is it to exchange experience in order to improve an individual agent's learning performance? We considered two cases: (I) independent learning without information exchange and (II) learning based on full information exchange between agents. In the proposed framework, the question can be reduced to the comparison of the disturbance estimate in case of independent estimation (I) and when solving a global estimation problem for (II). An upper bound for the performance improvement due to information exchange is derived analytically and reflects the limited benefit of sharing information in the given setup. In the best case – where the noise is due to measurement noise only, the agent's common disturbance dominates, and the individual disturbance component is small compared to the noise – joint estimation improves the performance by a factor equal to the number of agents. That is, instead of one agent performing a task $N$ times, $N$ agents performing the task once results in the same accuracy for the disturbance estimate. For the general case and, in particular, in the presence of process noise or a large individual disturbance component, the benefits are shown to be limited.

## A. Appendix

We derive an explicit representation of the variance $p_j^{(1,1)}$ that depends only on $\alpha$, $\beta$, $j$, and $N$ as presented in Proposition 1. Matlab and Mathematica files for reproducing the results below are available at `www.idsc.ethz.ch/Downloads/multiagentILC`.

*Proof.* A closed form of the covariance matrix $P_j$ is derived, cf. (16) and (17). Since noise is assumed to have the same characteristics for each agent, by symmetry,

$$p_j^{(k,l)} = \begin{cases} p_j^{(0,0)} & \text{if} \quad k=l=0 \\ p_j^{(0,1)} & \text{if} \quad kl=0 \text{ and } k\neq l \\ p_j^{(1,1)} & \text{if} \quad k=l\neq 0 \\ p_j^{(1,2)} & \text{otherwise}. \end{cases} \tag{43}$$

We obtain the previous values by solving the filter equations, cf. [26, 27],

$$\begin{aligned} Q_j &= HP_{j-1}H^T + I \\ K_j &= P_{j-1}H^T Q_j^{-1} \\ P_j &= (I-K_jH)P_{j-1}, \end{aligned} \tag{44}$$

where $Q_j = [q_j^{(k,l)}]$, $k,l \in \mathscr{I}$ and $K_j = [k_j^{(k,l)}]$, $k \in \mathscr{K}$, $l \in \mathscr{I}$. With (43) and (44), the matrix $Q_j$ and its inverse $Q_j^{-1} = [m_j^{(k,l)}]$ are directly computed,

$$\begin{aligned} q_j^{(k,l)} &= \begin{cases} 1+p_{j-1}^{(1,1)} & \text{if} \quad k=l \\ p_{j-1}^{(1,2)} & \text{otherwise} \end{cases} \\ m_j^{(k,l)} &= \begin{cases} m_j^{(1,1)} & \text{if} \quad k=l \\ m_j^{(1,2)} & \text{otherwise}, \end{cases} \end{aligned} \tag{45}$$

where

$$m_j^{(1,1)} = \frac{1 + p_{j-1}^{(1,1)} + \left(N-2\right) p_{j-1}^{(1,2)}}{n_1 n_2}, \qquad m_j^{(1,2)} = \frac{-p_{j-1}^{(1,2)}}{n_1 n_2}, \tag{46}$$

with

$$n_1 = \left(1 + p_{j-1}^{(1,1)} - p_{j-1}^{(1,2)}\right), \qquad n_2 = \left(1 + p_{j-1}^{(1,1)} + \left(N-1\right) p_{j-1}^{(1,2)}\right). \tag{47}$$

With this, the filtering matrix $K_j$ is given by

$$k_j^{(k,l)} = \begin{cases} k_j^{(0,1)} & \text{if} \quad k = 0 \\ k_j^{(1,1)} & \text{if} \quad k = l \\ k_j^{(1,2)} & \text{otherwise}, \end{cases} \tag{48}$$

where

$$\begin{aligned} k_j^{(0,1)} &= p_{j-1}^{(0,1)} \left(m_j^{(1,1)} + \left(N-1\right) m_j^{(1,2)}\right) \\ k_j^{(1,1)} &= p_{j-1}^{(1,1)} m_j^{(1,1)} + \left(N-1\right) p_{j-1}^{(1,2)} m_j^{(1,2)} \\ k_j^{(1,2)} &= p_{j-1}^{(1,1)} m_j^{(1,2)} + p_{j-1}^{(1,2)} m_j^{(1,1)} + \left(N-2\right) p_{j-1}^{(1,2)} m_j^{(1,2)}. \end{aligned} \tag{49}$$

From (44), the following values for $P_j$ are found,

$$p_j^{(k,l)} = \begin{cases} p_j^{(0,0)} & \text{if} \quad k = l = 0 \\ p_j^{(0,1)} & \text{if} \quad kl = 0 \ \text{ and } \quad l \neq k \\ p_j^{(1,1)} & \text{if} \quad k = l \neq 0 \\ p_j^{(1,2)} & \text{otherwise} \end{cases} \tag{50}$$

with

$$\begin{aligned} p_j^{(0,0)} &= p_{j-1}^{(0,0)} - N p_{j-1}^{(0,1)} k_j^{(0,1)} \\ p_j^{(0,1)} &= p_{j-1}^{(0,1)} + p_{j-1}^{(1,1)} k_j^{(0,1)} - \left(N-1\right) p_{j-1}^{(1,2)} k_j^{(0,1)} \\ p_j^{(1,1)} &= \left(1 - k_j^{(1,1)}\right) p_{j-1}^{(1,1)} - \left(N-1\right) p_{j-1}^{(1,2)} k_j^{(1,2)} \\ p_j^{(1,2)} &= \left(1 - k_j^{(1,1)}\right) p_{j-1}^{(1,2)} - k_j^{(1,2)} p_{j-1}^{(1,1)} - \left(N-2\right) p_{j-1}^{(1,2)} k_j^{(1,2)}. \end{aligned}$$

We prove the desired symmetry and obtain the following values for (43) by induction, using (50) with starting condition (16):

$$\begin{aligned} p_j^{(0,0)} &= \frac{(1+j\beta)\alpha}{1+j\beta+jN\alpha}, \qquad & p_j^{(1,1)} &= \frac{\alpha+\beta+j\beta^2+jN\alpha\beta}{(1+j\beta)(1+j\beta+jN\alpha)}, \\ p_j^{(0,1)} &= \frac{\alpha}{1+j\beta+jN\alpha}, \qquad & p_j^{(1,2)} &= \frac{\alpha}{(1+j\beta)(1+j\beta+jN\alpha)}. \end{aligned} \tag{51}$$

The only value of interest is $p_j^{(1,1)}$. □

## References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[2] Z. Bien and J. X. Xu, *Iterative learning control: analysis, design, integration and applications*. Kluwer Academic Publishers Norwell, MA, USA, 1998.

[3] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[4] K. L. Moore, "An iterative learning control algorithm for systems with measurement noise," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 1, 1999, pp. 270–275.

[5] J. H. Lee, K. S. Lee, and W. C. Kim, "Model-based iterative learning control with a quadratic criterion for time-varying linear systems," *Automatica*, vol. 36, no. 5, pp. 641–657, 2000.

[6] M. Phan and R. Longman, "Higher-order iterative learning control by pole placement and noise filtering," in *Proceedings of the IFAC World Congress*, 2002, pp. 1899–1904.

[7] M. Norrlöf, "An adaptive iterative learning control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, 2002.

[8] ——, "Disturbance rejection using an ILC algorithm with iteration varying filters," *Asian Journal of Control*, vol. 6, pp. 432–438, 2004.

[9] A. Schöllig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European Control Conference (ECC)*, 2009, pp. 1505–1510.

[10] K. S. Lee, J. Lee, I. Chin, J. Choi, and J. H. Lee, "Control of wafer temperature uniformity in rapid thermal processing using an optimal iterative learning control technique," *Industrial and Engineering Chemistry Research*, vol. 40, no. 7, pp. 1661–1672, 2001.

[11] I. Chin, S. J. Qin, K. S. Lee, and M. Cho, "A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection," *Automatica*, vol. 40, no. 11, pp. 1913–1922, 2004.

[12] M. Cho, Y. Lee, S. Joo, and K. S. Lee, "Semi-empirical model-based multivariable iterative learning control of an RTP system," *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 3, pp. 430–439, 2005.

[13] J. K. Rice and M. Verhaegen, "A structured matrix approach to efficient calculation of LQG repetitive learning controllers in the lifted setting," *International Journal of Control*, vol. 83, no. 6, pp. 1265–1276, 2010.

[14] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, vol. 45, no. 7, pp. 22–29, 2008.

[15] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–19, 2008.

[16] H.-S. Ahn, K. L. Moore, and Y. Chen, *Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems (Communications and Control Engineering)*, 1st ed. Springer, 2007.

[17] H.-S. Ahn and Y. Chen, "Iterative learning control for multi-agent formation," in *Proceedings of the International Joint Conference on Control, Automation, and Systems of the Society of Instrument and Control Engineers and the Institute of Control, Robotics and Systems (ICCAS-SICE)*, 2009, pp. 3111–3116.

[18] A. P. Schoellig, J. Alonso-Mora, and R. D'Andrea, "Independent vs. joint estimation in multi-agent iterative learning control," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2010, pp. 6949–6954.

[19] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the International Conference on Machine Learning*, 1993, pp. 330–337.

[20] M. Q. Phan and R. W. Longman, "A mathematical theory of learning control for linear discrete multivariable systems," in *Proceedings of the AIAA/AAS Astrodynamics Conference*, 1988, pp. 740–746.

[21] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 5, 2001, pp. 4463–4468.

[22] B. Bamieh, J. B. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[23] J. Hätönen, D. Owens, and K. Feng, "Basis functions and parameter optimisation in high-order iterative learning control," *Automatica*, vol. 42, no. 2, pp. 287–294, 2006.

[24] K. Barton and A. Alleyne, "A cross-coupled iterative learning control design for precision motion control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1218–1231, 2008.

[25] M. Butcher, A. Karimi, and R. Longchamp, "Iterative learning control based on stochastic approximation," in *Proceedings of the IFAC World Congress*, 2008, pp. 1478–1483.

[26] C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications (Springer Series in Information Sciences)*. Springer, 1998.

[27] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*. Cambridge University Press, 2007.

Paper IV

# Sensitivity of Joint Estimation in Multi-Agent Iterative Learning Control

Angela P. Schoellig · Raffaello D'Andrea

**Abstract**

This paper studies iterative learning control (ILC) in a multi-agent framework, wherein a group of agents We consider a group of agents that simultaneously learn the same task, and revisit a previously developed algorithm, where agents share their information and learn jointly. We have already shown that, as compared to an independent learning model that disregards the information of the other agents, and when assuming similarity between the agents, a joint algorithm improves the learning performance of an individual agent. We now revisit the joint learning algorithm to determine its sensitivity to the underlying assumption of similarity between agents. We note that an incorrect assumption about the agents' degree of similarity degrades the performance of the joint learning scheme. The degradation is particularly acute if we assume that the agents are more similar than they are in reality; in this case, a joint learning scheme can result in a poorer performance than the independent learning algorithm. In the worst case (when we assume that the agents are identical, but they are, in reality, not) the joint learning does not even converge to the correct value. We conclude that, when applying the joint algorithm, it is crucial not to overestimate the similarity of the agents; otherwise, a learning scheme that is independent of the similarity assumption is preferable.

Published in *Proc. of the 18th IFAC (International Federation of Automatic Control) World Congress,* 2011.
DOI: 10.3182/20110828-6-IT-1002.03687.

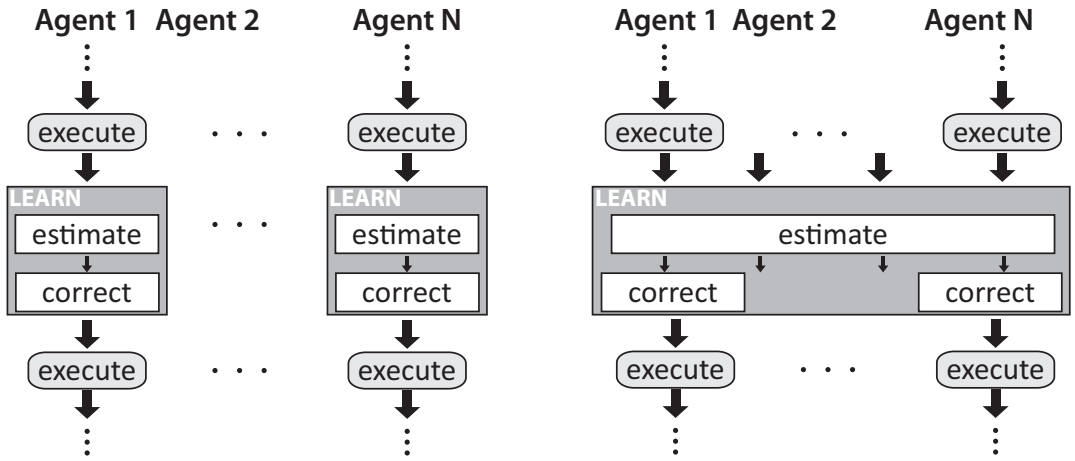



**Figure 1.** Independent (left) vs. joint (right) estimation and learning.

## 1. Introduction

In most cases, multi-agent learning aims towards improving the joint performance of a group of agents that solve a complex or distributed task together. Through interaction and collaboration, the agents are able to jointly approach the common task and learn to work together in order to achieve the global objectives, cf. [1]. Reinforcement learning is a powerful tool to solve such cooperative problems. Here, agents are generally categorized to be either homogeneous or heterogeneous, see e.g. [2, 3]. Though the robustness of such learning schemes to parameter variations was suggested in recent publications (see [4, 5]), it has yet to be studied in detail.

In this paper, we focus on the potential for an individual agent to improve its performance when conducting a task alongside a group of similar agents conducting the same task. We extend the work in [6][1], where the learning performance of an individual agent is analyzed and compared for two scenarios: (i) the agent learns independently, disregarding the information of the other agents, and (ii) the agent has access to the knowledge of the other agents and optimally takes this information into account when learning the desired task. An iterative learning control (ILC) scheme was used to approach this problem (see [7, 8] for an introduction to ILC). ILC has been viewed as a two-step process of first identifying the unknown repetitive disturbances that corrupt the agent's performance and later compensating for the disturbances by adapting the input, cf. [9–11]. This scheme allowed us to reduce the previous question to an estimation problem, and we were able to show that, when assuming similarity between the agents, a joint estimation scheme that exploits the information of all agents is always beneficial.

The results in [6] were obtained under the assumption that we know the degree of similarity between the agents; the goal of this work is to study the sensitivity of joint estimation to the underlying similarity assumption. We analyze the effects of an assumption error on the joint estimation result in order to determine whether it is possible that an incorrect assumption regarding the agents' similarity cause the joint estimation scheme to perform more poorly than

[1] Paper and additional material may be found on the project webpage: `www.idsc.ethz.ch/Downloads/multiagentILC`.

the independent estimation scheme.

The paper is organized as follows: Sec. 2 recalls the dynamic equations that define the multi-agent iterative learning problem as previously derived in [6], and introduces the parameter which defines the similarity between agents. Sec. 3 solves the estimation problem under the assumption that the degree of similarity between the agents is not known precisely. Using the results from Sec. 3 as a basis, the sensitivity of the joint estimation scheme to the assumption errors is studied in Sec. 4. To help readers visualize the analytical results derived herein, we present several numerical examples in Sec. 5. The work is summarized in Sec. 6. Proofs are presented in Appendix A.1, with additional files available at `www.idsc.ethz.ch/Downloads/multiagentILC`.

## 2. Problem Statement

We extend the work in [6], where we considered a group of $N$ agents that simultaneously and repeatedly perform the same task. The execution of the task is corrupted by an unknown, repetitive disturbance that is constant across iterations. In this context, we assume that the agents are similar in the sense that they have the same nominal dynamics and share a common iteration-independent, repetitive disturbance component. In addition, process noise acts on the agent's dynamics, and varies from trial to trial. Our goal is to improve the agents' performance by estimating the repetitive disturbance from past measurements; once the disturbance is known, an adapted input trajectory can be created to compensate for it.

### 2.1 Agent Dynamics

The dynamics of an agent $i \in \mathscr{I} = \{1,2,\ldots,N\}$ during a single execution of the task are represented in the lifted domain, cf. [12–14]. A given discrete-time input signal

$$u_j^i = \left(u^i(0), u^i(1), \ldots, u^i(T)\right) \tag{1}$$

at iteration $j \in \{1,2,\ldots\}$ is mapped to the corresponding lifted states $x^i$ via a constant matrix $F$, which represents the nominal dynamics of the agents,

$$x_j^i = F u_j^i + d^i + \xi_j^i, \tag{2}$$

In this context, $(T+1)$ samples represent a single run. The vector $d^i$ represents the repetitive disturbance and $\xi_j^i$ accounts for the trial-uncorrelated process noise. The vectors $x_j^i$ and $u^i$ are defined as the deviation from the desired task trajectory and the corresponding nominal input, see for example [11]. The agents' output $y_j^i$ (also defined as the deviation from the nominal output) is corrupted by measurement noise and similarly represented in the lifted domain,

$$y_j^i = x_j^i + \mu_j^i. \tag{3}$$

Differences between the agents are captured in the disturbance vector $d^i$, which is composed of a common part $d^0$ identical for all agents, and an individual part $d^{i,ind}$,

$$d^i = d^0 + d^{i,ind}, \qquad \forall\, i \in \mathscr{I}. \tag{4}$$

For a more detailed introduction to the lifted system representation refer to [8, 15–17]

In the above context, the goal of the iterative learning algorithm is to reduce $x_j^i$ (that is, the deviation from the desired task trajectory) with an increasing number of iterations $j$. In [6], we showed that the learning problem can be divided into two steps: (i) estimating the disturbance vector $d^i$ based on all measurements from previous iterations, and (ii) determining an appropriate open-loop input for the next trial that compensates for the disturbance, see Fig. 1. We saw that the characteristics of a joint learning scheme can be studied by focusing on the estimation problem; compensating input for each agent is found by solving an optimization problem for each agent once the disturbance estimate of $d^i$ is updated, cf. Fig. 1.

### 2.2 Simplified Model

Focusing on the estimation problem, we consider a condensed form of the above multi-agent system representation (2)-(3),

$$x_j^i = d^i + \xi_j^i \tag{5}$$
$$y_j^i = x_j^i + \mu_j^i, \tag{6}$$

which features the key noise and disturbance characteristics, but omits the *known* part $Fu_j^i$, without loss of generality. Equations (5) and (6) are summarized by

$$y_j^i = d^i + \upsilon_j^i, \tag{7}$$

where $\upsilon_j^i = \xi_j^i + \mu_j^i$ captures both process and measurement noise.

Moreover, assuming both identical noise characteristics and independence of the single entries in the vectors $d^i$ and $\upsilon_j^i$, the problem reduces to the scalar case,

$$y_j^i = d^0 + d^{i,ind} + \upsilon_j^i, \tag{8}$$

where all variables are scalar-valued. The probability distributions are given by

$$\begin{aligned} d^0 &\sim \mathcal{N}(0, \alpha) \\ d^{i,ind} &\sim \mathcal{N}(0, \beta) \\ \upsilon_j^i &\sim \mathcal{N}(0, 1), \qquad \alpha, \beta \geq 0, \end{aligned} \tag{9}$$

where all quantities, $\upsilon_j^i$, $i \in \mathscr{I}$, $j \in \{1, 2, \dots\}$, $d^{i,ind}$, $i \in \mathscr{I}$, and $d^0$, are assumed to be mutually independent. The notation $\mathcal{N}(0, \alpha)$ represents a normal distribution with mean 0 and variance $\alpha$. Note that in (9), the variance of the individual disturbance $d^{i,ind}$ is assumed to be identical for all agents $i \in \mathscr{I}$. Without loss of generality, the variances are normalized such that the variance of $\upsilon_j^i$ is 1. For the variances of the process and measurement noise, this means

$$\begin{aligned} \xi_j^i &\sim \mathcal{N}(0, \lambda) \\ \mu_j^i &\sim \mathcal{N}(0, 1-\lambda), \qquad 0 \leq \lambda \leq 1 \end{aligned} \tag{10}$$

assuming independence between $\xi_j^i$ and $\mu_j^i$. A value $\lambda = 1$ represents the case of encountering only process noise, whereas $\lambda = 0$ reflects the case where the noise is due to measurement only.

### 2.3 Similarity Assumption

In our previous work [6], we assumed that the variances of the individual and common disturbance, $\alpha$ and $\beta$, are known. In reality, however, these values are difficult to determine. While the sum $(\alpha+\beta)$ may be approximated with reasonable accuracy (it indicates the magnitude of the agent's disturbance $d^i$, or more precisely, the probability of having larger values for $d^i$), a prior partitioning of the disturbance $d^i$ into an individual and a common component is almost impossible. In other words, when facing a real multi-agent learning problem, the determined ratio between $\alpha$ and $\beta$, is subject to error.

For the subsequent analysis, we assume that the sum,

$$\gamma = \alpha + \beta, \tag{11}$$

is known precisely. With respect to the partitioning of $d^i$ into the individual and common disturbance component, we distinguish between the nominal values,

$$\bar{\alpha} = \bar{\varepsilon}\,\gamma \qquad \text{and} \quad \bar{\beta} = (1-\bar{\varepsilon})\,\gamma, \tag{12}$$

and the real variances $\alpha$ and $\beta$, defined analogously by $\varepsilon$, where $0 \leq \bar{\varepsilon}, \varepsilon \leq 1$. The nominal values represent our assumption on the individual and common disturbance component. The real ratio $\varepsilon$ of the multi-agent system is unknown. The assumption error $\delta$ defines the difference between the real disturbance ratio and our assumed partitioning,

$$\delta = \varepsilon - \bar{\varepsilon}\,. \tag{13}$$

Below we study the effects of the assumption error $\delta$ on the performance of the joint learning algorithm. Our goal is to determine the degree to which joint estimation is affected by incorrect assumptions of similarity between agents.

## 3. Estimation Problem

Analogously to [6], we consider two limiting approaches when solving the estimation problem: (I) independent estimation, and (II) joint estimation, see Fig. 1. In the case of independent estimation (I), each agent $i$ individually estimates its disturbance $d^i$, taking only its own measurements $y^i_j,\ j \in \{1, 2, \dots\}$, into account.

In the joint case (II), every agent has access to the measurements of all other agents. Based on this global knowledge, we can design a joint estimation scheme that exploits the measurements of all agents and provides estimates $d^i$ for every agent $i \in \mathscr{I}$. A vector $D$, which reflects the estimation objective in this case, is defined as: $D = (d^0, d^1, \dots, d^N) \in \mathbb{R}^{(N+1)}$. The measurements of all agents in the $j$th trial are combined in $Y_j = (y^1_j, y^2_j, \dots, y^N_j)$, and analogously, the noise vector is $V_j = (v^1_j, v^2_j, \dots, v^N_j)$. Based on this representation, the joint estimation problem can be formulated as a Kalman filter problem, cf. [18, 19]:

$$\begin{aligned} D_j &= D_{j-1}\,, && \forall\, j \geq 1, \\ Y_j &= H\,D_j + V_j\,, \end{aligned} \tag{14}$$

where $H = [\mathbf{0}, I]$ is a matrix with zeros in the first column, concatenated with an identity matrix of appropriate dimensions. The Kalman filter returns an unbiased state estimate $\widehat{D}_j$ for $j \geq 1$ that minimizes the error covariance matrix

$$S_j = E\left[(D_j - \widehat{D}_j)(D_j - \widehat{D}_j)^{\mathrm{T}}\right], \tag{15}$$

of trial $j$, taking measurements $Y_m,\ 1 \leq m \leq j$, into account. $E[\cdot]$ denotes the expected value.

The recursive algorithm is based on the stochastic characteristics of the noise terms $v_j^i$, defined by (9), and relies on a given initial covariance matrix $S_0$, which reflects the characteristics of the disturbances $d^i$. The initial disturbance estimate is obtained from (9),

$$\widehat{D}_0 = (0, 0, \ldots, 0), \tag{16}$$

and the initial covariance $S_0 = [s_0^{(k,l)}],\ k,l \in \mathscr{K} = \{0, 1, \ldots, N\}$ is given by

$$S_0 = E\left[D_0 D_0^{\mathrm{T}}\right] \tag{17}$$

and with (4),

$$s_0^{(k,l)} = E\left[d^k d^l\right] = E\left[\left(d^0 + d^{k,ind}\right)\left(d^0 + d^{l,ind}\right)\right], \tag{18}$$

where $d^{0,ind} = 0$.

When solving the filter equations, we distinguish between the real variance values of the system denoted by $\alpha, \beta$ and the nominal values $\bar{\alpha}, \bar{\beta}$ that represent our *assumption* on the individual and common disturbance component, see Sec. 2.3. The Kalman filter derivation below is based on the nominal values $\bar{\alpha}, \bar{\beta}$. The real values $\alpha, \beta$ are unknown and difficult to identify a priori. Note that in [6], we derived the estimation problem under the assumption $\alpha = \bar{\alpha}$ and $\beta = \bar{\beta}$.

The Kalman filter proceeds in two steps:

**Step 1** The Kalman gains $K_j$ are calculated prior to the experiment based on the nominal values $\bar{\alpha}, \bar{\beta}$ by solving the filter equations

$$\begin{aligned} Q_j &= HS_{j-1}H^T + I \\ K_j &= S_{j-1}H^T Q_j^{-1} \\ S_j &= (I - K_j H)S_{j-1} \end{aligned} \tag{19}$$

with initial covariance $S_0$, see (17) and (18). Recalling the mutual independence of $d^0$ and $d^{i,ind}$ for all $i \in \mathscr{I}$, the initial covariance is given by

$$s_0^{(k,l)} = \begin{cases} \bar{\alpha} + \bar{\beta} & \text{for } k = l \geq 1 \\ \bar{\alpha} & \text{otherwise}. \end{cases} \tag{20}$$

A closed-form representation of $K_j$ that depends only on $\bar{\alpha}$, $\bar{\beta}$, $N$ and $j$ was derived in [6]. The values are explicitly stated in (66) and (67) (in terms of $\gamma$ and $\bar{\varepsilon}$) and are used to derive the results in Sec. 4.

**Step 2** The disturbance estimate is updated in each iteration $j$ based on the measurement $Y_j$,

$$\widehat{D}_j = \widehat{D}_{j-1} + K_j\left(Y_j - H\widehat{D}_{j-1}\right), \tag{21}$$

where $\widehat{D}_0$ is given by (16).

Important to note is that the independent estimation problem (I) is simply a special case of the cooperative framework (II) with $N = 1$. We compare the performance of both estimation schemes via the variance of the disturbance estimate.

### 3.1 Variance of Disturbance Estimate

We determine the variance of the disturbance estimate when applying the above Kalman filter equations to the real system. The covariance matrix of the real system is denoted by $P_j = [p_j^{(k,l)}]$, $k,l \in \mathscr{K}$. A recursive equation for calculating $P_j$ is derived from (21),

$$P_j = \left(I - K_j H\right) P_{j-1} \left(I - K_j H\right)^T + K_j K_j^T \tag{22}$$

where $K_j$ represent the Kalman gains calculated from the nominal covariance matrices $S_j$, based on the assumed variance values $\bar{\alpha}, \bar{\beta}$. The initial covariance matrix $P_0$ is defined analogously to $S_0$, see (20), but is based on the unknown system variances $\alpha, \beta$,

$$p_0^{(k,l)} = \begin{cases} \alpha + \beta & \text{for } k = l \geq 1 \\ \alpha & \text{otherwise}. \end{cases} \tag{23}$$

In brief, the estimation algorithm is run based on our assumed variance values $\bar{\alpha}, \bar{\beta}$ and yields the Kalman gain $K_j$ used in (21); to determine the variance of the estimate when running the estimation on the real system, we have to take the real variance parameters $\alpha, \beta$ into account, via (23) and (21). This step is artificial, since the real values are not known; however, it allows us to study the effects of incorrect variance assumptions on the estimation result, see Sec. 4. In the ideal case when $\bar{\alpha} = \alpha$ and $\bar{\beta} = \beta$, $S_j = P_j$ for all $j \in \{1, 2, \dots\}$. This scenario was studied in [6].

When we compare the performance of the independent (I) and the joint (II) estimation, we use the variance of an individual's disturbance estimate, which in both cases is given by

$$E\left[(d^i - \widehat{d}^i_j)^2\right] = p_j^{(i,i)} = p_j^{(1,1)}, \qquad \forall\, i \in \mathscr{I}, \tag{24}$$

where $\widehat{D}_j = [\widehat{d}^i_j]$, $i \in \mathscr{I}$, and $P_j = [p_j^{(k,l)}]$, $k,l \in \mathscr{K}$. The variance is identical for all agents, since the same assumptions on the dynamics (8) and the initial noise characteristics (9) hold for every agent. The variance of an individual's disturbance (24) is a measure for the effectiveness of the disturbance compensation, since in the general ILC framework, cf. (2)-(3), the input update rule is based on the current estimate $\widehat{d}^i_j$. See for example [11].

Below, we distinguish between the individual disturbance variance (24) in the cases of joint and independent estimation, where the latter is given when evaluating the disturbance variance for $N = 1$, i.e.

$$p_j^{(1,1)}\big|_{N=1}. \tag{25}$$

Thus, the initial question can be reformulated: To what degree does joint estimation benefit the individual learning of an agent? How does an incorrect assumption on the initial variances affect the learning performance?

### 3.2 Performance Index

The performance of independent (I) vs. joint (II) estimation is analyzed through the variance of the state estimate. As mentioned in Section 2.1, the goal of ILC is to reduce the value $x_j^i$, cf. (5). This is best achieved if the variance in the estimate of $x_j^i$ is small; in other words, the variance of the state estimate can be used as a measure of learning performance, see [6].

Given (5) and (9), the best estimate of the state $\widehat{x}_j^i$ at iteration $j$ is equal to the current disturbance estimate,

$$\widehat{x}_j^i = \widehat{d}_j^i, \tag{26}$$

since the noise $\xi_j^i$ has zero mean. Recalling the noise characteristics (9) and the previous assumption of mutual independence between $d^i$ and $\xi_j^i$, we obtain the variance of state estimate from the sum of the variance of the estimate $\widehat{d}_j^i$ and the variance of $\xi_j^i$. That is, with (24) and (10),

$$E\left[(x_j^i - \widehat{x}_j^i)^2\right] = E\left[(d^i + \xi_j^i - \widehat{d}_j^i)^2\right] = p_j^{(1,1)} + \lambda\,. \tag{27}$$

We introduce the performance index as the ratio of the state variance in the independent case vs. the joint case,

$$R = \frac{p_j^{(1,1)}\big|_{N=1} + \lambda}{p_j^{(1,1)} \qquad + \lambda}\,, \tag{28}$$

using the notation of (25). Given this definition, a value $R > 1$ indicates that the joint estimation scheme is more beneficial than an independent estimation, while a value $R < 1$ means that the independent estimation yields a better performance. The larger the value $R$, the more beneficial the joint estimation algorithm.

In Sec. 4, we analyze the independent and joint estimation schemes for their sensitivity to inaccurate disturbance assumptions, cf. (9). In this context, the performance index (28) allows us to compare the performance of the two estimation schemes and to determine in which cases a joint estimation is more beneficial.

## 4. Sensitivity Analysis

In our previous work [6], we studied independent and joint estimation under the assumption that the variances of the individual and common disturbance, $\alpha$ and $\beta$, are known. These values were used when solving for the Kalman gains $K_j$. They also served as initial conditions when calculating the variance of the disturbance estimate (23) in each iteration $j$.

Below, the effects of incorrect variance assumptions on the performance of the estimation algorithm are studied for both independent and joint estimation, and compared with the result derived in [6]. This analysis allows us to deduce rules on how to choose $\bar{\alpha}$ and $\bar{\beta}$ in order to achieve robustness to assumption errors.

### 4.1 Variance of Disturbance Estimate

We derive the individual's disturbance variance $p_j^{(1,1)}$ given the assumptions in Sec. 2.3. In this context, we introduce the notation ${}^*(\cdot)$ to represent a respective quantity assuming perfect knowledge, i.e.

$$\bar{\varepsilon} := \varepsilon \,. \tag{29}$$

The ideal value of the individual's disturbance variance was derived in [6],

$${}^*p_j^{(1,1)} = \frac{\alpha+\beta+j\beta^2+jN\alpha\beta}{(1+j\beta)\,(1+j\beta+jN\alpha)}\,, \tag{30}$$

under the assumption that the disturbance characteristics of $d^{i,ind}$ and $d^0$ are known, i.e. $\delta = 0$. For the following sensitivity analysis, we express (30) in terms of the total disturbance variance $\gamma$, see (11), and the disturbance ratio $\varepsilon$,

$${}^*p_j^{(1,1)} = \frac{\gamma+j\gamma^2\,(1-\varepsilon)\,(1+\varepsilon\,(N-1))}{m_j(\gamma,\varepsilon)\,n_j(\gamma,\varepsilon,N)}\,, \tag{31}$$

$$:= f_j\,(\gamma,\varepsilon,N) \tag{32}$$

where

$$\begin{aligned} m_j(\gamma,\varepsilon) &= 1+j\gamma(1-\varepsilon) \\ n_j(\gamma,\varepsilon,N) &= 1+j\gamma(1+\varepsilon\,(N-1))\,. \end{aligned} \tag{33}$$

The variance ${}^*p_j^{(1,1)}$ serves as reference value for determining the robustness of the joint estimation scheme to an incorrect similarity assumption, $\bar{\varepsilon} \neq \varepsilon$.

We derive an analytical expression for the variance of an agent's disturbance estimate for the general case, where $\varepsilon$ is not assumed to be known.

PROPOSITION 1 The variance of an agent's disturbance estimate can be expressed in terms of the combined variance $\gamma$, the nominal disturbance ratio $\bar{\varepsilon}$, the assumption error $\delta$, the number of agents $N$, and the iteration $j$,

$$p_j^{(1,1)} = f_j\,(\gamma,\bar{\varepsilon},N) - \delta\; g_j\,(\gamma,\bar{\varepsilon},N)\,, \tag{34}$$

where

$$g_j(\gamma,\bar{\varepsilon},N) = \frac{j\gamma^2\bar{\varepsilon}\,(N-1)\,(2+j\gamma(2+\bar{\varepsilon}\,(N-2)))}{m_j(\gamma,\bar{\varepsilon})^{\,2}\,n_j(\gamma,\bar{\varepsilon},N)^{\,2}} \tag{35}$$

and $f_j\,(\gamma,\bar{\varepsilon},N)\,, m_j(\gamma,\bar{\varepsilon}), n_j(\gamma,\bar{\varepsilon},N)$ are defined by (32) and (33). Recalling (13), the assumption error is bounded by

$$-\,\bar{\varepsilon} \;\leq\; \delta \;\leq 1-\bar{\varepsilon}\,. \tag{36}$$

■

The result is obtained by first solving the Kalman filter equations (19) for the Kalman gain $K_j$, given the initial conditions (20). Finally, the recursive equation (22) yields the above result, given the starting values (23). A more detailed proof is found in the Appendix A.1.

The goal of the following analysis is to compare a real scenario (where $\bar{\varepsilon} \neq \varepsilon$) to an ideal case, where we have perfect system knowledge ($\bar{\varepsilon} = \varepsilon$).

***Independent Estimation*** If every agent estimates its disturbance $d^i$ independently, a partitioning of $d^i$ into a common part and an individual part, cf. (4), is not necessary. Consequently, an incorrect assumption $\bar{\varepsilon} \neq \varepsilon$ has no effect on the disturbance estimate; that is,

$$p_j^{(1,1)}\big|_{N=1} = {}^{*}p_j^{(1,1)}\big|_{N=1}. \tag{37}$$

This is also reflected by the equations (34)-(35) and (31)-(32), where

$$g_j(\gamma, \bar{\varepsilon}, 1) = 0 \tag{38}$$

and

$$f_j(\gamma, \bar{\varepsilon}, 1) = f_j(\gamma, \varepsilon, 1) = \frac{\gamma}{1 + j\gamma} \tag{39}$$

depends only on the sum $\gamma$, which is assumed to be known precisely, $\gamma = \alpha + \beta = \bar{\alpha} + \bar{\beta}$, see (11).

In the limit case when $j \to \infty$, the variance of the disturbance estimate approaches zero monotonically,

$$\lim_{j\to\infty} p_j^{(1,1)}\big|_{N=1} = 0 \qquad \text{with} \quad \frac{\partial\, p_j^{(1,1)}\big|_{N=1}}{\partial j} \leq 0. \tag{40}$$

Fig. 2 illustrates the evolution of the variance $p_j^{(1,1)}\big|_{N=1}$ for the example introduced in Sec. 5.

***Joint Estimation*** If $N$ agents jointly estimate their disturbance $d^i$, the assumption on the disturbance partitioning is crucial to the estimation performance. We compare the variance $p_j^{(1,1)}$ with the ideal value by using the relation $\varepsilon = \bar{\varepsilon} + \delta$ in (31) and subtracting (31) from (34). This yields

$${}^{*}p_j^{(1,1)} < \; p_j^{(1,1)}, \tag{41}$$

for $N > 1$ and $\bar{\varepsilon} \neq \varepsilon$. The performance of the joint estimation algorithm when assuming perfect knowledge is better (i.e. results in a smaller variance) than in the realistic scenario, where the disturbance partitioning is not accurately known.

When analyzing (34), (35) with respect to the assumption error $\delta$, we obtain

$$\frac{\partial\, p_j^{(1,1)}}{\partial \delta} \leq 0, \qquad \frac{\partial\, p_j^{(1,1)}}{\partial\, \varepsilon} \leq 0. \tag{42}$$

For a given ratio $\bar{\varepsilon}$, the performance of the joint estimation scheme improves (i.e. smaller variance) if the real ratio $\varepsilon$ increases. In other words, no matter how wrong our assumption on the disturbance partitioning $\bar{\varepsilon}$ is, the joint estimation scheme becomes more effective if the agents show an increasing similarity in reality (and as long as $\bar{\varepsilon} \neq 0$). However, if the nominal ratio is chosen to be zero ($\bar{\varepsilon} = 0$, assuming the agents are completely different), the joint variance (34) corresponds to the individual variance (39) and is not improved by the agent's actual similarity, see Fig. 2.

An interesting next step is to study the limit behavior of the disturbance variance for $j \to \infty$. We first consider $f_j(\gamma, \bar{\varepsilon}, N)$ in (34) and state

$$\lim_{j\to\infty} f_j(\gamma, \bar{\varepsilon}, N) = 0 \qquad \text{with} \quad \frac{\partial f_j(\gamma, \bar{\varepsilon}, N)}{\partial j} \leq 0\,. \tag{43}$$

The function $f_j(\gamma, \varepsilon, N)$ also defines the perfect variance and its limit, cf. (30), and hence,

$$\lim_{j\to\infty} {}^{*}p_j^{(1,1)} = 0 \qquad \text{with} \quad \frac{\partial\, {}^{*}p_j^{(1,1)}}{\partial j} \leq 0\,. \tag{44}$$

This means that, in the perfect knowledge case, the disturbance is accurately estimated after a large number of iterations. We keep this in mind when analyzing the limit behavior of the variance $p_j^{(1,1)}$. Two cases are distinguished:

(1) If we assume that the agents are not perfectly identical, that is $\bar{\varepsilon} \neq 1$, the variance converges to zero,

$$\lim_{j\to\infty} p_j^{(1,1)} = 0\,. \tag{45}$$

The joint estimation algorithm provides an increasingly accurate estimate of the disturbance with each additional iteration. Even if our assumption is wrong, and $\delta \neq 0$, the estimation algorithm provides us with the correct disturbance estimate in the limit case when the number of iterations approaches infinity[2].

(2) If we assume that the agents are identical, $\bar{\varepsilon} = 1$, the limit behavior for $j \to \infty$ is

$$\lim_{j\to\infty} p_j^{(1,1)} = -\delta\,\gamma\frac{N-1}{N} \quad := \ell(\gamma, \delta, N)\,, \tag{46}$$

where $-1 \leq \delta \leq 0$, cf. (36). The variance of the disturbance estimate has a finite, nonzero limit value (if $\delta \neq 0$, which is equivalent to saying $\varepsilon \neq 1$). In other words, when we assume the agents are identical and they are (in fact) not, the joint estimation scheme does not provide an accurate estimate, even after a large number of iterations. The variance in the limit $j \to \infty$ depends on the total disturbance level $\gamma$, the number of agents $N$, and the assumption error $\delta$, where

$$\frac{\partial\,\ell(\gamma, \delta, N)}{\partial N} \geq 0, \qquad \frac{\partial\,\ell(\gamma, \delta, N)}{\partial\,\delta} \leq 0, \qquad \frac{\partial\,\ell(\gamma, \delta, N)}{\partial\,\gamma} \geq 0\,. \tag{47}$$

The limiting variance grows with an increasing number of agents, an increasing difference $|\,\delta\,|$ between the assumed and real ratio (note that $\delta \leq 0$), and an increasing overall disturbance level. When $\varepsilon = 0$ (meaning that the agents, in reality, have no common disturbance component), the difference between the assumed and real ratio is largest, $\delta = -1$, and

$$\lim_{N\to\infty} \ell(\gamma, -1, N) = \gamma. \tag{48}$$

[2] Mathematica files including the presented results are available at `www.idsc.ethz.ch/Downloads/multiagentILC`.

Based on the above results, first conclusions can be drawn on how our assumption on the agents' similarity affects the joint estimate, and on how to choose the nominal ratio $\bar{\varepsilon}$:

- Generally, an incorrect assumption about the agents' similarity ($\bar{\varepsilon} \neq \varepsilon$) decreases the accuracy of the disturbance estimate and increases the variance of the disturbance estimate, cf. (41).
- In the worst case, the variance does not approach zero in the limit when the number of iterations goes to infinity. This happens if we assume that the agents are perfectly identical, $\bar{\varepsilon} = 1$, but they are, in fact, not. In this case, the limit value of the variance for $j \to \infty$ increases with the number of agents and with an increasing total disturbance. The variance is worst if the agents share no common disturbance component in the real scenario, $\varepsilon = 0$ or $\delta = -1$. *To assure that the variance converges to zero for any ratio $\varepsilon$, we must choose $\bar{\varepsilon} \neq 1$.*
- As shown in (42), the joint estimation scheme is more beneficial if the agents are more similar in reality. This is independent of the assumed value and holds for all $\bar{\varepsilon} \neq 0$. Hence, if we want to benefit from this characteristic of the joint estimation scheme, and if we expect a certain degree of similarity between the agents, the ratio $\bar{\varepsilon}$ should not be set to zero.

Fig. 2 summarizes the characteristics derived above: the perfect knowledge case results in the smallest variance values and, in the limit case, variances approach zero except for the case $\bar{\varepsilon} = 1$, where the limit value is obtained from (46), cf. Sec. 5.

Thus far we have compared the disturbance estimate $p_j^{(1,1)}$ with the perfect value ${}^*p_j^{(1,1)}$. What remains is to compare the independent and joint estimation schemes given an insufficient knowledge $\bar{\varepsilon} \neq \varepsilon$. In the following section, we attempt to determine whether, in light of our new findings, the joint estimation continues to perform better than the independent estimation.

### 4.2 Performance Index

We introduced the performance index $R$ as the ratio of the state variance in the independent case vs. the joint case, see Sec. 3.2 and [6]. As shown in [6], in the nominal case, assuming the real values $\alpha$, $\beta$ are known, joint estimation is always beneficial and yields a performance index

$$1 \leq {}^*R = \frac{{}^*p_j^{(1,1)}\big|_{N=1} + \lambda}{{}^*p_j^{(1,1)} \qquad + \lambda}. \tag{49}$$

When the number of iterations increases, the performance index shows the following limit behavior,

$$\lim_{j\to\infty} {}^*R = \begin{cases} N & \text{for } \varepsilon = 1 \text{ and } \lambda = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{50}$$

If the agents are not identical ($\varepsilon \neq 1$), the performance index ${}^*R$ converges to one. The same limit behavior is observed if process noise acts on the system, $\lambda \neq 0$. Only if (i) the agents are identical, and (ii) noise is due to measurement only, is the limit value of the performance index equal to $N$, see [6] for a detailed analysis.

From Sec. 4.1 we know that a mismatch between the real disturbance characteristics and the nominal values does not affect the independent estimate, see (37), but it does corrupt the disturbance estimate when jointly estimating, see (41). In this context, it is interesting to ask: Is it possible that the performance index becomes smaller than one, meaning that the individual estimation performs better than the joint estimation? Taking into account the disturbance variances derived in Sec. 4.1, we answer this question below.

First, we compare the performance index in the ideal case, where $\bar{\varepsilon} = \varepsilon$, with the performance index $R$ of the realistic scenario, $\bar{\varepsilon} \neq \varepsilon$. Given the definition of the performance index in (28) and the derived characteristics of the disturbance variance, cf. (37) and (41), we conclude that

$$R < {}^{*}R\,. \tag{51}$$

The ideal performance index represents an upper bound to $R$ and, recalling (49), is larger or equal to one.

Second, before studying the evolution of $R$ with $j$, we focus on the limiting behavior of performance index $R$ as $j \to \infty$. Taking into account the limit values of the disturbance variance derived in Sec. 4.1 and explicitly stated in (40), (45) and (46), the following statement is derived:

LEMMA 1 As $j \to \infty$, the performance index $R$, defined by (28) and (34), shows two distinct limiting behaviors:

(1) If $\bar{\varepsilon} \neq 1$, the performance index converges to one,

$$\lim_{j\to\infty} R = 1 \tag{52}$$

for all possible values $N, j > 1$, $\gamma \geq 0$, and $\varepsilon, \lambda \in [0,1]$, where $0 \leq \bar{\varepsilon} < 1$.

(2) If $\bar{\varepsilon} = 1$, the limit behavior for $j \to \infty$ is

$$\lim_{j\to\infty} R \;=\; \frac{\lambda N}{\lambda N - \delta\gamma(N-1)} \;:=\; \ell_R(\gamma, N, \delta, \lambda) \qquad \leq 1\,, \tag{53}$$

for all possible values $N, j > 1$, $\gamma > 0$, and $\lambda \in [0,1]$, where $-1 \leq \delta < 0$.

Since

$$\frac{\partial\, \ell_R}{\partial N} \leq 0 \qquad \text{and} \qquad \frac{\partial\, \ell_R}{\partial\, \delta} \geq 0, \tag{54}$$

the limit value $\ell_R$ reaches its minimum for $\delta = -1 \;\Leftrightarrow\; \varepsilon = 0$, and $N \to \infty$. In this case,

$$\lim_{N\to\infty} \ell_R(\gamma, N, -1, \lambda) = \frac{\lambda}{\lambda + \gamma}\,. \tag{55}$$

■

***Interpretation of the result*** If we assume the agents are not identical, $\bar{\varepsilon} \neq 1$, the joint estimation algorithm converges and provides us with an accurate disturbance estimate for $j \to \infty$, cf. (45) and (52). In other words, with respect to the convergence of the disturbance variance, the joint estimation scheme is robust to assumption errors as long as $\bar{\varepsilon} \neq 1$. If we choose $\bar{\varepsilon} = 1$ and $\delta \neq 0$, however, we lose this property, cf. (46) and (55). That is, under the assumption that all agents are identical, the joint estimation algorithm is highly sensitive to assumption errors. In the case of pure measurement noise $\lambda = 0$, (53) is zero. Note that the case, $\lambda = 0$ and $\bar{\varepsilon} = 1$, yields the best performance improvement in the ideal case, see (50), but is the most sensitive to assumption errors. *In brief, when building upon a joint estimation scheme, the disturbance ratio $\bar{\varepsilon}$ should not be set to one.* Moreover, we have shown that the performance index can be less than one (see statement (2) in Lemma 1). In these cases, an independent estimation is preferable.

The above result guarantees that we eventually obtain a precise estimate of the disturbance, but it does not provide insight into the transient performance of the joint estimation scheme as compared to an independent algorithm.

As a last step, we perform a more detailed analysis and identify parameter combinations that support an application of the joint estimation scheme (where $R > 1$). At an iteration $j$, the joint estimation is beneficial, if the inequality $R \geq 1$ is satisfied, which is equivalent to saying

$$f_j(\gamma, \bar{\varepsilon}, 1) \;\geq\; f_j(\gamma, \bar{\varepsilon}, N) - \delta\, g_j(\gamma, \bar{\varepsilon}, N) \tag{56}$$

for a given number of agents $N$, a known nominal disturbance ratio $\bar{\varepsilon}$, and an overall disturbance $\gamma$, see (28) and (34). The functions $f_j$ and $g_j$ are non-negative for all possible values $N, j \geq 1$, $\gamma \geq 0$, and $\bar{\varepsilon} \in [0,1]$, cf. (32) and (35),

$$f_j(\gamma, \bar{\varepsilon}, N) \geq 0 \qquad \text{and} \qquad g_j(\gamma, \bar{\varepsilon}, N) \geq 0\,, \tag{57}$$

and

$$\frac{\partial f_j(\gamma, \bar{\varepsilon}, N)}{\partial N} = \frac{-\bar{\varepsilon}^2 \gamma^2 j}{m_j(\gamma, \varepsilon)\, n_j(\gamma, \varepsilon, N)^2} \leq 0\,. \tag{58}$$

Combining (56) with (57),(58) yields the following lemma:

LEMMA 2 A sufficient condition for the joint estimation to yield a better (or equal) learning performance than the independent estimation is a disturbance ratio $\varepsilon$ that is larger than (or equal to) the assumed one. The following implication holds,

$$\varepsilon \geq \bar{\varepsilon} \qquad \Rightarrow \qquad R \geq 1\,, \tag{59}$$

for all possible values $N, j \geq 1$, $\gamma \geq 0$, and $\bar{\varepsilon} \in [0,1]$. ■

***Interpretation of the result*** If the agents are more similar in reality than assumed, it is beneficial to jointly estimate the repetitive disturbances. We should avoid an overestimation of the similarity between the agents, since in this case an independent scheme would actually be more

beneficial. However, if we underestimate the similarity for a given situation defined by $\varepsilon$, we increase the variance of the disturbance estimate, cf. (42). Moreover, from (28) and (34) with $\delta = \varepsilon - \bar{\varepsilon}$, we obtain

$$\frac{\partial R}{\partial \bar{\varepsilon}} \geq 0 \qquad \text{if} \quad \varepsilon \geq \bar{\varepsilon}\,, \tag{60}$$

which means that, if we underestimate the similarity of the agents (reducing $\bar{\varepsilon}$), we reduce the benefit of the joint estimation vs. the independent estimation.[2] *A design rule for $\bar{\varepsilon}$ is consequently: Given a priori knowledge about the multi-agent system, make $\bar{\varepsilon}$ as large as possible while, at the same time, ensuring that it is less than the real value $\varepsilon$.* In this case, applying the joint estimation guarantees a better learning performance than the independent estimation scheme; however, the benefits of joint estimation remain marginal, as shown in [6].

Note that if $\delta \geq 0 \;\Leftrightarrow\; \varepsilon \leq \bar{\varepsilon}$, there are also cases for which $R \geq 1$, see Fig. 3. From (56) with (13), we derive the necessary and sufficient condition[2]:

LEMMA 3 In the proposed multi-agent learning framework, joint estimation yields a better (or equal) learning performance as the independent estimation *if and only if* the following inequality is satisfied for given values $N, j > 1,\ \gamma, \bar{\varepsilon} > 0$ and $\varepsilon$:

$$\varepsilon \;\geq\; \bar{\varepsilon}\,\frac{1+\gamma j\left(2+\gamma j\left(1+\bar{\varepsilon}^2\,(N-1)\right)\right)}{(1+\gamma j)\,(2+\gamma j\,(2+\bar{\varepsilon}\,(N-2)))} \;:=\; \bar{\varepsilon}\,h_j(\gamma,\bar{\varepsilon},N) \tag{61}$$

with $h_j(\gamma,\bar{\varepsilon},N) < 1$. The inequality becomes less restrictive for an increasing number of agents, $\partial h_j/\partial N < 0$, and reduces to:

$$\varepsilon \;\geq\; \bar{\varepsilon}\,\frac{\bar{\varepsilon}\gamma j}{1+\gamma j} \qquad \text{for} \quad N \to \infty\,, \tag{62}$$

and, in the limit case for $j \to \infty$, to:

$$\varepsilon \;\geq\; \bar{\varepsilon}^2 \qquad \text{for} \quad j, N \to \infty\,. \tag{63}$$

■

Fig. 3 illustrates the previously derived characteristics for $\varepsilon = 0.5$ and the nominal values $\bar{\varepsilon}$ ranging from 0 to 1. Note that for the case $\bar{\varepsilon} = 0.75$, the performance index crosses the $R = 1$ line and finally approaches one.

The previous analysis focused on identifying the cases for which the joint estimation is beneficial despite an incorrect assumption $\bar{\varepsilon} \neq \varepsilon$ and, in turn, defined the cases where an incorrect assumption $\bar{\varepsilon} \neq \varepsilon$ corrupts the performance of the joint estimation to such an extent that an independent estimation is more effective. The numerical examples below summarize the results of this section and highlight the sensitivity of the joint estimation scheme to incorrect noise assumptions.

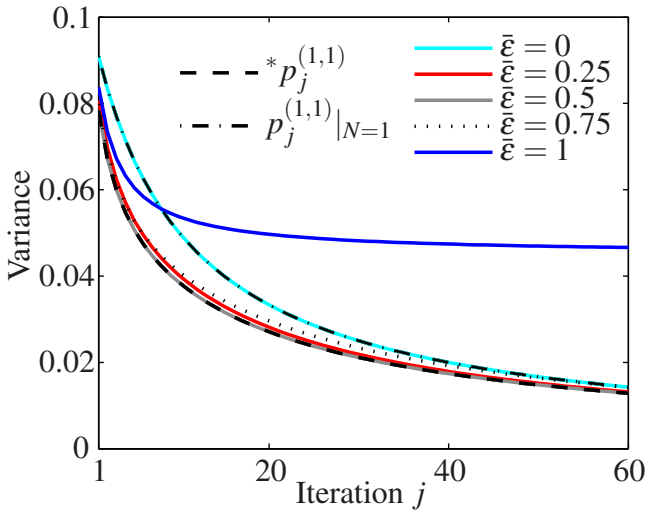



**Figure 2.** The variance of an individual's disturbance estimate for $N = 10$ agents, $\varepsilon = 0.5$, $\gamma = 0.1$ and $\lambda = 0$ (pure measurement noise).

## 5. Numerical Examples

We consider a group of $N = 10$ agents with a similarity of $\varepsilon = 0.5$. The noise is due to measurement only ($\lambda = 0$) and the overall disturbance $\gamma$ is 0.1. Fig. 2 and Fig. 3 show the evolution of the variance and the performance index for various nominal values $\bar{\varepsilon}$ ranging from 0 to 1. Note that different intervals of $j$ are chosen in Fig. 2 and Fig. 3 to emphasize the main characteristics. Both figures show that an incorrect similarity assumption results in a worse performance, i.e. in a higher variance in Fig. 2 and a lower performance index in Fig. 3. The limiting behavior in the case of $\bar{\varepsilon} = 1$, see (46) and (53), is for the given scenario

$$\lim_{j\to\infty} p_j^{(1,1)} = 0.5 \cdot 0.1 \cdot \frac{9}{10} = 0.045 \qquad \text{and} \qquad \lim_{j\to\infty} R = 0\,.$$

In all other cases, the variance approaches 0 (see Fig. 2) and the performance index 1 (see Fig. 3) as $j \to \infty$. For the two cases where $\bar{\varepsilon} > \varepsilon$, the performance index is (partly) smaller than one.

## 6. Conclusions

We analyzed the sensitivity of joint estimation to the underlying assumption of similarity between agents. The analysis was driven by our previous results, which showed that the learning performance of an agent is improved by exchanging information with other agents that are learning the same task.

While previous results assumed perfect knowledge about the degree of similarity between the agents, this paper studied the effects of an incorrect similarity assumption. We found that an incorrect assumption not only degrades the performance of the joint estimation scheme (when

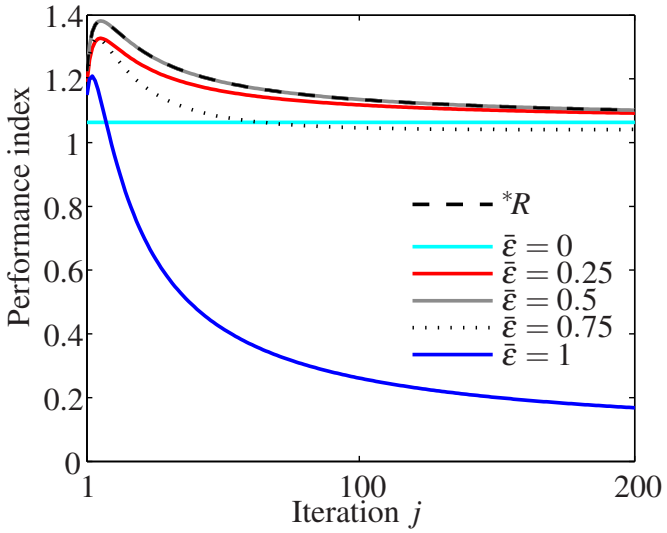



**Figure 3.** The performance index for $N = 10$ agents, $\varepsilon = 0.5$, $\gamma = 0.1$ and $\lambda = 0$ (pure measurement noise).

compared to the case of perfect knowledge), but that, for some problems, the joint estimation performs worse than an independent estimation scheme. This is particularly true if we overestimate the similarity between the agents. As a consequence, it is not advisable to assume that agents are identical because, if agents are not identical in reality, the joint estimation does not even converge to the correct disturbance value. However, note that from our previous analysis, we know that the case of identical agents (and no process noise) provided the best performance improvement of joint vs. independent estimation – an improvement of a factor equal to the number of agents. In other words, the case with the highest performance improvement (due to joint estimation) shows the highest sensitivity to assumption errors.

*To conclude: in order to guarantee improved performance over an individual learning scheme, a joint estimation scheme must not overestimate the similarity between the agents.*

# A. Appendix

## A.1 Proof of Proposition 1

We derive an explicit representation of the variance $p_j^{(1,1)}$, as presented in Proposition 1, that depends on the total variance $\gamma$, the nominal similarity factor $\bar{\varepsilon}$, the assumption error $\delta$, the iteration $j$, and the number of agents $N$.

The proof proceeds similarly to the derivation of the ideal variance (31) in [6]. Note that in [6], we assumed that the real noise characteristics (9) were known precisely and

$$p_j^{(1,1)} \text{ simply denoted the ideal variance } {}^*p_j^{(1,1)}. \tag{64}$$

Matlab and Mathematica files for reproducing the results below are available on the project webpage[3].

*Proof.* For the general case, we derive a closed form of the covariance matrix $P_j$, with the assumptions in Sec. 2.3. Since the disturbance and noise characteristics are the same for each agent, the covariance matrix is of the following symmetric structure,

$$p_j^{(k,l)} = \begin{cases} p_j^{(0,0)} & \text{if} \quad k = l = 0 \\ p_j^{(0,1)} & \text{if} \quad kl = 0 \text{ and } k \neq l \\ p_j^{(1,1)} & \text{if} \quad k = l \neq 0 \\ p_j^{(1,2)} & \text{otherwise}. \end{cases} \tag{65}$$

We derive a recursive relationship for the values in (65) based on (21) and the closed-form representation of $K_j$ (derived in [6]). The Kalman gains $K_j$ are calculated based on the nominal disturbance variances $\bar{\alpha} = \bar{\varepsilon}\gamma$ and $\bar{\beta} = (1-\bar{\varepsilon})\gamma$ and are given by

$$k_j^{(k,l)} = \begin{cases} k_j^{(0,1)} & \text{if} \quad k = 0 \\ k_j^{(1,1)} & \text{if} \quad k = l \\ k_j^{(1,2)} & \text{otherwise}, \end{cases} \tag{66}$$

where

$$k_j^{(0,1)} = \frac{\bar{\varepsilon}\gamma}{n_j(\gamma,\bar{\varepsilon},N)}, \qquad k_j^{(1,2)} = \frac{\bar{\varepsilon}\gamma}{m_j(\gamma,\bar{\varepsilon})n_j(\gamma,\bar{\varepsilon},N)}, \qquad k_j^{(1,1)} = f_j(\gamma,\bar{\varepsilon},N). \tag{67}$$

From (21) with (65) and (67), we obtain recursive equations for

$$p_j^{(0,0)}, \quad p_j^{(0,1)}, \quad p_j^{(1,1)} \quad p_j^{(1,1)}, \tag{68}$$

that depend only on the $(j-1)$th values of (68), on the Kalman gains (67) and the number of agents $N$. A proof by induction using the recursive equations for (68) with initial condition (23) verifies the closed-form expression in (34). For the proof, the closed-form representations of all values (68) were needed and derived. However, the only value of interest is $p_j^{(1,1)}$.

Mathematica files with the recursive equations for (68) and the closed-from representations of all quantities (68) are available on the project webpage[3]. □

## Acknowledgements

The authors would like to thank Javier Alonso-Mora for many fruitful discussions.

[3] Mathematica files including the presented results are available at `www.idsc.ethz.ch/Downloads/multiagentILC`.

## References

[1] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[2] A. Schaerf, Y. Shoham, and M. Tennenholtz, "Adaptive load balancing: a study in multi-agent learning," *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 475–500, 1994.

[3] M. Matarić, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, vol. 4, pp. 73–83, 1997.

[4] S. Mannor and J. Shamma, "Multi-agent learning for engineers," *Artificial Intelligence*, vol. 171, no. 7, pp. 417–422, 2007.

[5] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[6] A. Schöllig, J. Alonso-Mora, and R. D'Andrea, "Independent vs. joint estimation in multi-agent iterative learning control," in *Proceedings of the 49th IEEE Conference on Decision and Control*, 2010, pp. 6949–6954.

[7] Z. Bien and J. Xu, *Iterative learning control: analysis, design, integration and applications*. Kluwer Academic Publishers Norwell, MA, USA, 1998.

[8] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[9] M. Phan and R. Longman, "Higher-order iterative learning control by pole placement and noise filtering," in *Proceedings of the IFAC World Congress*, 2002, pp. 1899–1904.

[10] M. Norrlöf, "Disturbance rejection using an ILC algorithm with iteration varying filters," *Asian Journal of Control*, vol. 6, pp. 432–438, 2004.

[11] A. Schöllig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European Control Conference*, Budapest, Hungary, 2009, pp. 1505–1510.

[12] M. Phan and R. Longman, "A mathematical theory of learning control for linear discrete multivariable systems," in *Proceedings of the AIAA/AAS Astrodynamics Conference*, 1988, pp. 740–746.

[13] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 5, 2001, pp. 4463–4468.

[14] B. Bamieh, J. Pearson, B. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[15] J. Hätönen, D. Owens, and K. Feng, "Basis functions and parameter optimisation in high-order iterative learning control," *Automatica*, vol. 42, no. 2, pp. 287–294, 2006.

[16] K. Barton and A. Alleyne, "A cross-coupled iterative learning control design for precision motion control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1218–1231, 2008.

[17] M. Butcher, A. Karimi, and R. Longchamp, "Iterative learning control based on stochastic approximation," in *Proceedings of the 17th IFAC World Congress*, 2008, pp. 1478–1483.

[18] C. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications (Springer Series in Information Sciences)*. Springer, 1998.

[19] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*. Cambridge University Press, 2007.

# Part B

# RHYTHMIC FLIGHT PERFORMANCES

# Paper V

# Synchronizing the Motion of a Quadrocopter to Music

Angela P. Schoellig · Federico Augugliaro · Sergei Lupashin · Raffaello D'Andrea

**Abstract**

This paper presents a quadrocopter flying in rhythm to music. The quadrocopter performs a periodic side-to-side motion in time to a musical beat. Underlying controllers are designed that stabilize the vehicle and produce a swinging motion. Synchronization is then achieved by using concepts from phase-locked loops. A phase comparator combined with a correction algorithm eliminate the phase error between the music reference and the actual quadrocopter motion. Experimental results show fast and effective synchronization that is robust to sudden changes in the reference amplitude and frequency. Changes in frequency and amplitude are tracked precisely when adding an additional feedforward component, based on an experimentally determined look-up table.

Published in *Proc. of the 2010 IEEE International Conference on Robotics and Automation (ICRA),* 2010. Submission includes multimedia attachment, also found at `www.tiny.cc/QuadroDance`. DOI: 10.1109/ROBOT.2010.5509755.

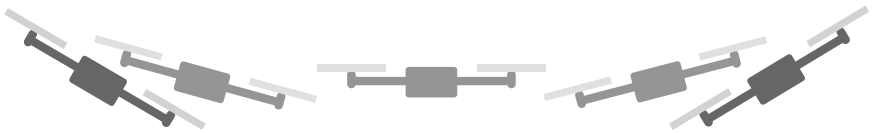


**Figure 1.** The desired side-to-side motion.

## 1. Introduction

Controls for the synchronization of movement are indispensable in any robotics application where high performance, precision, and agility are required. Involving a coordination in time between two or more systems or events, controls for synchronized behavior fall into two broad categories: algorithms focused on coordinating internal signals (for example, from multiple dynamic subsystems), and algorithms focused on coordinating with external inputs (from other bodies or the environment).

Driven by the need to control the movement of complex industrial robots, early synchronized control algorithms dealt with coordinating internal signals from multiple dynamic subsystems. Examples include multi-axis machine tools [1,2], parallel manipulators [3], and multi-robot assembling machines [4]. More recently, motion synchronization has led, for example, to the development of algorithms for spacecraft formation flying [5] and is used in mobile robots to reduce the velocity errors between driving wheels [6].

Control algorithms for synchronizing robots with external inputs have been largely developed within the field of humanoid robotics, where researchers aim for a lively interaction between robots and their environment. Examples of motion synchronization with external inputs have thus far dealt mostly with easily repeatable motion primitives like steps or up-and-down arm movements, including robots that step and sing along to music [7–10] or drum in tempo with an exogenous signal [11–13]. Periodic inputs and rhythmic robot movements are often featured in these applications for their simplicity, repetitiveness, aesthetic pleasure, and entertainment value.

The objective of this paper is to synchronize the motion of a quadrocopter to music. The vehicle's nonlinear and unstable dynamics present significant challenges in motion synchronization. Stabilizing control is required just to keep the vehicle in the air, and modeling errors, motor saturation, and communication delays have noticeable effects on the quadrocopter dynamics. An appropriate synchronization algorithm is indispensable to time precisely the response of the vehicle with the music reference. Note that this is not the case in most other approaches dealing with synchronizing rhythmic movement. In digital animation the motion of a character is not affected by mass inertia or system delay, but is directly dictated by the developer, eliminating the need for a synchronization algorithm. Examples of virtual dancing characters are found in [14, 15]. Other papers on dancing robots, as summarized in [16], do not focus on synchronization either, since they deal with systems that are better understood, less sensitive to disturbances, and, generally, easier to manage. The *Keepon Robot* [17] and *Ms*

*DanceR* [18], which dance with humans, are two such examples.

In this paper, a simple motion primitive is chosen for studying the feasibility of our idea. The quadrocopter undergoes a planar side-to-side motion as depicted in Fig. 1, where at beat times the vehicle reaches the outermost points of the trajectory, either on the left or right. In a preliminary step, the music is *pre-processed* and the beat times obtained are transformed into a periodic signal which is used as a reference trajectory for the quadrocopter. Fig. 2 shows the overall control system. Synchronized rhythmic behavior is achieved if the music reference signal and the actual quadrocopter motion are in phase, cf. Fig. 3.

The general idea is borrowed from phase-locked loops (PLL). A phase-locked loop acts on the frequency of a controlled oscillator and matches its output to a periodic reference signal both in frequency and phase. PLLs are widely used in radio, telecommunications, computers, and many other electronic applications, cf. [19, 20]. Inspired by this concept, underlying controllers are designed that turn the unstable quadrocopter dynamics into an oscillating system behavior. A phase comparator detects the phase error between desired reference trajectory and quadrocopter motion. After having determined the phase error, it is compensated for by closing the loop on the phase error, similar to [9] and [12]. The efficacy of the proposed synchronization algorithm is experimentally studied. The derived algorithm proves to be able to accurately coordinate the movements of the flying vehicle with the desired music reference. The accompanying video shows the 'dancing' quadrocopter.

In the following sections, the overall control system is presented. Section 2 explains how the reference signal is generated from the music signal and introduces the quadrocopter dynamics. Section 3 describes how the required oscillating system is realized by using controllers for the side-to-side motion, while Section 4 presents the phase detection and correction step which cause the desired synchronized behavior. Experimental results complete this work.

## 2. System Representation

This section introduces the desired quadrocopter motion as extracted from the music and presents a two-dimensional model of the quadrocopter that is used for the controller design.

### 2.1 Periodic Motion

The goal of this work is to show a quadrocopter flying in rhythm to music, where the main focus lies on solving the underlying synchronization problem. A simple motion primitive was selected as reference trajectory for the vehicle with the goal of being able to visualize the existing phase error and successful phase locking. The quadrocopter performs a planar side-to-side motion where beats occur at the outermost positions, see Fig. 1. The amplitude and frequency of this lateral motion are modulated by the music's melody and beat intervals, respectively.

For the derivations, a constant amplitude and constant beat interval are considered; that is, music beats occur with a constant frequency. This is a reasonable assumption for many types of music. The efficacy of the derived algorithm for more complicated scenarios is shown in Section 5.3.

Analyzing a a piece of music yields a desired beat interval $T$ and amplitude $A_d$. These values define the reference trajectory that is fed to the quadrocopter. As depicted in Fig. 1, the

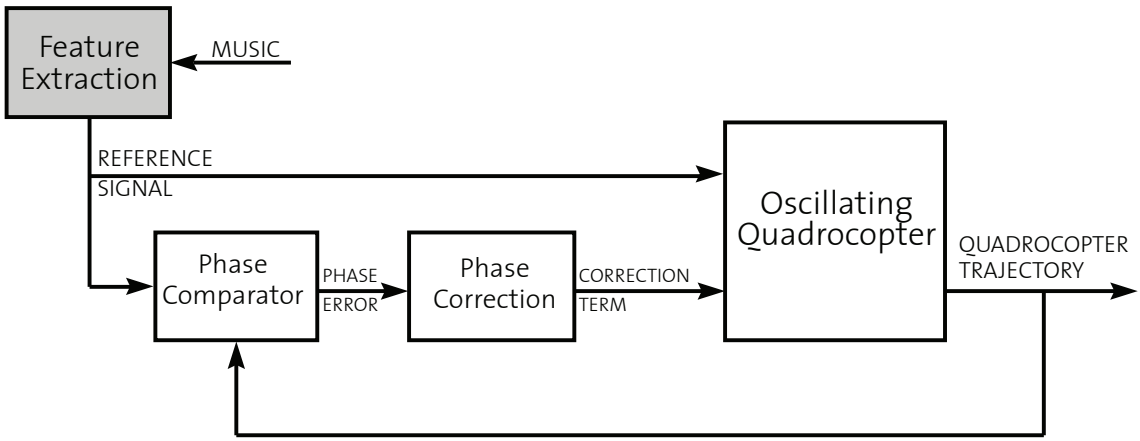



**Figure 2.** The overall control system transforming music into an appropriate quadrocopter motion.

corresponding desired vehicle trajectory is a sinusoidal side-to-side motion in the $xz-$plane,

$$x_d(t) = A_d \cos(\omega_d t), \qquad z_d(t) = z_d = \text{constant}, \tag{1}$$

with $\omega_d = \pi/T$. Fig. 3 illustrates the beat-motion relation. Beats occur at the peaks of the trajectory, i.e. two times per period. As an example, if the music tempo is 120 beats per minute, the desired frequency of the oscillating trajectory is $\omega_d = 2\pi$ rad/s. The altitude of the quadrocopter is stabilized at a given height $z_d$.

### 2.2 Quadrocopter Model

The side-to-side motion (1) is defined in the $xz-$plane. In-plane and out-of-plane dynamics are thus decoupled and additional degrees of freedom are separately stabilized. A simplified two-dimensional model of the quadrocopter is depicted in Fig. 4. The equations governing the dynamics of the system are given by

$$\ddot{z}(t) = f(t)\cos\theta(t) - g \tag{2}$$

$$\ddot{x}(t) = f(t)\sin\theta(t) \tag{3}$$

$$\dot{\theta}(t) = u(t), \tag{4}$$

where $g$ is the gravitational constant and $\theta(t)$ is the pitch angle. The inputs to the system are the normalized thrust $f(t)$ in m/s$^2$ and the pitch rate $u(t)$ in rad/s.

## 3. Controller Design

The oscillating quadrocopter motion is achieved by a cascaded controller design: the $z$-direction is stabilized first and, assuming a constant height, the trajectory-tracking controller for the $x$-direction is designed.

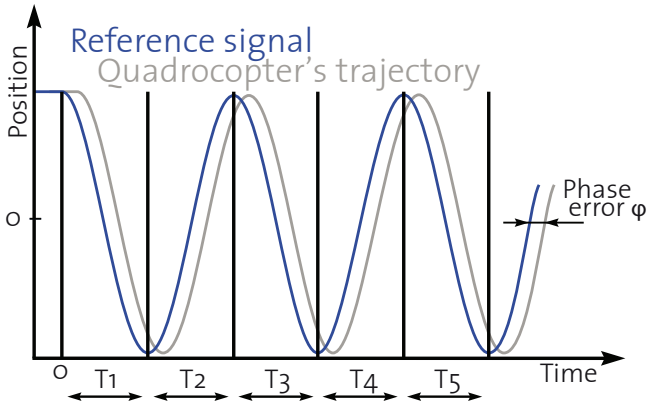



**Figure 3.** The desired rhythmic side-to-side motion of the quadrocopter. Vertical lines represent beat times.

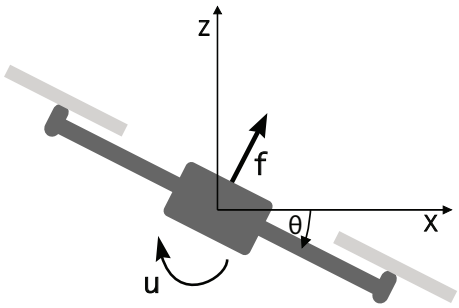



**Figure 4.** Schematic of the 2D quadrocopter model.

### 3.1 Height Stabilization

In order to stabilize the quadrocopter at a desired altitude $z_d$, the thrust input $f(t)$ in (2) is chosen such that a linear second-order system is obtained. With

$$f(t) = \frac{1}{\cos\theta(t)}\Big(g - 2\delta_z\omega_z\,\dot{z}(t) - \omega_z^2\,(z(t) - z_d)\Big), \tag{5}$$

the closed-loop dynamics are given by

$$\ddot{z}(t) + 2\delta_z\omega_z\,\dot{z}(t) + \omega_z^2\,z(t) = \omega_z^2\,z_d. \tag{6}$$

In most cases, a damping ratio $\delta_z$ between 0.7 (underdamped case) and 1 (critically damped case) is a reasonable choice, see [21]. The only remaining design parameter is the natural frequency $\omega_z$.

### 3.2 Trajectory Tracking

Building upon the above control scheme and assuming a constant height $z_d$, the $x$-dynamics (3) reduce to

$$\ddot{x}(t) = g\tan\theta(t). \tag{7}$$

In addition, the pitch angle $\theta(t)$ is assumed to be small. This is a good approximation for reference trajectories with a small frequency $\omega_d$ as compared to the desired amplitude $A_d$. To first order, $\tan\theta(t) = \theta(t)$, resulting in a linear approximation for the sideways dynamics,

$$\dddot{x}(t) = g\,\dot{\theta}(t) = g\,u(t), \tag{8}$$

relating the position $x(t)$ directly to the angle rate input $u(t)$.

With the aim of following the desired sinusoidal side-to-side motion (1), the input $\overline{u}(t) = gu(t)$ is composed of a feedforward component,

$$\overline{u}_1(t) = \dddot{x}_d(t) = A_d\omega_d^3\sin(\omega_d t), \tag{9}$$

and an additional feedback term to correct for errors,

$$\overline{u}_2(t) = \alpha\left(\ddot{x}_d(t) - \ddot{x}(t)\right) + \beta\left(\dot{x}_d(t) - \dot{x}(t)\right) + \gamma\left(x_d(t) - x(t)\right), \tag{10}$$

where the control parameters $\alpha$, $\beta$, and $\gamma$ are defined through

$$\alpha = \omega_x(1+2\delta_x), \qquad \beta = \omega_x^2(1+2\delta_x), \qquad \gamma = \omega_x^3 \tag{11}$$

and act on the acceleration, velocity, and position errors, respectively. With the choice (11), the characteristic polynomial of the closed-loop system is

$$(s+\omega_x)(s^2 + 2\delta_x\omega_x s + \omega_x^2) = 0. \tag{12}$$

The damping ratio $\delta_x$ is again chosen to be a value between 0.7 and 1, and $w_x$ remains the only parameter to be chosen in order to achieve satisfactory tracking performance. Finally, the input

$$u(t) = \frac{1}{g}\left(\overline{u}_1(t) + \overline{u}_2(t)\right) \tag{13}$$

is applied to the quadrocopter.

## 4. Synchronization

When applying the input $u(t)$ as defined in (13), a phase shift is noticed between the reference trajectory of the sideways motion and the actual quadrocopter trajectory, illustrated in Fig. 3. (Corresponding experimental results are shown in Fig. 6 and Fig. 7.) This phenomenon results mainly from unmodeled dynamics (for example communication delays and the propeller dynamics), which were neglected in the controller design presented in Section 3. (We refer to Section 5.1 for more details of the vehicle dynamics.)

### 4.1 Phase Comparator

The phase shift $\varphi(t)$ between the quadrocopter trajectory and the desired motion (1) is determined by multiplying the quadrocopter output separately with two different sinusoidal reference signals and integrating the product. Define the reference signals,

$$r_{cos}(t) = \cos(\omega_d t), \qquad r_{sin}(t) = \sin(\omega_d t), \tag{14}$$

where $r_{cos}(t)$ is the same frequency and phase as the desired motion (1). Under the assumption that the vehicle dynamics are linear, the response of the controlled quadrocopter system (derived in Section 3) to the periodic reference signal $x_d(t)$, see (1), is a sinusoidal signal with the same frequency but possibly shifted phase and different amplitude,

$$x(t) = A\ \cos(\omega_d t + \varphi_t). \tag{15}$$

Multiplying the signals (14) with the vehicle output (15) and using trigonometric identities lead to

$$\begin{aligned} q_{cos}(t) &= x(t) r_{cos}(t) = \frac{A}{2}\left[\cos\varphi(t) + \cos\big(2\omega_d t + \varphi(t)\big)\right] \\ q_{sin}(t) &= x(t) r_{sin}(t) = \frac{A}{2}\left[-\sin\varphi(t) + \sin\big(2\omega_d t + \varphi(t)\big)\right]. \end{aligned}$$

Integrating these signals over one period $T_d = 2\pi/\omega_d$ and assuming a constant phase shift during that time interval

$$\varphi(\tau) = \varphi_t = \text{constant}, \qquad t - T_d \le \tau \le t, \tag{16}$$

results in

$$\eta_1(t) = \frac{1}{T_d}\int_{t-T_d}^{t} q_{cos}(t)\ dt = \frac{A}{2}\cos\varphi_t \tag{17}$$

$$\eta_2(t) = \frac{1}{T_d}\int_{t-T_d}^{t} q_{sin}(t)\ dt = -\frac{A}{2}\sin\varphi_t. \tag{18}$$

The value $\varphi_t$ can be interpreted as the mean value of the phase shift $\varphi(t)$ during the last period, and when exciting a linear system with a periodic input, the phase shift is in fact constant (after a transient phase). Therefore, (16) is a valid assumption in steady state. Finally, the phase shift $\varphi_t$ is obtained by

$$\varphi_t = -\arctan\left(\frac{\eta_2(t)}{\eta_1(t)}\right). \tag{19}$$

Note that in steady state, integration over several periods improves the robustness of the estimate of $\varphi_t$.

### 4.2 Phase Correction

The phase error $\varphi_t$ is corrected by a feedback technique borrowed from PLL design [19]. The reference signal $x_d(t)$ in (1) is shifted by a correction term $e(t)$,

$$x_d^s(t) = A_d \cos\big(\omega_d t + e(t)\big), \tag{20}$$

which is defined as

$$e(t) = -k \int_0^t \varphi_t \, dt. \tag{21}$$

Similarly, the derivatives of $x_d(t)$ are shifted in phase by $e(t)$. Replacing the reference signal $x_d(t)$ and its derivatives in the controller equations (9) and (10) by the shifted values,

$$x_d^s(t), \;\; \dot{x}_d^s(t), \;\; \ddot{x}_d^s(t), \;\; \text{and} \;\; \dddot{x}_d^s(t), \tag{22}$$

produces a new input $u(t)$, cf. (13), which compensates for the phase error. With the feedback integrator term $e(t)$, precise and robust phase locking is achieved. Convergence behavior is controlled by tuning the gain factor $k$.

## 5. Results

The developed synchronization scheme is demonstrated on small quadrocopters of about 30 cm diameter operated in the ETH Flying Machine Arena, a $10 \times 10 \times 10$ m indoor flight-test facility.

### 5.1 Experimental Setup

The setup is similar to [22]: An 8-camera Vicon motion capture system provides pose data for any vehicle in the space at 200Hz with a latency of about 25 ms. The localization data is sent to a PC, which runs the control algorithm, and which in turn sends commands to the quadrocopters, delivered with a latency between 30 to 60 ms. The flying vehicles are modified commercially available quadrocoptor described in [23]. Each vehicle accepts a collective thrust command and three angular rate commands at 50 Hz. An onboard 1 kHz feedback controller uses rate gyros to track the given commands. More details about this test environment may be found in [24] and [25].

For the experiments described below, two degrees of freedom (collective thrust and desired pitch rate) were controlled by the algorithm, while the other two degrees of freedom were handled by a linear controller described in [24]. The measurements $x(t)$ and $\theta(t)$ are provided directly by the Vicon system while velocity is obtained by approximately differentiating position data. The acceleration in the $x$-direction used in the tracking controller (10) is obtained from (2) and (3) by assuming the acceleration in the $z$-direction is negligible, and by using the measured pitch angle $\theta(t)$. In the referenced video, the music was pre-processed and the beat times were stored on the same PC that runs the control algorithm and sends the commands to the vehicle.

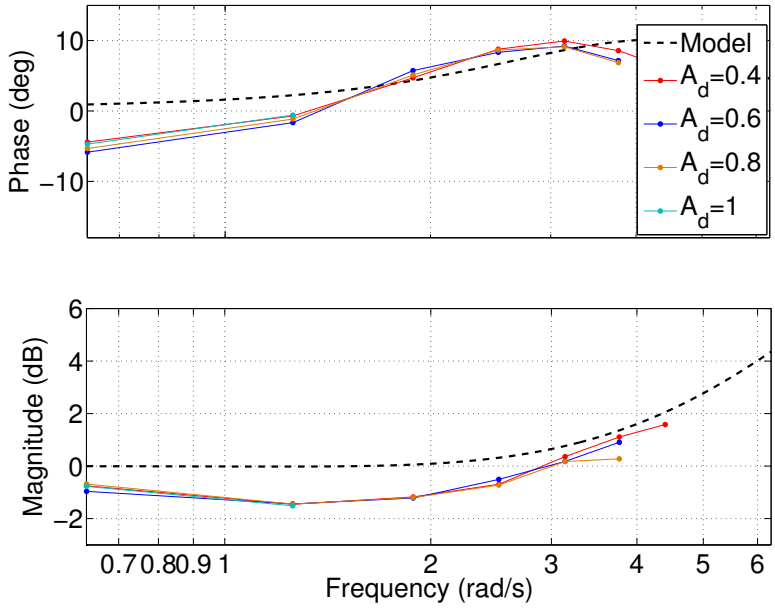



**Figure 5.** Experimentally determined Bode diagram for the closed-loop oscillating quadrocopter system *without* phase correction. The dashed line shows the theoretically derived transfer function taking communication delays and propeller dynamics into account.

## 5.2 System Characteristics

To begin with, the control laws (5) and (13) without synchronization are applied to the real system, and the behavior of the resulting oscillating quadrocopter motion is studied. In this case the desired amplitude $A_d$ and frequency $\omega_d$, cf. (1), can be interpreted as the inputs to the oscillating quadrocopter, Fig. 2, whereas the actual quadrocopter trajectory is the output. Since (2)-(4) represent a simplified model of the actual vehicle dynamics that neglects aerodynamic effects, delays in the system, the onboard controller, etc., the control laws (5) and (13) do not achieve synchronized trajectory tracking.

The transfer function of the closed-loop oscillating quadrocopter system (without the phase error compensation) was studied by varying the input frequency $\omega_d$ and amplitude $A_d$. The closed-loop response exhibits linear behavior with the frequency of the quadrocopter response equal to the input frequency. However, the motion is shifted in time, and depending on the frequency $\omega_d$ the amplitude is attenuated or amplified. Fig. 5 depicts the experimentally determined Bode plot for the frequency range of interest. Note that the Bode plot is independent of the amplitude $A_d$, as expected for a linear system.

The observed behavior can be explained by the non-idealities in the system. In particular, the delayed information exchange affects the overall system behavior. In addition, the dynamics of both the onboard controllers and the propeller motors are neglected in the system description (2)-(4). A quadrocopter model including realistic latency values (45 ms on sending commands and of 25 ms on receiving position data, see Section 5.1) and motor dynamics modeled as first-

order system,

$$\ddot{\theta}(t) = \frac{1}{T_\theta}\left(u(t) - \dot{\theta}(t)\right), \qquad T_\theta = 25\,\text{ms}, \tag{23}$$

produces the dashed-line behavior in Fig. 5. For deriving the transfer function, the simple relation (8) between the $x$-position and the pitch rate input was used. This approximate model accounts for the general trend in the experimental data.

Besides providing a better understanding of the closed-loop oscillating system, the experimental data can be incorporated as feedforward phase and amplitude compensation in the proposed synchronization scheme. The results are shown in the subsequent section, supporting the idea of an online identification of the values shown in Fig. 5 and constructing a look-up table. The look-up table might be continuously adapted to changing conditions in the environment (for example caused by worn out propellers). This allows a later implementation of highly agile maneuvers and fast changes between different motion primitives, cf. Fig. 8 and Fig. 9.

### 5.3 Synchronization Behavior

The proposed synchronization algorithm is applied to the quadrocopter and the resulting performance of the vehicle is analyzed. A reference signal with a frequency $\omega_d = 1.2\pi\,\text{rad/s}$ and an amplitude $A_d = 0.4\,\text{m}$ was chosen, cf. (1). Fig. 6 shows the quadrocopter response for three cases: (i) no phase correction, i.e. $k = 0$ in (21), (ii) phase error compensation with $k = 0.28$, and (iii) feedforward phase and amplitude correction based on the pre-determined values depicted in Fig. 3. After a short transient phase, the phase comparator (introduced in Section 4.1) outputs a constant phase error in the case of no phase correction, see Fig. 7. Perfect phase-locking is achieved when adding phase compensation through either a feedback or feedforward component. Note that the proposed phase comparator needs at least one full period to converge to the correct phase error. While the phase error between the reference trajectory and the actual quadrocopter response is hardly noticeable in Fig. 6, small phase errors are very visible and audible in actual experiments. In particular, humans expect zero vehicle velocity at beat times. Correspondingly, Fig. 7 plots the velocity of the quadrocopter at beat times, i.e. when the reference trajectory reaches its maximum or minimum value. The sign of every second velocity value is altered such that a constant positive phase delay is represented by positive velocity values, cf. Fig 7.

While assuming a constant reference frequency and amplitude in the theoretic derivations, frequency and amplitude changes are followed closely when compensating for phase and amplitude errors with a feedforward term, see Fig. 8 and Fig. 9. In fact, this better reflects the properties of real music, where beat intervals $T_i$, cf. Fig. 3, might vary over time. For changing frequencies, the reference is still given by a signal as depicted in Fig. 3, but corresponding to the next beat interval $T_i$ a different frequency $\omega_d$ is used for each half period of the sinusoidal signal (1).

A 'dancing' choreography was performed on the song *Beat goes on* from Dj Ross Vs Dy. A video of the corresponding quadrocopter motion is available at `www.tinyurl.com/dancingquadro` and also accompanying this paper. Throughout all experiments $\delta_z = \delta_x = 1$, $\omega_z = 1.25$, and $\omega_x = 2.5$, although the results are not sensitive to changes in these parameters.

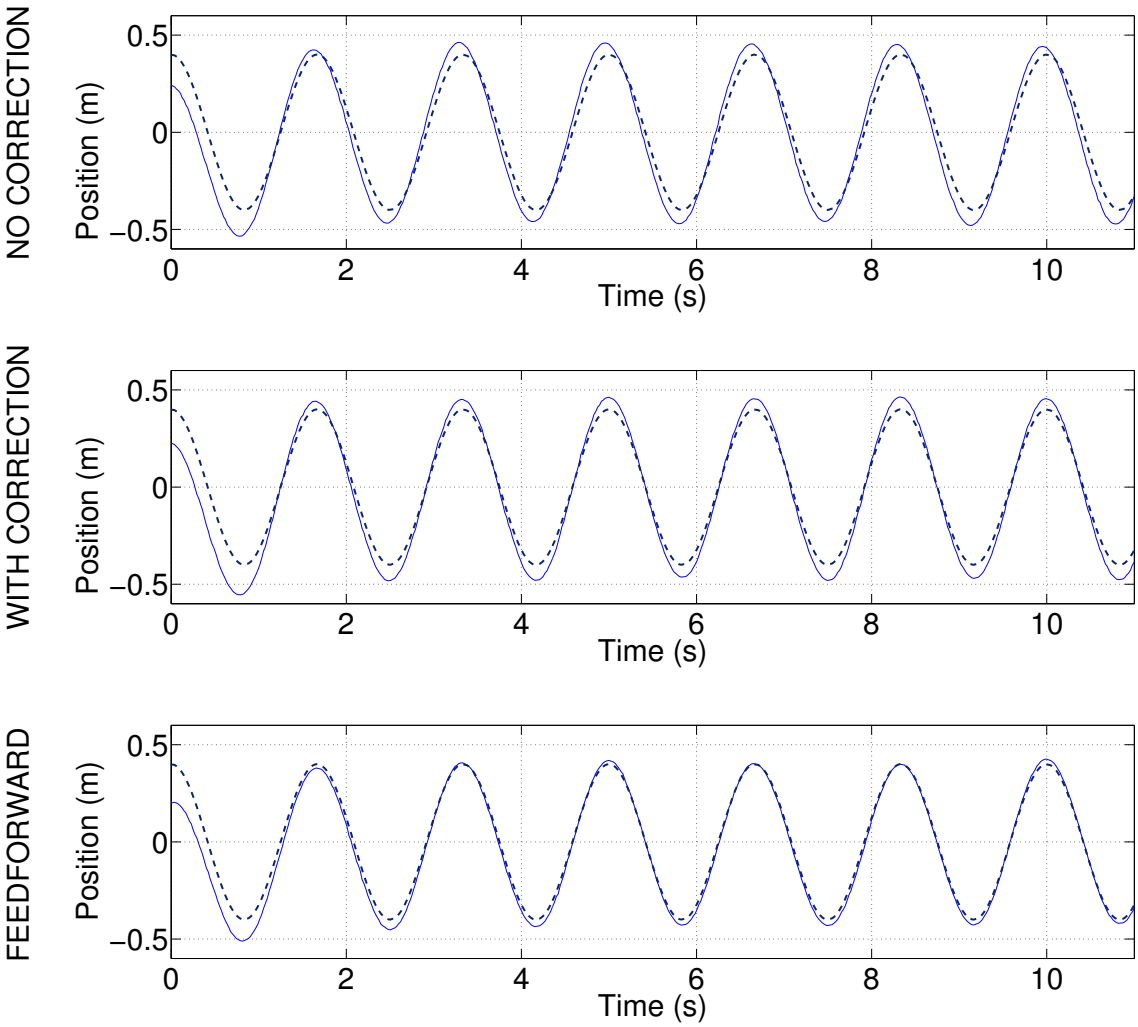



**Figure 6.** Quadrocopter response to the dashed-lined input signal for the case of no phase correction (top), phase error compensation (center), and feedforward phase and amplitude correction (bottom). The solid blue line represents the vehicle trajectory.

## 6. Conclusions

This paper presents a control and synchronization scheme that enables flying vehicles to perform rhythmic movements. The proposed algorithm synchronizes the side-to-side motion of a quadrocopter with the beat from an arbitrary piece of music. A feedback scheme is used to adjust the phase of the oscillating quadrocopter to the music reference signal that was deduced from the music song in a pre-processing step. Based on prior experimental data, The derived control and synchronization techniques generalize to any other periodic vehicle motion.

The presented results are a first step towards developing the algorithms capable of controlling multiple quadrocopters as they perform sophisticated aerobatic maneuvers timed to music. An aerobatic dance performance of a group of quadrocopters is envisioned in the Flying Machine Arena, where the music's features, like beat, volume, and melody, are reflected in the

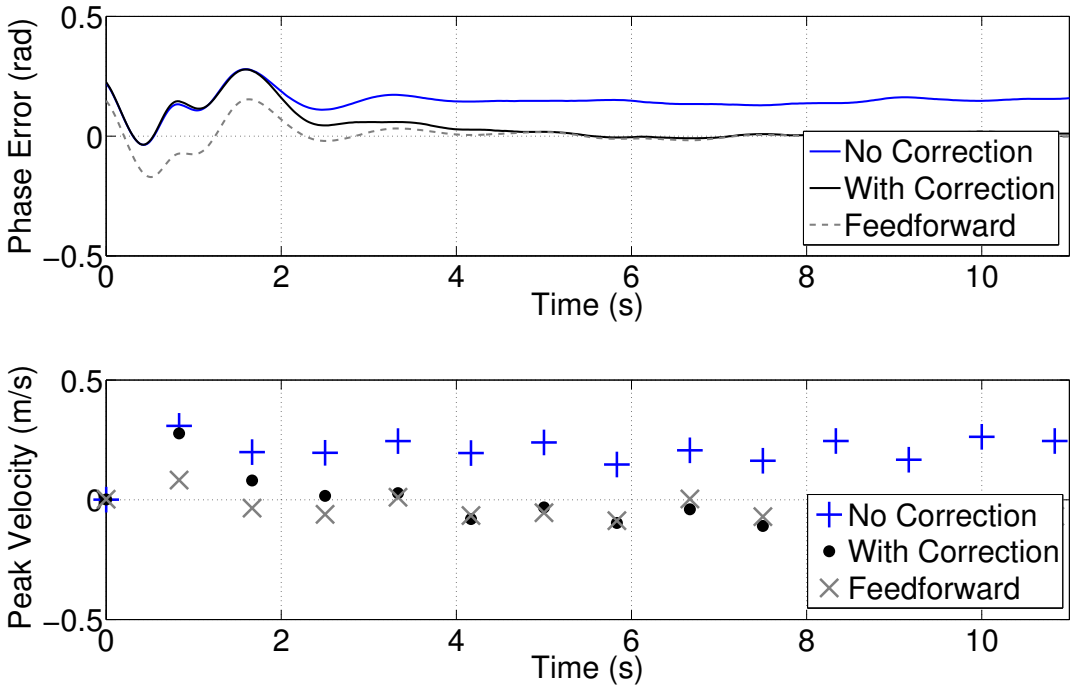



**Figure 7.** Top: Phase error of the vehicle trajectories shown in Fig. 6 (detected by the phase comparator). Bottom: Vehicle velocity at the peak values of the corresponding reference signal (using altered signs, see text for details).

movement of the quadrocopters.

## Acknowledgements

This work would never have been possible without the prior work of Guillaume Ducard and Felix Althaus. Their contributions are gratefully acknowledged.

## References

[1] Y. Koren, "Cross-coupled biaxial computer control for manufacturing systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 4, pp. 265–272, 1980.

[2] M. Tomizuka, J. S. Hu, T. C. Chiu, and T. Kamano, "Synchronization of two motion control axes under adaptive feedforward control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 114, no. 2, pp. 196–203, 1992.

[3] L. Ren, J. K. Mills, and D. Sun, "Trajectory tracking control for a 3-DOF planar parallel manipulator using the convex synchronized control method," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 613–623, 2008.

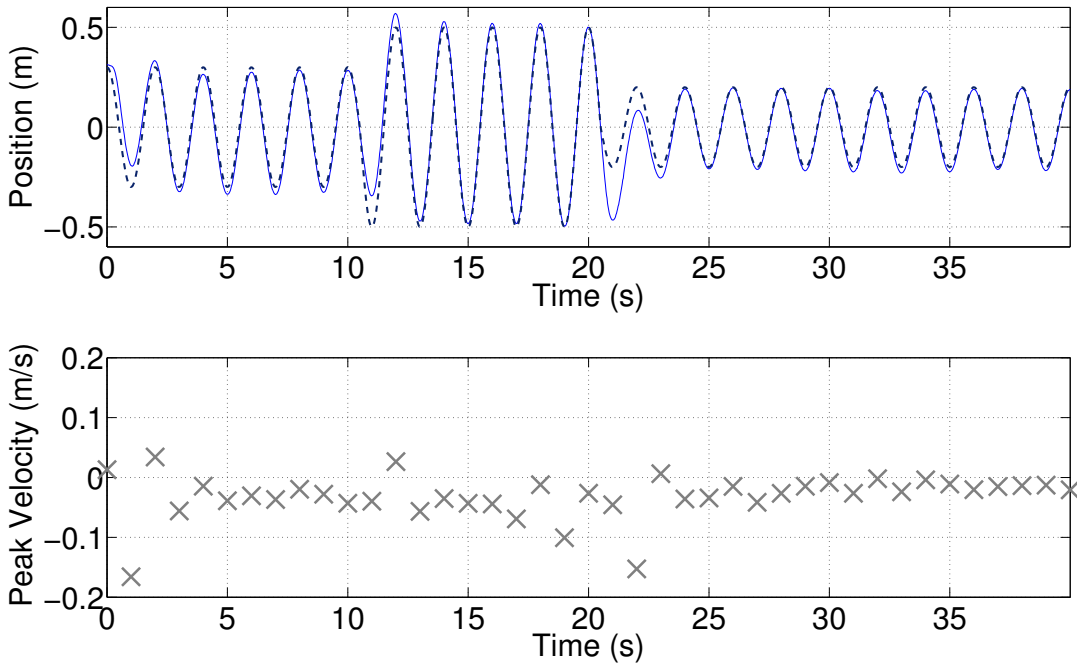



**Figure 8.** Top: Quadrocopter response (solid blue line) to an input sequence with changing amplitudes (dashed line) using feedforward phase and amplitude correction. Bottom: Vehicle velocity at peak values of the corresponding reference signal (using altered signs).

[4] D. Sun and J. K. Mills, "Adaptive synchronized control for coordination of multirobot assembly tasks," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 498–510, 2002.

[5] H.-T. Liu, J. Shan, and D. Sun, "Adaptive synchronization control of multiple spacecraft formation flying," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 3, pp. 337–342, 2007.

[6] L. Feng, Y. Koren, and J. Borenstein, "Cross-coupling motion controller for mobile robots," *IEEE Control Systems Magazine*, vol. 13, no. 6, pp. 35–43, 1993.

[7] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2459–2464.

[8] ——, "A robot singer with music recognition based on real-time beat tracking," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 199–204.

[9] K. Yoshii, K. Nakadai, T. Torii, Y. Hasegawa, H. Tsujino, K. Komatani, T. Ogata, and H. G. Okuno, "A biped robot that keeps steps in time with musical beats while listening to music with its own ears," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 1743–1750.

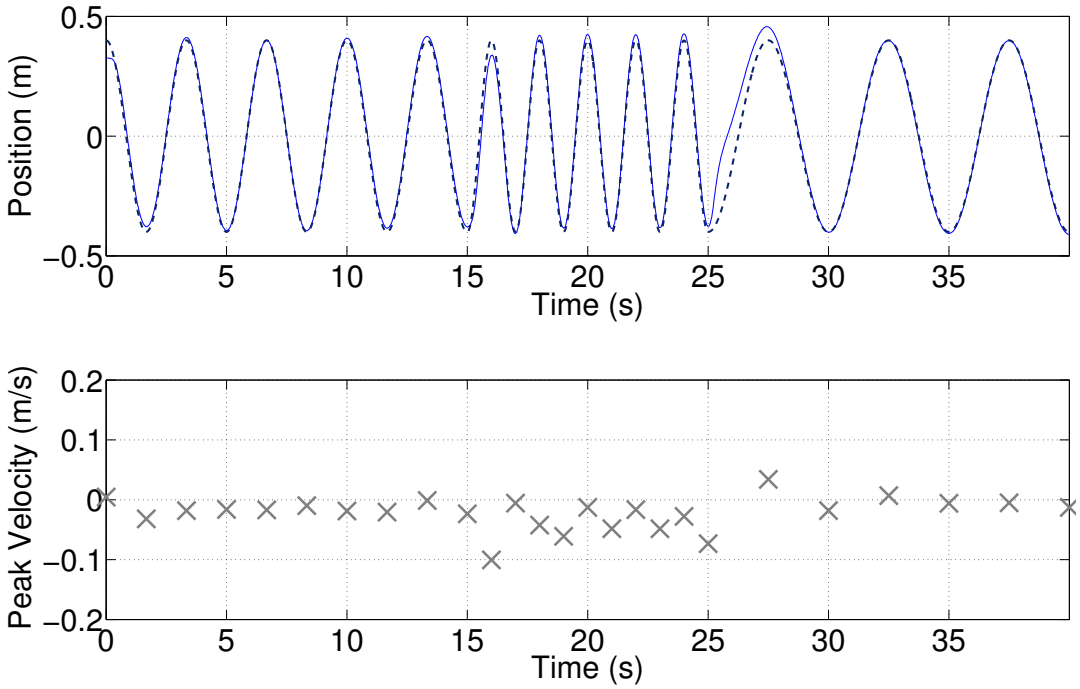



**Figure 9.** Top: Quadrocopter response (solid blue line) to an input sequence with changing frequencies (dashed line) using feedforward phase and amplitude correction. Bottom: Vehicle velocity at peak values of the corresponding reference signal (using altered signs).

[10] C. A. A. Calderon, M. R. Elara, C. Zhou, L. Hu, and B. Iniya, "Neural oscillator for rhythmic motion control of biped robot," in *Proceedings of the International Conference on Signal Processing, Communications and Networking (ICSCN)*, 2008, pp. 450–453.

[11] K. Shin'ya and S. Schaal, "Synchronized robot drumming by neural oscillator," *Journal of the Robotics Society of Japan*, vol. 19, pp. 116–123, 2001.

[12] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 2911–2916.

[13] C. Crick, M. Munz, and B. Scassellati, "Synchronization in social tasks: Robotic drumming," in *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, 2006, pp. 97–102.

[14] G. Kim, Y. Wang, and H. Seo, "Motion control of a dancing character with music," in *Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, 2007, pp. 930–936.

[15] T. Kim, S. I. Park, and S. Y. Shin, "Rhythmic-motion synthesis based on motion-beat analysis," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 392–401, 2003.

[16] J.-J. Aucouturier, "Cheek to chip: Dancing robots and AI's future," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 74–84, 2008.

[17] M. P. Michalowski, S. Sabanovic, and H. Kozima, "A dancing robot for rhythmic social interaction," in *Proceedings of the ACM/IEEE international Conference on Human-Robot Interaction (HRI)*, 2007, pp. 89–96.

[18] K. Kosuge, T. Hayashi, Y. Hirata, and R. Tobiyama, "Dance partner robot – Ms DanceR," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, 2003, pp. 3459–3464.

[19] W. F. Egan, *Phase-Lock Basics*, 2nd ed. Wiley-Interscience, 2008.

[20] G.-C. Hsieh and J. C. Hung, "Phase-locked loop techniques. A survey," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 6, pp. 609–615, 1996.

[21] K. Ogata, *Modern Control Engineering*, 4th ed. Prentice Hall, 2002.

[22] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[23] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 361–366.

[24] G. Ducard and R. D'Andrea, "Autonomous quadrotor flight using a vision system and accommodating frames misalignment," in *Proceedings of the IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2009, pp. 261–264.

[25] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

Paper VI

# A Platform for Dance Performances with Multiple Quadrocopters

Angela P. Schoellig · Federico Augugliaro · Raffaello D'Andrea

**Abstract**

This paper presents a platform for rhythmic flight with multiple quadrocopters. We envision an expressive multi-media dance performance that is automatically composed and controlled, given a random piece of music. Results in this paper prove the feasibility of audio-motion synchronization when precisely timing the side-to-side motion of a quadrocopter to the beat of the music. An illustration of the indoor flight space and the vehicles shows the characteristics and capabilities of the experimental setup. Prospective features of the platform are outlined and key challenges are emphasized. The paper concludes with a proof-of-concept demonstration showing three vehicles synchronizing their side-to-side motion to the music beat. Moreover, a dance performance to a remix of the sound track 'Pirates of the Caribbean' gives a first impression of the novel musical experience. Future steps include an appropriate multiscale music analysis and the development of algorithms for the automated generation of choreography based on a database of motion primitives.

Published in *Proc. of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) - Workshop on Robots and Musical Expressions,* 2010. Submission includes multimedia attachment, also found at `www.tinyurl.com/DancePlatform`.

## 1. Introduction

The interface of music and robotics has become a prominent area of research, attracting the attention of not only roboticists, but also musicians, artists, and the general public. When robot technology is brought together with musical expression and rhythmic performance, it opens up the space for a novel human-robot interplay and a unique musical experience that cannot be achieved by traditional means.

During the past decade research on musical robots has, for the most part, been driven by two distinct communities.

The first – consisting mostly of musicians, composers and music technologists – has generally sought to develop innovative forms of musical expression and sound production that overcome the limitations of human music generation and traditional musical instruments. Early robotic developments from this first community include the automation and mechanization of instruments, such as the piano, percussion and woodwinds (see [1, 2] and, for a historical overview, [3]); more recently, this group has been behind the development of perceptual music robots [4], some of which facilitate music collaboration with human musicians [5–7].

The second community, working out of the fields of robotics and engineering, has sought to use music to establish a new dimension of human-robot communication, interaction and collaboration. Research from this second community has been largely focused on the development of humanoid robots capable of imitating human musical behavior: robots that perform human-inspired rhythmic motions, such as dancing [8–13], drumming [14], stepping and singing [15–17] along to music. Common to both communities is the desire to create an audio-visual performance with both aesthetic and entertainment value; this merging of musical expression, robotic technology and entertainment has also been a fascinating playground for artists [18, 19]. The goal of this work is to create a novel visual musical experience: Multiple quadrocopters fly in a rhythmic performance, expressing the character of the music as they move in a coordinated and precisely-timed fashion through the three-dimensional space. Prior to flight performance, a random piece of music is analyzed and its main features are extracted. These features are later used to guide the quadrocopters' choreography. This approach differs from most other research on musical robots, which typically aims for a real-time interaction with the environment and which studies the problem of real-time music analysis, see e.g. [5, 17, 20–23]. In contrast, the focus of this work is the design, control, and synchronization of the rhythmic quadrocopter motion. Major challenges include:

**Motion Design** Given the body and dynamics of a quadrocopter, translating music into suitable motion patterns is a significant artistic challenge. Unlike humanoid dance robots, which may imitate any human dance movement and produce an easily-recognized expressive gesture [8–13], quadrocopters do not move the way humans do. Choreography of their movement is new, and requires creativity, aesthetic judgment and a deep understanding of the system's dynamics and its limits.

**Motion Control** A quadrocopter's nonlinear and unstable dynamics, cf. [24], demands sophisticated controls in order to achieve precise trajectories and exact audio-motion synchronization. Modeling errors and other nonidealities, such as motor saturation and communication delays, have a noticeable effect on the quadrocopter performance. Note that a stabilizing controller is required just to keep the vehicle in the air. Moreover, addi-

tional underlying controllers impose a periodic oscillation corresponding to the music frequency, presenting the basis for an overall rhythmic behavior.

**Motion Synchronization** To make the quadrocopters 'dance to music', it is essential to time the vehicle movement with the music beat precisely. In the proposed setup, however, the quadrocopters' dynamics are highly non-linear and not fully identified. Furthermore, the system's time delays are variable. An appropriate synchronization algorithm is therefore indispensable. Note that in most other approaches on musical robots, simple adjustments (like a constant time shift of the reference signal) are sufficient for synchronization, since they deal with systems that are better understood, less sensitive to disturbances, and, generally, easier to manage, see [9,10,15,22,25]. Inspiring work that explicitly considers synchronization is presented in [14,16,23,26].

**Motion Composition** Given an arbitrary piece of music, music analysis extracts the associated sequence of music features with the corresponding timing information. The goal is to automatically generate a dance performance with multiple quadrocopters, where the quadrocopters' motion not only reflects the character of the music but also takes into account the physical limitations of the space and the vehicle. We plan to develop an automated tool that will compose a suitable multi-vehicle choreography based on a selection of motion primitives (see **Motion Design**). Previous work on automatic music-driven dance synthesis focused exclusively on human dance motions; music features were analyzed and mapped manually to real human dance movements, cf. [27–29]. Algorithms were visualized and tested solely in virtual environments.

The core components mentioned above build upon a prior music analysis that extracts im-

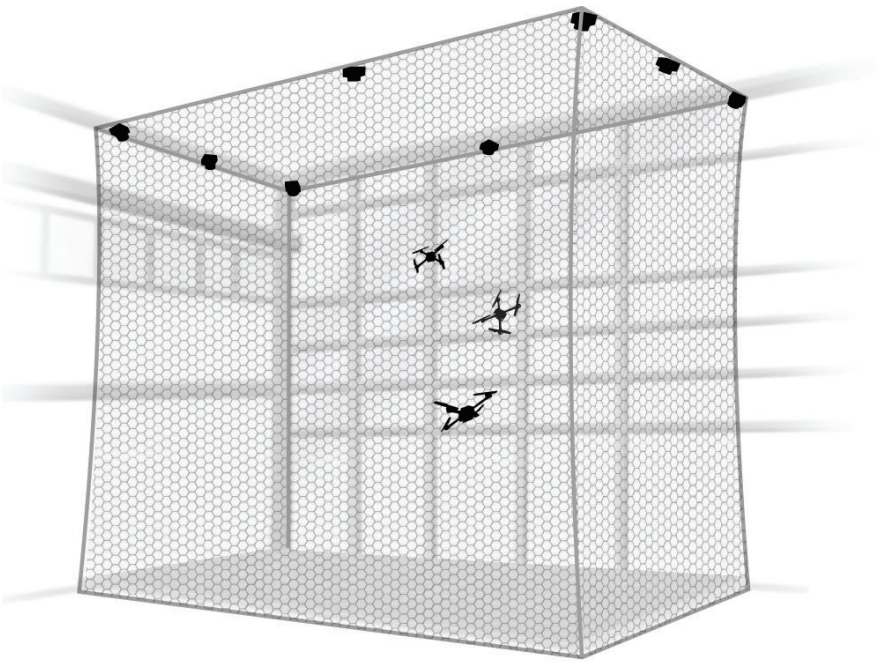


**Figure 1.** The Flying Machine Arena, an indoor flight space for multiple quadrocopters.

**Figure 2.** The quadrocopter, measuring 53 cm in diameter.

portant music features, such as beat, dynamic range, pitch, melody, etc., together with their timing information. The success of this project depends heavily on the quality of the multiscale music analysis. Various algorithms and software solutions (see [30, 31]) are available, especially for tempo and beat analysis, among others [32–36]. Moreover, every year researchers present the latest music analysis solutions at the Music Information Retrieval Evaluation eXchange (MIREX). In collaboration with those music experts, we would like to ask the following questions: Which music features are critical to the development of a musical quadrocopter performance? And which algorithms fit the proposed application?

This project aims to combine musical expertise, emotion, and aesthetic judgment with agile aerobatic maneuvers and sophisticated control techniques. In this interdisciplinary context, collaboration with a variety of experts is essential.

The objectives and prospective features of the proposed platform are presented in Section 4 together with a proof-of-concept demonstration in Section 5. The associated video, available also at `www.tinyurl.com/DancePlatform`, provides a first impression of the intended quadrocopter flight dance. The experimental setup and the characteristics of the vehicles are described in Section 2, and Section 3 explains the synchronization problem encountered when working within the proposed environment.

## 2. Experimental Setup

It was the agility of the quadrocopters, the dimension of the flight space, and the reliability of the communication and control infrastructure that laid the foundation for a musical dance performance with multiple vehicles. In the following, we sketch the experimental setup in order to convey an impression of the project's starting conditions.

## 2.1 The Flying Vehicle

Figure 2 shows the current flying vehicle. The quadrocopter is characterized by its small size, light weight, and structural and electronic robustness, making it a versatile and easy-to-operate experimental platform. The baseline platform is a X3D 'Hummingbird' quadrocopter designed and manufactured by Ascending Technologies GmbH [37]. Measuring 53 cm in diameter, the vehicle's overall weight including the onboard battery is approximately 460 g. The operational flying time varies between 10 and 20 minutes depending on the aggressiveness of the performed flight maneuvers. The vehicle is equipped with four brushless DC motors, which together are able to produce a vertical acceleration of around 12.5 m/s.

Though the original propulsion system, the motor controllers and the frame of the standard X3D quadrocopter were preserved, cf. [38], the central electronics and onboard sensors were replaced to obtain better control over the onboard algorithms and to have access to better-quality and higher-range rate gyro data. These changes allow for more aggressive maneuvers, faster turn rates, and generally better flight performance. With the new rate gyros, rotations of up to 2000 deg/s are possible around the body's principal axes of inertia. Detailed documentation of the changes are found in [39].

The quadrotor accepts three body angle rate commands and a collective thrust command at 50 Hz. An onboard 800 Hz feedback controller uses rate gyros to track the given commands. The exact onboard controller design is presented in [39]. Propeller wear trim factors allow for precise balancing of the quadrocopter. Each vehicle is equipped with two radio systems: a one-way 2.4 GHz module used exclusively for controlling the vehicle (to constrain the amount of variable latency in the system), and a bidirectional 2.4 GHz transceiver with a different modulation for non-time-critical communication such as data feedback or onboard parameter reads/writes.

Each quadrocopter is also equipped with three retro-reflective ball-shaped markers, which enable unambiguous vehicle identification by the Vicon communication system (described in Section 2.2 and 2.3) during flight.

Currently, six vehicles are ready to fly.

## 2.2 The Space

A $10 \times 10 \times 10$ m cube of indoor space, called the ETH Flying Machine Arena (FMA), is reserved for testing and validating autonomous quadrocopter flight, see Fig. 1. For a safe operation, the space is equipped with protective nets delimiting the space on three sides. A glass front on the fourth side allows visitors an undisturbed view of the flying vehicles. To reduce the occurrence of catastrophic crashes, 12 cm thick foam mattresses cover the ground. An 8-camera Vicon motion capture system [40] provides position and attitude data for all appropriately marked vehicles in the arena at 200 Hz with millimeter accuracy, and a latency of about 10 ms.

## 2.3 Control and Communication

The overall organization of the system is similar to [41]. The localization data provided by the Vicon system is sent to off-the-shelf PCs, which then execute estimation and control algorithms. These in turn deliver commands to the quadrocopters with an approximated latency of 20 ms. The overall system time delay, from sending a vehicle command to detecting the corresponding

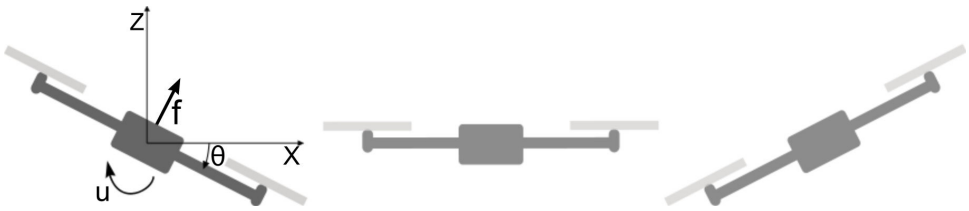



**Figure 3.** The desired side-to-side motion, a schematic of the two-dimensional quadrocopter model.

effects in the vehicle's Vicon pose data, varies between 10 ms and 40 ms with a mean value of 35 ms.

Data is sent via a multicast UDP scheme, allowing for convenient online visualization of all data sent over the network, and also for recording, playback, and export of arbitrary customized data series. A convenient side-effect of this setup is that estimation and control components are binary-identical when running in the real system or in simulation. The wireless and Vicon data bridges are simply replaced by a simulator process, with all of the other components remaining completely unaffected and unaware of any difference.

During normal operation the vehicle's translational degrees of freedom are controlled by linear PID controllers designed for near-hover operation. Yaw is held at a constant angle via a proportional controller. To achieve trajectory tracking, a sequence of reference points are fed to the controller together with appropriate feedforward commands. More details about this test environment may be found in [42] and [39].

## 3. The Synchronization Problem

Autonomous quadrocopters are fundamentally different in their dynamic behavior from other existing musical robots. To prove the feasibility of the project, we must therefore ask: Is it possible to precisely time a rhythmic quadrocopter movement with a music beat? In the following, a simple motion primitive is selected which highlights the quadrocopter's characteristic properties and for which a solution to the synchronization problem is sketched. The details of the synchronization algorithm are found in [43].

### 3.1 A Side-To-Side Motion

A planar side-to-side motion as depicted in Fig. 3 is considered, where at beat times the vehicle reaches the outer-most points of the trajectory. Given the music frequency $f_d$ (obtained from an a priori music analysis), a corresponding desired sinusoidal vehicle trajectory can be defined in the $xz-$plane,

$$x_d(t) = A_d \cos(\omega_d t), \qquad z_d(t) = z_d = \text{constant}, \tag{1}$$

with $\omega_d = 2\pi f_d$ assumed to be constant. Fig. 4 illustrates the beat-motion relation. The altitude of the quadrocopter is stabilized at a given height $z_d$.

### 3.2 The Quadrocopter Dynamics

The side-to-side motion (1) is defined in the $xz-$plane. In-plane and out-of-plane dynamics are thus decoupled. Additional degrees of freedom are separately stabilized and do not affect the rhythmic motion. The equations governing the dynamics of the system are then given by

$$\ddot{z}(t) = f(t)\cos\theta(t) - g \tag{2}$$
$$\ddot{x}(t) = f(t)\sin\theta(t) \tag{3}$$
$$\dot{\theta}(t) = u(t), \tag{4}$$

where $g$ is the gravitational constant and $\theta(t)$ is the pitch angle, see Fig. 3. The inputs to the system are the normalized thrust $f(t)$ in m/s$^2$ and the pitch rate $u(t)$ in rad/s.

### 3.3 Control and Synchronization

Underlying controllers turn the unstable quadrocopter dynamics into an oscillating system. A phase comparator detects the phase error between the desired reference trajectory and the actual quadrocopter motion. After having determined the phase error, it is compensated for by closing the loop, similar to [16] and [26].

The oscillating quadrocopter motion is achieved by using a cascading controller: the $z$-direction is stabilized first and, assuming a constant height, the trajectory-tracking controller for the $x$-direction is then designed. With

$$f(t) = \frac{1}{\cos\theta(t)}\Big(g - 2\delta_z\omega_z\dot{z}(t) - \omega_z^2\left(z(t) - z_d\right)\Big), \tag{5}$$

and an appropriate choice of the parameter $\delta_z$ and $\omega_z$, the dynamics in $z$-direction are stabilized, cf. [43]. Building upon the above control scheme (5) and assuming a resulting constant height $z_d$, the thrust $f(t)$ is

$$f(t) = \frac{g}{\cos\theta(t)}. \tag{6}$$

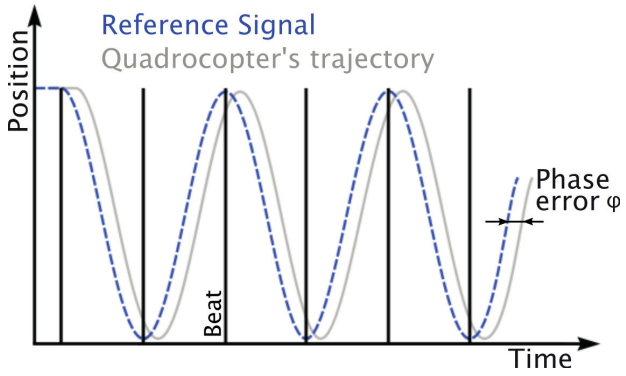



**Figure 4.** Reference trajectory in lateral direction (dashed line) with corresponding quadrocopter motion (solid line). Vertical lines represent beat times.

With (6), to first order, the $x$-dynamics (3) reduce to

$$\ddot{x}(t) = g\,\theta(t) \quad \Leftrightarrow \quad \dddot{x}(t) = g\,u(t), \tag{7}$$

assuming small $\theta(t)$. With the aim of following the desired sinusoidal side-to-side motion (1), the input

$$u(t) = \frac{1}{g}\left(\overline{u}_1(t) + \overline{u}_2(t)\right) \tag{8}$$

is composed of a feedforward component,

$$\overline{u}_1(t) = \dddot{x}_d(t) = A_d\omega_d^3 \sin(\omega_d t), \tag{9}$$

and an additional feedback term to correct for errors,

$$\overline{u}_2(t) = \alpha\left(\ddot{x}_d(t) - \ddot{x}(t)\right) + \beta\left(\dot{x}_d(t) - \dot{x}(t)\right) + \gamma\left(x_d(t) - x(t)\right), \tag{10}$$

with the tuning parameters $\alpha$, $\beta$, and $\gamma$. When applying the input $u(t)$ as defined in (8), a phase shift is observed between the reference trajectory of the sideways motion and the actual quadrocopter trajectory, illustrated in Fig. 4. (Corresponding experimental results are shown in Fig. 5.) This phenomenon results mainly from unmodeled dynamics which were neglected in the controller design, such as communication delays and propeller dynamics. This problem is resolved in two stages. First, the phase shift $\varphi(t)$ between the quadrocopter trajectory $x(t)$ and the desired motion (1) is determined by an integration over a full period $T_d = 1/f_d$,

$$\eta_1(t) = \frac{1}{T_d}\int_{t-T_d}^{t} x(t)\cos(\omega_d t)\; dt = \frac{A}{2}\cos\varphi_t \tag{11}$$

$$\eta_2(t) = \frac{1}{T_d}\int_{t-T_d}^{t} x(t)\sin(\omega_d t)\; dt = -\frac{A}{2}\sin\varphi_t, \tag{12}$$

where

$$\varphi_t = -\arctan\left(\frac{\eta_2(t)}{\eta_1(t)}\right). \tag{13}$$

In stage two, the phase error $\varphi_t$ is corrected by a feedback technique borrowed from PLL design [44] shifting the reference signal $x_d(t)$ in (1) by a correction term $e(t)$,

$$x_d^s(t) = A_d\cos\big(\omega_d t + e(t)\big), \qquad \text{with} \quad e(t) = -k\int_0^t \varphi_t\; dt. \tag{14}$$

Similarly, the derivatives of $x_d(t)$ in (8) are shifted in phase by $e(t)$. With the feedback integrator term $e(t)$, precise and robust phase locking is achieved. Convergence behavior is controlled by tuning the gain factor $k$.

Experimental results support the theoretical idea, see Fig. 5 and 6. Without phase correction a constant, non-zero phase error remains, while with the proposed phase correction method, the phase error converges to zero and perfect synchronization is achieved. Note that even when the phase error between the reference trajectory and the actual quadrocopter response is hardly noticeable in Fig. 5, actual experiments show that small phase errors remain visible and audible to humans.

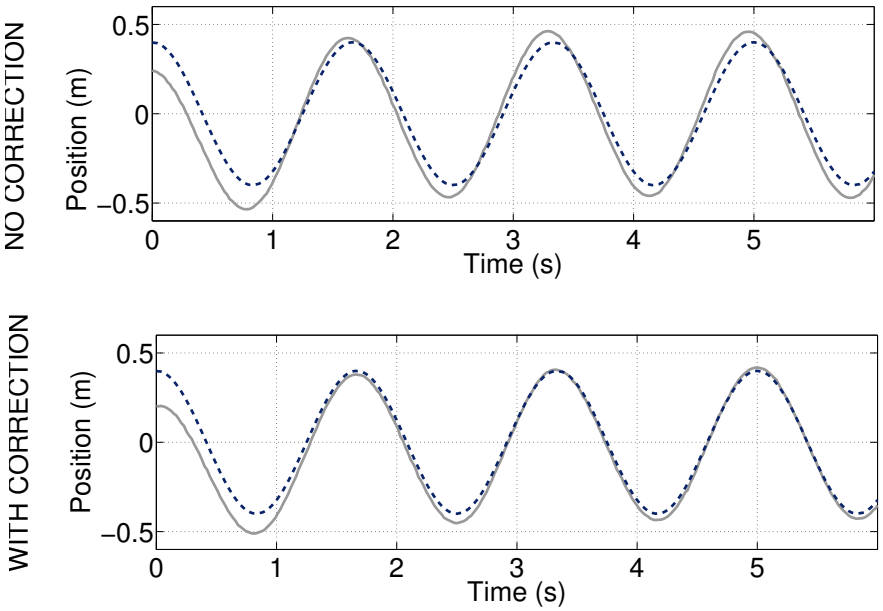



**Figure 5.** Quadrocopter response to the dashed-lined input signal for the case of no phase correction (top) and phase error compensation (bottom). The solid line represents the vehicle trajectory.

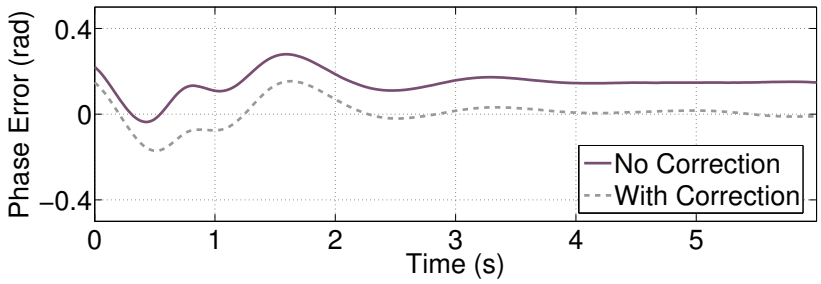



**Figure 6.** Phase error of the vehicle trajectories shown in Fig. 5 (detected by the phase comparator).

## 4. A Dance Performance

In the previous section, the fundamental basis of the work was presented: a quadrocopter motion that is synchronized to music. This is the first successful step towards choreographing a flying music show. Below, the overall concept and vision is outlined together with the corresponding steps necessary to realize this idea.

### 4.1 Vision

We envision the Flying Machine Arena as a stage for a dance performance featuring multiple quadrocopters, which move in rhythm to music and perform flips, eights, circles and other aggressive maneuvers. Our goal is to be able to quickly process any arbitrary piece of music and translate it into choreography that reflects the music's character.

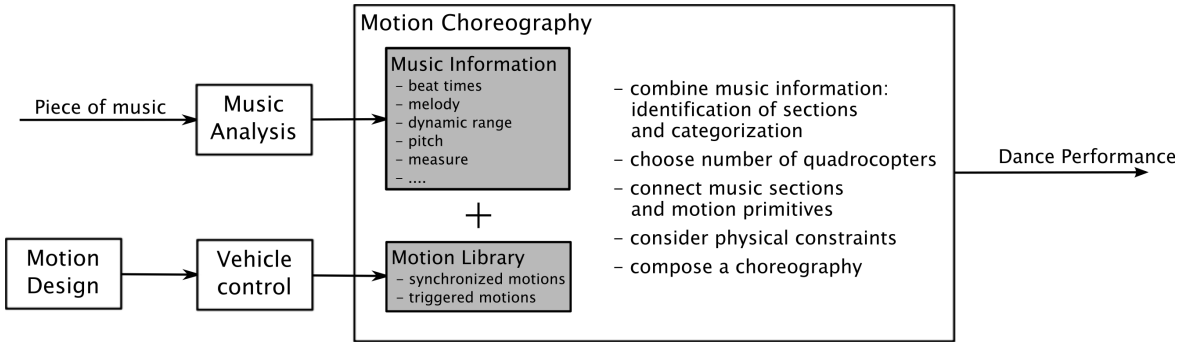



**Figure 7.** The proposed platform: combining music and quadrocopter motion into a multi-vehicle dance performance.

To this end, the development of the musical platform can be divided into four main components: music analysis, motion design, vehicle control, and, finally, motion choreography. The different steps are illustrated in Fig. 7 and discussed next.

## 4.2 Music Analysis

Humans often tap their feet to the beat of music as a reflection of their rhythmic perception. Music comprehension is not based solely on rhythm, however. Humans are able to perceive a sophisticated ensemble of musical features - including melody, measure, pitch, dynamic range and recurring themes - which help them to associate emotions with what they are listening to, and to react with a corresponding range of emotive movements we refer to as 'dance'.

Thus far, robotic perception of music has been largely limited to beat tracking. Early beat tracking algorithms were developed as early as 1994 [45], and automated beat tracking is still an active area of research. Recent work includes [30,31,34–36]. Given the current state of the art, can we expect robots to perceive and respond to music in a manner sophisticated enough to call 'dance'? Which music features are responsible for human musical perception? Is it possible to extract these features from the music and correlate them with emotional expression?

A multiscale music analysis is fundamental to the creation of a meaningful choreography. The music is digitized and divided into different sections that are categorized according to rhythm, melody and character. The result is a vocabulary that describes the temporal development of the music, which is in turn coupled with a variety of quadrocopter movements.

The proposed application aims to automate and accelerate this process. For now, however, a semi-assisted system is a reasonable expectation: Software processes a piece of music and suggests a categorization to a human user, who oversees and corrects the results. This information is then forwarded to the choreography component, which will create a suitable composition of motions for the quadrocopters, see Section 4.5.

## 4.3 Motion Design

As already mentioned in Section 1, defining suitable rhythmic motion primitives for quadrocopters is a major challenge. When developing a dancing performance, two different types of motion are distinguished:

**Synchronized Motion** Movements that aim to follow the rhythm of the music must be pre-

cisely synchronized to the beat (or multiple of it). Normally, these are swing motions, consisting of sinusoids – which enable the ready measurement and correction of phase error – to achieve perfect synchronization. The side-to-side motion described in Section 3 belongs to this type of motion primitives.

**Triggered Motion** Movements that are not strictly linked to the rhythm of the music, but start at a beat time, are executed during a given time interval, and end again with a music beat. These motion primitives are used to transition between two different synchronized motions, or to reflect a particular section of the music. Aggressive trajectories, like flips, eights, and circles, belong to this group of motion primitives.

Numerous motions can be created and cataloged according to the above categories. In this way, a library of motion primitives is built, serving as an important resource for the creation of new choreographies.

### 4.4 Vehicle Control

As emphasized in Section 1, sophisticated methods are required in order to control a quadrocopter's dynamics.

The synchronized and triggered motions introduced above have to be carried over into the real world, and different control challenges are associated with each. Triggered motions are complex and highly dynamic maneuvers requiring fast controllers and appropriate feedforward terms, which may need to be acquired through (machine) learning. In contrast, synchronized motions are less aggressive and easier to follow, but require special attention to maintain precise timing.

According to [46], two notes are perceived as being simultaneous if they occur within less than 30 ms. Accuracy with respect to beat times must therefore be in this range, too. To define and measure the timing error, we introduced the concept of phase in Section 3. Indeed, the music beat can be translated into a sinusoidal signal which builds the reference trajectory for the synchronized motion. In this case, a timing error is reflected by a non-zero phase error between the music reference and the actual vehicle trajectory. If trajectories are representable by a sinusoidal, phase-dependent description (as e.g. eights or circles), the timing error can always be obtained from the phase error. After having recognized the phase error, an approach like the one described in Section 3 can be used to compensate for it, achieving a precisely synchronized motion.

### 4.5 Motion Choreography

The previous steps result in a categorized piece of music (from the music analysis, see Section 4.2) and a library of motion primitives (from the motion design, see Section 4.3). The physical limits of the vehicles and the space are already known. Is it possible to combine all this information in a meaningful way? How can multiple quadrocopters fly in a manner we can recognize as 'dance'?

Well-established and recognizable parameters for human dance have long been in use by professional dancers, choreographers and dance teachers to build choreography with interest, dynamics and aesthetic appeal. These parameters can provide us with a framework for meaningful quadrocoptor choreography, and are described as follows:

**Space** Space refers to the area the dancer is performing in and also relates to how the dancer moves through the area as characterized by the direction and path of a movement, its size, level, and shape.

**Time** Time encompasses rhythm, tempo, duration, and phrasing of movements. Using time in different combinations to music can create intricate visual effects. Ideas such as quick-quick, slow or stop movements are examples.

**Energy** Energy relates to the quality of movement. This concept is recognizable when comparing ballet and tap dance. Some types of choreography are soft and smooth, while others are sharp and energetic.

**Structure** Structure represents the organization of movement sequences into larger concepts: the combination and variation of movements using recurring elements, contrast, and repetition. Movements can even follow a specific story line to convey specific information through a dance.

With these parameters in mind, the motion primitives described in Section 4.3 can eventually be combined into sequences, which can in turn be combined to create an overall choreographic performance. Endless permutations are possible, much the way individual words can be combined into a variety of subtle and sophisticated stories. Finally, an expressive connection of music sequences and motion primitives is established capable of visually conveying the music's emotions to the audience. As choreographing performances for human dancers, creativity and aesthetic judgment is required to achieve artistic quality.

## 5. Current Status

So far, our research has been primarily focused on control and synchronization. As explained in Section 3, a method was developed that is able to precisely synchronize a sinusoidal quadrocopter motion to a given music reference signal. Currently, a horizontal swing motion in the $xz$-plane and a similar motion in the $yz$-plane are implemented using the proposed algorithm. This allows a dancing behavior both along the $x$- and $y$-axis. Using multiple vehicles, this already leads to a wide range of combinations and patterns.

In addition to synchronized motions, aggressive trajectories are flown in the FMA with high accuracy and an impressive smoothness. Our research group is pushing the limits of what can be achieved with quadrocopters. For example, the quadrocopters learned to perform up to three flips, cf. [39].

A variety of motion primitives are now available in our library, discussed in Section 4.3. More complex motions of both types will be investigated in order to add variation to our quadrocopter dance performances. As mentioned in Section 1, the motion design is not trivial and requires further research. Specifically, a transition motion capable of taking the quadrocopters from the final position of one rhythmic motion primitive to the starting position of the following motion must be implemented. The transition motion should be fast and guarantee a collision-free position change.

We developed a software platform built on a collection of synchronized motion primitives to guide the choreography of multiple quadrocopters flying together. A graphical user interface

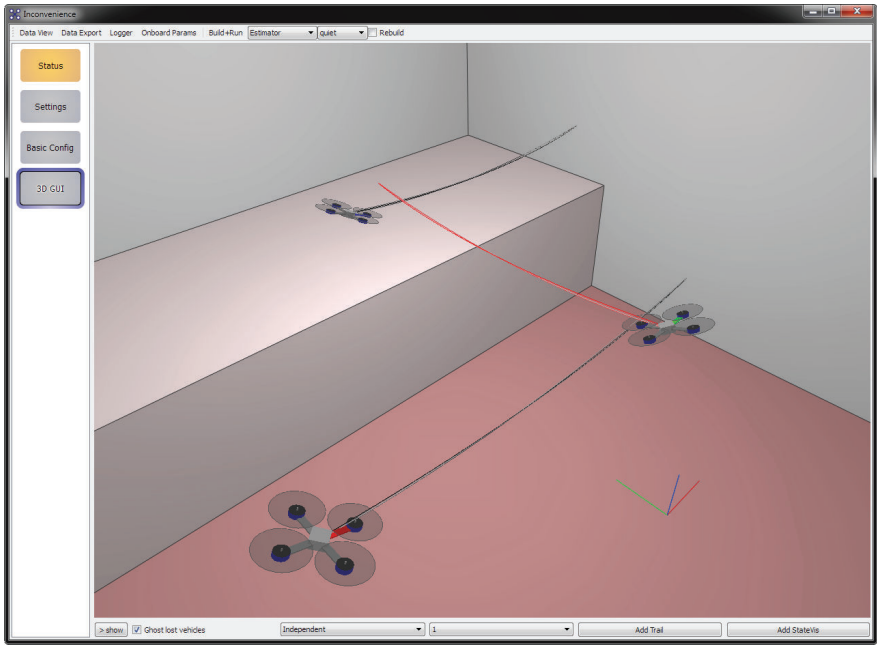



**Figure 8.** A three-vehicle swing motion synchronized to the music beat; screen shot of our simulation environment.

(GUI) allows the user to assign desired motions to specific quadrocopters, set all the necessary parameters, take off and fly the vehicles. This platform is especially important for testing new motions. The results shown in Section 5.1 were developed using this software platform. The dance motion presented in Section 5.2 is also incorporated into the graphical interface. The platform is connected to a simulation environment that is part of the FMA's infrastructure, see Section 2. The simulation environment provides a 3D view of the FMA and allows to visually determine the quality of a new motion or controller. As new motion primitives are created, they can be immediately added to the software platform.

In order to create a dancing performance, as in Section 5.2, the beat times have to be extracted from the music. This process was done with the aid of Beatroot [31]. Beatroot analyzes the music and provides beat times. An interface visualizes the beats, such that the user can listen to the music and adjust the beat times if mismatches are noticed.

For designing the dance choreography, the extracted beat times give the underlying frequency. The performance combines the synchronized motions and the triggered motions. After having analyzed a piece of music, the setup of a new dancing performance is simple and can be done in short time; however, this procedure is not yet automated. The assignment of different motions to different sections of the music is done manually in order to reflect the character of the musical piece. In an interdisciplinary exchange, we hope to learn more about methods that are able to perform the multiscale analysis, as discussed in Section 4.2, and identify the different parts of a musical piece.

Most of our work has focused on the control of the quadrocopters and their motions. We are still looking for an automated method to retrieve information from music. Given this information, the key challenge will be to develop an algorithm for automatic choreography and motion

planning. The results presented in this paper show the first steps towards a musical platform as presented in Section 4.

The two examples presented below show the capabilities of the current platform.

### 5.1 Example: Three-Vehicle Swing Motion

One possibility offered by the platform is to manually assign synchronized motions to specific quadrocopters. This can be done for testing purposes or simply to produce variation in the performance. The example demonstrates how the platform integrates multiple quadrocopters executing different motions: two vehicles perform a swing motion in the $x$-direction, while another one crosses their trajectories with a swing motion in the $y$-direction, see Fig. 8. Using the methods developed in [43], synchronization is achieved between the motion of the three quadrocopters and the music beat. The method is robust to the differences between the vehicles (e.g. due to the marker's position on the vehicle). The experimental results are shown in the attached video `www.tinyurl.com/DancePlatform`.

### 5.2 Example: Dancing Performance with Two Vehicles

We also developed a dancing performance for two vehicles based on a remix of the 'Pirates of the Caribbean' sound track. First the music is analyzed and the beat times are extracted. Then the different parts of the musical piece are identified and manually assigned to suitable motions from the library. In this example, a circle trajectory is chosen while the quadrocopters spin. Combinations with different speed and circle radii are shown in the associated video `www.tinyurl.com/DancePlatform`. The dancing parts consist of the swing motion presented in Section 3. The amplitude is varied to achieve different effects.

## 6. Conclusions

In this paper, a novel musical experience is proposed: flying vehicles that express the character of the music by performing an aerial dance. A cubic indoor flight space forms the stage, and small autonomous quadrocopters are the actors of this performance. Current features of the platform include the control of the quadrocopters and an audio-motion synchronization. A software infrastructure is built to be easily extendible with regard to the future objectives. One prospective feature is an automated multiscale music analysis that is able to identify distinct musical characteristics including beat, measure, dynamic range, melody and recurring themes. Building upon this information, the project aims towards an automated generation of coordinated musical multi-vehicle movements.

This project lies at the interface between robotics, control and music information retrieval, and has thus far been focused on the control aspect. The project's interdisciplinary nature requires intensive exchange and close cooperation with researcher in other fields. To this end, we are able to provide the experimental setup and control knowledge; we seek input concerning musical information retrieval and interpretation of music features.

## Acknowledgements

This work would have not been possible without the prior work of Markus Hehn and Sergei Lupashin. Their contributions are gratefully acknowledged.

## References

[1] E. Singer, J. Feddersen, C. Redmon, and B. Bowen, "LEMUR's musical robots," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, 2004, pp. 181–184.

[2] R. B. Dannenberg, B. Brown, G. Zeglin, and R. Lupish, "McBlare: a robotic bagpipe player," in *Proceedings of the International Conference on New Interface for Musical Expression (NIME)*, 2005, pp. 80–84.

[3] A. Kapur, "A history of robotic musical instruments," in *Proceedings of the International Computer Music Conference*, 2005, pp. 21–28.

[4] N. A. Baginsky, "The three sirens: a self-learning robotic rock band," Available online at www.the-three-sirens.info (accessed June 23, 2010), 2004.

[5] G. Weinberg, S. Driscoll, and M. Parry, "Musical interactions with a perceptual robotic percussionist," in *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, 2005, pp. 456–461.

[6] G. Weinberg and S. Driscoll, "The design of a perceptual and improvisational robotic marimba player," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, 2007, pp. 769–774.

[7] G. Hoffman and G. Weinberg, "Gesture-based human-robot jazz improvisation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 582–587.

[8] J.-J. Aucouturier, "Cheek to chip: dancing robots and AI's future," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 74–84, 2008.

[9] K. Shinozaki, A. Iwatani, and R. Nakatsu, "Construction and evaluation of a robot dance system," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, 2008, pp. 366–370.

[10] D. Grunberg, R. Ellenberg, Y. Kim, and P. Oh, "Creating an autonomous dancing robot," in *Proceedings of the International Conference on Hybrid Information Technology (ICHIT)*, 2009, pp. 221–227.

[11] A. Nakazawa, S. Nakaoka, K. Ikeuchi, and K. Yokoi, "Imitating human dance motions through motion structure analysis," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002, pp. 2539–2544.

[12] J. Oliveira, F. Gouyon, and L. P. Reis, "Towards an interactive framework for robot dancing applications," in *Proceedings of the International Conference on Digital Arts (ARTECH)*, 2008.

[13] K. Kosuge, T. Hayashi, Y. Hirata, and R. Tobiyama, "Dance partner robot – Ms DanceR," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, 2003, pp. 3459–3464.

[14] S. Kotosaka and S. Schaal, "Synchronized robot drumming by neural oscillator," *Journal of the Robotics Society of Japan*, vol. 19, pp. 116–123, 2001.

[15] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2459–2464.

[16] K. Yoshii, K. Nakadai, T. Torii, Y. Hasegawa, H. Tsujino, K. Komatani, T. Ogata, and H. G. Okuno, "A biped robot that keeps steps in time with musical beats while listening to music with its own ears," in *Proceediings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 1743–1750.

[17] R. Takeda, K. Yoshii, K. Komatani, T. Ogata, and H. G. Okuno, "A robot listens to music and counts its beats aloud by separating music from counting voice," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 1538–1543.

[18] F. Thorn, "Felix's machines," Available online at www.felixsmachines.com (accessed June 22, 2010), 2008.

[19] C. Southworth and L. Hasan, "Ensemble robot," Online at www.ensemblerobot.com (accessed June 22, 2010), 2004-2006.

[20] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot singer with music recognition based on real-time beat tracking," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008, pp. 199–204.

[21] K. Murata, K. Nakadai, R. Takeda, H. G. Okuno, T. Torii, Y. Hasegawa, and H. Tsujino, "A beat-tracking robot for human-robot interaction and its evaluation," in *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*, 2008, pp. 79–84.

[22] M. P. Michalowski, S. Sabanovic, and H. Kozima, "A dancing robot for rhythmic social interaction," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2007, pp. 89–96.

[23] C. Crick, M. Munz, and B. Scassellati, "Synchronization in social tasks: robotic drumming," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, 2006, pp. 97–102.

[24] G. M. Hoffmann, H. Huang, S. L. Wasl, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2007.

[25] T. Takeda, K. Kosuge, and Y. Hirata, "HMM-based dance step estimation for dance partner robot – Ms DanceR," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 3245–3250.

[26] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 2911–2916.

[27] K.-M. Chen, S.-T. Shen, and S. D. Prior, "Using music and motion analysis to construct 3D animations and visualisations," *Digital Creativity*, vol. 19, no. 2, pp. 91–104, 2008.

[28] K. Takahashi and H. Ueda, "A dance synthesis system using motion capture data," *Knowledge Acquisition: Approaches, Algorithms and Applications*, pp. 208–217, 2009.

[29] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp, "Multi-modal analysis of dance performances for music-driven choreography synthesis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[30] G. Tzanetakis, "Marsyas: music analysis, retrieval and synthesis for audio signals," Available online at www.marsyasweb.appspot.com/ (accessed June 26, 2010), 2009.

[31] S. Dixon, "Beatroot: an interactive beat tracking and visualisation system," Available online at www.elec.qmul.ac.uk/people/simond/beatroot/ (accessed June 26, 2010), 2006.

[32] M. S. Puckette, T. Apel, and D. D. Zicarelli, "Real-time audio analysis tools for Pd and MSP," in *Proceedings of the International Computer Music Conference*, 1998, pp. 109–112.

[33] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[34] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri, "Evaluation of audio beat tracking and music tempo extraction algorithms," *Journal of New Music Research*, vol. 36, no. 1, pp. 1–16, 2007.

[35] D. P. W. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[36] Y. Shiu and C.-C. J. Kuo, "Musical beat tracking via Kalman filtering and noisy measurements selection," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 3250–3253.

[37] Ascending Technologies GmbH, "Multi-rotor air vehicles," Available online at www.asctec.de (accessed June 27, 2010), 2010.

[38] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 361–366.

[39] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[40] Vicon, "Motion capture systems," Available online at www.vicon.com (accessed June 27, 2010), 2010.

[41] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[42] G. Ducard and R. D'Andrea, "Autonomous quadrotor flight using a vision system and accommodating frames misalignment," in *Proceedings of the IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2009, pp. 261–264.

[43] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3355–3360.

[44] W. F. Egan, *Phase-Lock Basics*, 2nd ed. Wiley-Interscience, 2008.

[45] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," in *Proceedings of the ACM International Conference on Multimedia*, 1994, pp. 365–372.

[46] R. A. Rasch, "The perception of simultaneous notes such as in polyphonic music," *Acustica*, vol. 40, no. 1, pp. 21–33, 1978.

Paper VII

# Feed-Forward Parameter Identification for Precise Periodic Quadrocopter Motions

Angela P. Schoellig · Clemens Wiltsche · Raffaello D'Andrea

**Abstract**

This paper presents an approach for precisely tracking periodic trajectories with a quadrocopter. In order to improve temporal and spatial tracking performance, we propose a feed-forward strategy that adapts the motion parameters sent to the vehicle controller. The motion parameters are either adjusted on the fly or, in order to avoid initial transients, identified prior to the flight performance. We outline an identification scheme that tunes parameters for a large class of periodic motions, and requires only a small number of identification experiments prior to flight. This reduced identification is based on analysis and experiments showing that the quadrocopter's closed-loop dynamics can be approximated by three directionally decoupled linear systems. We show the effectiveness of this approach by performing a sequence of periodic motions on real quadrocopters using the tuned parameters obtained by the reduced identification.

Published in *Proc. of the 2012 American Control Conference (ACC),* 2012.

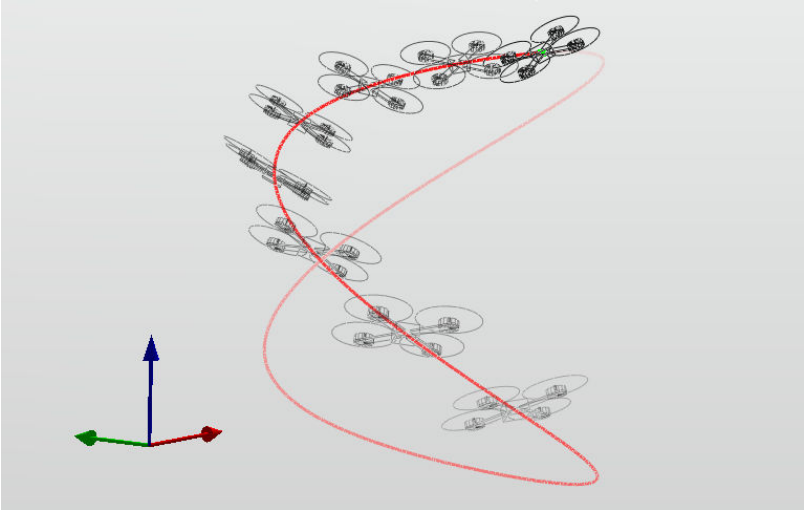


**Figure 1.** An example of a 3D periodic motion (with $\omega_d^{x,y} = 2\omega_d^z = \pi\,\mathrm{rad/s}$, $A_d^{x,y} = A_d^z/2 = 0.4\,\mathrm{m}$, $\theta_d^x = \pi/2$).

## 1. Introduction

The objective of the research presented in this paper is to have a quadrocopter accurately track three-dimensional periodic motions, without incurring large transients at the beginning of the motion. This research is motivated by the ultimate goal of performing quadrocopter choreographies along to music. To achieve precise synchronization to a given periodic (music) reference signal, and to achieve exact reference trajectory tracking, we concentrate on adapting the parameters of the feed-forward input signal sent to the vehicle controller. An example of the 3D periodic motion considered in this paper is pictured in Fig. 1.

Trajectory tracking with quadrocopters is typically achieved using feedback control approaches. Methods range from classical PID control, backstepping techniques and nonlinear control [1–4], to LQ optimal solutions and model predictive control, e.g. [5]. Such controllers, however, are not usually able to achieve high-performance trajectory tracking with zero phase lag for arbitrary periodic quadrocopter motions of varying angular frequencies. Perfect temporal accuracy can only be achieved by using different controllers (or controller parameters) for different motions and motion frequencies. Moreover, feedback control is inherently causal because the control actions depend only on past measurements. Causality, imperfect initial conditions and model errors effect an initial transient phase, in which the tracking errors are substantial.

For choreographies in which motions are changed in quick succession, designing separate controllers for different motions is impractical, and the transient behavior as described above is highly undesirable. Switching between different controllers may even cause instability.

In this paper, we propose a control strategy that builds upon the same basic trajectory-tracking controller for periodic motions at all frequencies, but adapts the parameters of the actual input to the controller in order to guarantee precise tracking and synchronization. These parameters can be identified prior to the flight performance to effectively reduce transient time and tracking errors.

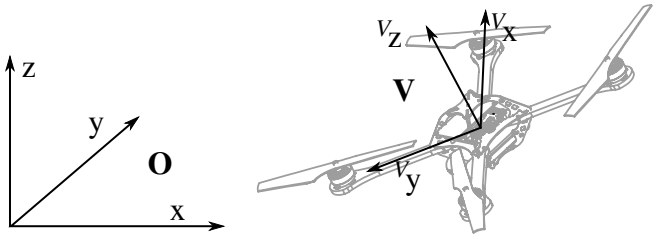



**Figure 2.** The inertial coordinate system **O** and the vehicle coordinate system **V** defining the vehicle attitude.

Other research on motion synchronization with external inputs has focused largely on real-time interaction between robots and the environment. This research mostly deals with real-time signal processing and synchronization schemes, and typically features humanoid robots that perform rhythmic motions such as dancing, drumming and singing in tempo with an exogenous signal [6–8]. In contrast, our work features aerial robots and *a priori* known reference signals.

The contribution of this paper is a feed-forward strategy that avoids the large transients, preserves the shape of the periodic motion and maintains high temporal accuracy, even in the first period of the motion. It is possible to adapt only the amplitude and phase of the motion, which is done either online, or offline prior to the actual flight performance. We show that, for directionally decoupled linear systems, the identification of offline parameters requires only a small number of experiments that can be stored concisely in a table. The general idea is based on first identifying the linear closed-loop transfer function and then using it to compensate for the steady-state errors in advance [9]. This 'black box' approach allows the strategy presented herein to be used for any (approximately) linear system with independent directions.

This feed-forward adaptation scheme was applied to quadrocopter music performances. Videos are available at `www.tiny.cc/MusicInMotion`.

The 3D periodic motion primitives considered in this paper are presented in Sec. 2. The quadrocopter dynamics and the trajectory-tracking controller are introduced in Sec. 3 and 4, respectively. The quadrocopter response to the motion primitives is investigated in Sec. 5, where an online parameter adaptation strategy is introduced and relevant system properties are derived. The system properties are then used in Sec. 6 to develop an offline parameter adaptation strategy. We show the effectiveness of the offline strategy by performing a sequence of motion primitives with a quadrocopter. Experiments are conducted in the Flying Machine Arena, an indoor test environment for quadrocopters. For a detailed description of the experimental setup, refer to [10].

## 2. Periodic Motions

We present a framework for periodic motion primitives in three dimensions, generalizing from the one-dimensional side-to-side motion in our previous work [11]. We specify motion primitives on the vehicle's position in the inertial coordinate system **O**, see Fig. 2. The heading of the quadrocopter (that is, the yaw angle in Z-Y-X Euler attitude representation) is held at zero. The remaining rotational degrees of freedom are defined by the quadrocopter dynamics, cf. [10]

and Sec. 3. The desired position of the quadrocopter at time $t$, $s_d(t) = (x_d(t), y_d(t), z_d(t))$ is given by

$$\begin{bmatrix} x_d(t) \\ y_d(t) \\ z_d(t) \end{bmatrix} = \begin{bmatrix} \delta_d^x \\ \delta_d^y \\ \delta_d^z \end{bmatrix} + \begin{bmatrix} A_d^x \cos(\omega_d^x t + \theta_d^x) \\ A_d^y \cos(\omega_d^y t + \theta_d^y) \\ A_d^z \cos(\omega_d^z t + \theta_d^z) \end{bmatrix}, \tag{1}$$

where $(\delta_d^x, \delta_d^y, \delta_d^z)$ is the desired center position of the motion primitive. The shape of the primitive is adjustable by a set of nine motion parameters: in each direction $i \in \{x, y, z\}$, we specify the desired amplitude $A_d^i$, frequency $\omega_d^i$ and phase $\theta_d^i$. By varying the nine parameters, a wide range of different motions can be expressed, such as side-to-side motions, bounces, ellipses, eights and spirals, see for example Fig. 1.

Not all parameter values result in periodic motion primitives that can be followed by the quadrocopter. If the amplitudes or frequencies are too high, the motion becomes infeasible due to thrust limitations of the propellers and limited sensor range. This is investigated in [10]. The motions considered in this paper are assumed to be feasible.

One of our goals is to synchronize the motion to an external reference signal (for example, the beat of a music piece), which sets the motion frequencies $\omega_d^i$. Ultimately, motion primitives can be arranged in a choreographic sequence and be timed to music [12].

## 3. Quadrocopter Dynamics

The translational motion of a quadrocopter in the inertial frame **O** is described by

$$\begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} = \mathbf{R}(t) \begin{bmatrix} 0 \\ 0 \\ c(t) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \Leftrightarrow \begin{array}{l} \ddot{x} = c\, b^x \\ \ddot{y} = c\, b^y \\ \ddot{z} = c\, b^z - g \end{array}, \tag{2}$$

where $\mathbf{R}(t)$ is the rotation matrix from the body frame **V** to the inertial frame **O**, $c(t)$ is the collective thrust of the four propellers, and $g$ is the acceleration due to gravity. The values $(b^x, b^y, b^z)$ correspond to the third column of the rotation matrix, namely $(\mathbf{R}_{13}, \mathbf{R}_{23}, \mathbf{R}_{33})$, and represent the direction of the collective thrust in the inertial frame **O**.

The rotation matrix **R** evolves according to

$$\dot{\mathbf{R}}(t) = \mathbf{R}(t) \begin{bmatrix} 0 & -r(t) & q(t) \\ r(t) & 0 & -p(t) \\ -q(t) & p(t) & 0 \end{bmatrix}, \tag{3}$$

where $(p, q, r)$ represent the quadrocopter angular body velocities around the body $({}^V x, {}^V y, {}^V z)$ axes, see Fig. 2 and [13]. The quadrocopter is controlled by four inputs: a collective thrust command $c_c$, and commanded angular body velocities $(p_c, q_c, r_c)$, see Fig. 3 and Sec. 4.1.

The controller design introduced below is based on the above dynamics model. For a more detailed quadrocopter model that includes rotational dynamics refer to [10].

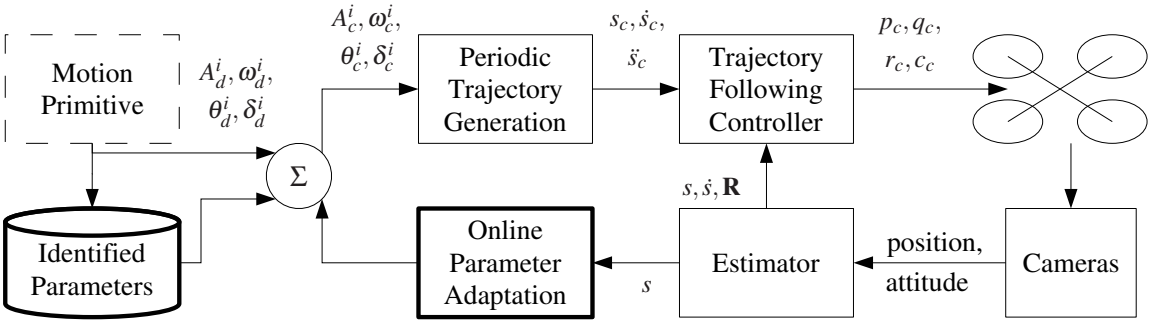



**Figure 3.** The control configuration shows the online and offline motion parameter adaptation (bold boxes), and the underlying trajectory-following controller.

## 4. Quadrocopter Control

The overall control configuration of our approach is outlined in Fig. 3. Below we describe the trajectory-following controller (TFC) that is the basis of our approach, and analyze its properties with particular regard to periodic motions as described in Sec. 2.

### 4.1 Approach

The TFC accepts position, velocity, and acceleration commands, denoted by $s_c(t)$, $\dot{s}_c(t)$ and $\ddot{s}_c(t)$ respectively, and attempts to maintain the quadrocopter on this specified trajectory. Control is based on the estimated quadrocopter position $s = (x, y, z)$, velocity $\dot{s}$ and attitude $\mathbf{R}$, see Fig. 3. The TFC outputs the commands $c_c$ and $(p_c, q_c, r_c)$ to the vehicle. The TFC consists of three separate loops for altitude, horizontal position, and attitude, see Fig. 4. While the TFC operates in discrete time, the controller design is based on the continuous-time system dynamics representation.

The altitude control is designed such that it responds to altitude errors $(z - z_c)$ like a second-order system with time constant $\tau_z$ and damping ratio $\zeta_z$,

$$\ddot{z} = -\frac{2\zeta_z}{\tau_z}(\dot{z} - \dot{z}_c) - \frac{1}{\tau_z^2}(z - z_c) + \ddot{z}_c. \tag{4}$$

It uses the collective thrust to achieve this. With (2) and (4), we obtain

$$c_c = (\ddot{z} + g)/b^z. \tag{5}$$

Similarly, the two horizontal position control loops are shaped based on (2) with $c_c$ from (5), resulting in the commanded rotation matrix entries $b_c^x$ and $b_c^y$. The attitude control is shaped such that the two rotation matrix entries $b^x, b^y$ react in the manner of a first-order system with time constant $\tau_{rp}$; that is, for $x$: $\dot{b}_c^x = (b^x - b_c^x)/\tau_{rp}$. The values $\dot{b}_c^x, \dot{b}_c^y$ are directly mapped to the commanded angular body velocities $(p_c, q_c)$ using (3) and the estimated attitude $\mathbf{R}$,

$$\begin{bmatrix} p_c \\ q_c \end{bmatrix} = \frac{1}{\mathbf{R}_{33}} \begin{bmatrix} \mathbf{R}_{21} & -\mathbf{R}_{11} \\ \mathbf{R}_{22} & -\mathbf{R}_{12} \end{bmatrix} \begin{bmatrix} \dot{b}_c^x \\ \dot{b}_c^y \end{bmatrix}. \tag{6}$$

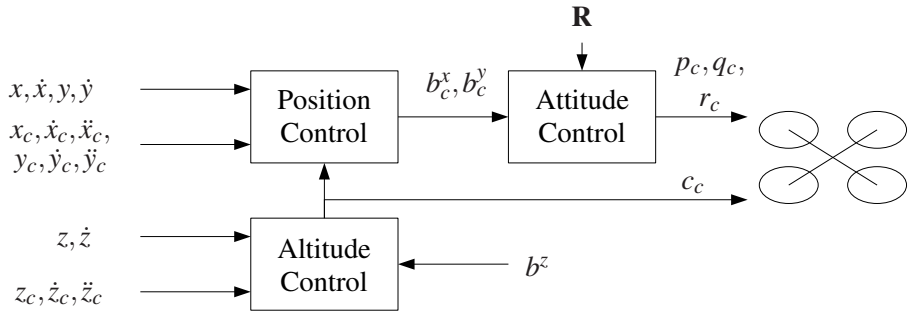



**Figure 4.** Cascaded control loops of the trajectory-following controller (TFC).

Vehicle yaw control can be considered separately, since rotations around the body ${}^V z$ axis do not affect the above dynamics. The yaw controller is a proportional controller and the resulting yaw angle rate is mapped to $r_c$ using the kinematic relations of Euler angles. The innermost loop, on board the quadrocopter, controls the angle rates $(p,q,r)$ to the calculated set points $(p_c,q_c,r_c)$.

The feedback loop closed by the TFC is responsible for maintaining the quadrocopter on a trajectory, which is provided by the periodic trajectory generation (PTG). The PTG is based on the motion primitives in Sec. 2 and implements (1) with motion parameters $\delta_c^i, A_c^i, \omega_c^i, \theta_c^i$, where the subscript '$c$' stands for 'commanded'. The commanded parameters are one of the following: simply the desired parameters (no adaptation); adapted online (see Sec. 5); or adapted both online and offline (see Sec. 6). Fig. 3 illustrates the three options.

### 4.2 Analysis

To highlight the key characteristics of the above control architecture, we analyze the closed-loop dynamics under the following simplifying assumptions: *(i)* the commanded collective thrust can be changed instantaneously, that is, $c(t) = c_c(t)$; *(ii)* the estimated rotation matrix entry $b^z$ corresponds to the actual one; and *(iii)* we have direct control over the other two rotation matrix entries, namely $b^x(t) = b_c^x(t)$ and $b^y(t) = b_c^y(t)$. Then, (2) can be written as

$$\begin{aligned} \ddot{x} &= u_x, \qquad \text{with} \quad u_x = f_x(t, b_c^x) = c(t) b_c^x(t), \\ \ddot{y} &= u_y, \qquad \text{with} \quad u_y = f_y(t, b_c^y) = c(t) b_c^y(t), \\ \ddot{z} &= u_z, \qquad \text{with} \quad u_z = f_z(t, c_c) = c_c b^z(t) - g, \end{aligned} \tag{7}$$

where $u_i,\ i \in \{x,y,z\}$, represent the flat inputs resulting from a feedback linearization [14]. Such a transformation between the virtual inputs $u_i$ and $b_c^x, b_c^y, c_c$ was applied in the previous controller equations, cf. (5), allowing us to use techniques from liner feedback control design. Eq. (7) decouples the three directions and shows linear system behavior in each direction. In the ideal case where the quadrocopter dynamics correspond to the model (2), the combination of feedback linearization with velocity and acceleration feed-forward (see $\dot{z}_c, \ddot{z}_c$ in (5)) results in perfect trajectory tracking.

Assumptions *(i)* and *(ii)* are good approximations due to the fast motor dynamics (with motor time constants more than six times faster than the controller time constants, cf. [15])

and due to precise attitude estimates that are based on high-accuracy camera measurements and that include a prediction step to compensate for system latencies. Assumption *(iii)* is true only if the system has zero rotational inertia (cf. [10]), which is not the case. In reality, exceptionally high angular accelerations (cf. [15]) can be achieved and rotational inertia terms are small. Thus, the overall closed-loop dynamics may still be expected to be approximately linear. The approximate directional independence and linear dynamics behavior is exploited in Sec. 5 and further investigated using experimental data, since additional effects of latencies, time discretization, on-board dynamics, and modeling errors are difficult to predict.

### 4.3 Results

In a first attempt, the desired periodic trajectory is directly fed to the vehicle controller (TFC); that is, $s_c(t) := s_d(t)$. The quadrocopter response is a sinusoidal motion with a (after the transient phase) *constant* change in amplitude, phase and center position. Phase shift and amplitude error are observed in each translational direction and are not necessarily equal in size. The frequency of the quadrocopter motion corresponds to the commanded one. This suggests that the quadrocopter controlled by the TFC can be regarded as a *linear* system, which explains the phase offset and amplitude amplification.

Fig. 5 (top figure) shows the result for a planar side-to-side motion. The amplitude error of the quadrocopter response (black solid line) is obvious, whereas the phase error between the reference trajectory and the actual quadrocopter response is hardly noticeable. In actual experiments, however, small phase errors are visible and audible when, for example, quadrocopter choreography is timed to music, as a phase shift causes a misalignment between the flight trajectory and the music beat. For the side-to-side motion, music beats may occur at the outermost points of the trajectory. In this case, humans are particularly sensitive to non-zero vehicle velocity at beat times. Correspondingly, the bottom plot of Fig. 5 illustrates the velocity of the quadrocopter at beat times, i.e. when the reference trajectory reaches its maximum or minimum value. Note that if different directions exhibit different a phase shift or amplitude error, the shape of the motion primitive can even be changed, see Fig. 6.

In the following section, we correct for the amplitude error and phase shift, and investigate the closed-loop behavior in more detail by analyzing the steady-state correction terms. Because the commanded amplitude is often amplified by the closed-loop system (rendering feasible commanded trajectories into infeasible quadrocopter motions), the correction is done first.

## 5. Online Correction

### 5.1 Approach

The goal is to accurately track the desired trajectory $s_d(t)$ by minimizing the deviation from the estimated trajectory $s(t)$. To this end, the motion parameters in the commanded trajectory $s_c(t)$ are adjusted by directionally decoupled integral controllers, see Fig. 3. The approach is based on our previous work, see [11].

For notational convenience, we drop the superscripts $i \in \{x,y,z\}$ indicating the direction

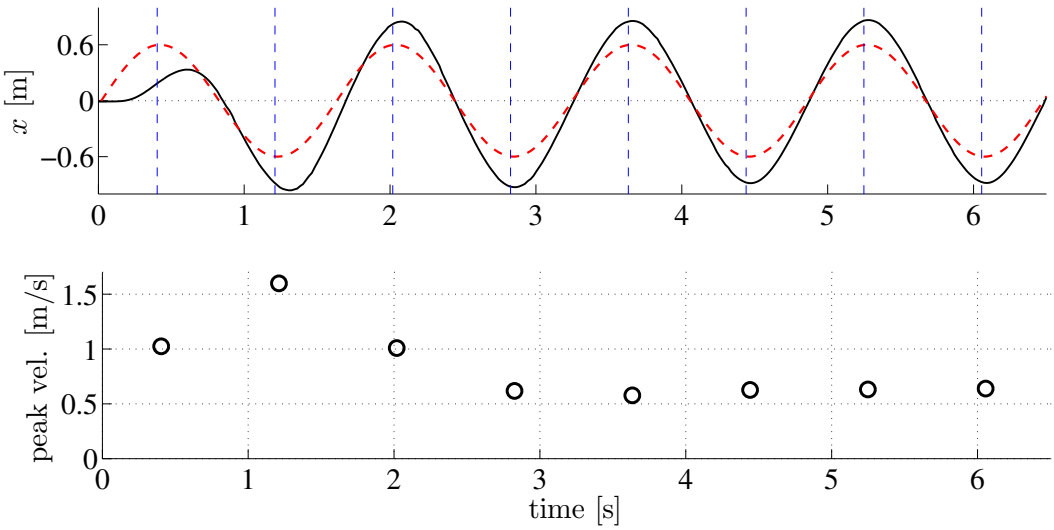



**Figure 5.** Side-to-side motion (with $\omega_d^x = 3.25\,\mathrm{rad/s}$, $A_d^x = 0.6\,\mathrm{m}$): *no motion parameter adaptation*. **Top:** quadrocopter response (solid) for a desired oscillation in the $x$ direction (dashed). **Bottom:** corresponding peak velocities, i.e. absolute value of vehicle velocity at the peaks of the desired trajectory. High peak velocities imply a large phase error.

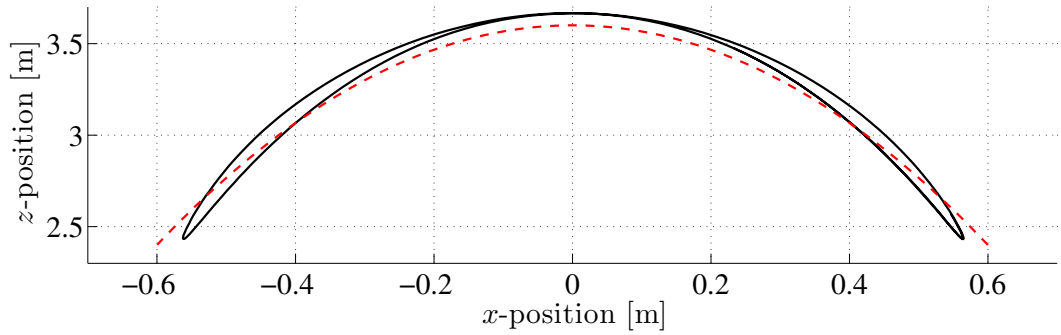



**Figure 6.** Vertical bounce motion (with $\omega_d^{x,y} = \omega_d^z/2 = 1.56\,\mathrm{rad/s}$, $A_d^{x,y} = A_d^z/2 = 0.6\,\mathrm{m}$): *no motion parameter adaptation.* The vehicle's response (solid) can differ in shape from the desired trajectory (dashed).

throughout this section. The motion parameters of the commanded trajectory are set to

$$\theta_c(t) = \theta_d + \theta_{\mathrm{on}}(t), \qquad A_c(t) = A_d + A_{\mathrm{on}}(t), \qquad \delta_c(t) = \delta_d + \delta_{\mathrm{on}}(t),$$

where the subscript 'on' indicates the online correction terms. They are updated in real time, during the flight.

We first determine the additive error in amplitude $A_t$, phase $\theta_t$ and position $\delta_t$ of the quadrocopter response at time $t$. We multiply the position estimate $s(t)$ and with the two reference signals, $r_{\cos}(t) = \cos(\omega_d t + \theta_d)$ and $r_{\sin}(t) = \sin(\omega_d t + \theta_d)$, and integrate the result over $N$ periods, that is $T = (2\pi N)/\omega_d$. We assume that the errors stay constant over the interval $[t-T, t]$.

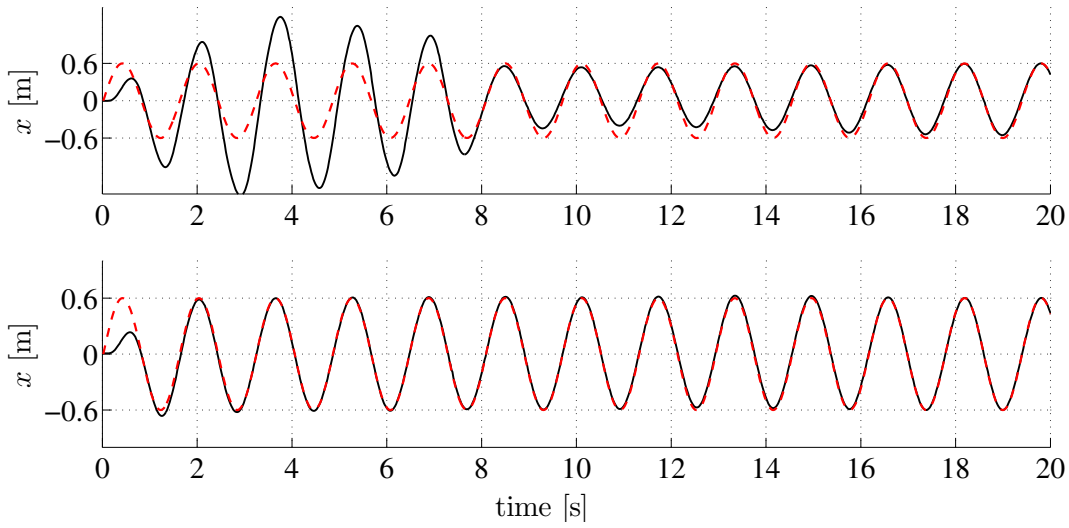



**Figure 7.** Side-to-side motion. **Top:** *online motion parameter adaptation only,* quadrocopter response (solid) for a desired oscillation in the $x$ direction (dashed). **Bottom:** *offline motion parameter adaptation,* with online motion parameter adaptation turned on after 2 periods.

This yields

$$\begin{aligned} \eta_1(t) &= \frac{1}{T}\int_{t-T}^{t} s(t) r_{\cos}(t)\, dt = \frac{A_t + A_d}{2}\cos(\theta_t) \\ \eta_2(t) &= \frac{1}{T}\int_{t-T}^{t} s(t) r_{\sin}(t)\, dt = \frac{A_t + A_d}{2}\sin(\theta_t), \end{aligned} \tag{8}$$

and finally

$$A_t = 2\sqrt{\eta_1(t)^2 + \eta_2(t)^2} - A_d, \qquad \theta_t = -\arctan\left(\eta_2(t)/\eta_1(t)\right). \tag{9}$$

The position error of the sinusoidal response is $\delta_t = \int_{t-T}^{t}(\delta_d - s(t))dt$. Since an insufficient number of measurements is available at the beginning of a motion, the integrals deliver reliable values only after several periods of the motion primitive. The online correction terms are calculated by integrating the errors according to

$$A_{\text{on}}(t) = k_A \int_0^t A_\tau\, d\tau, \qquad \theta_{\text{on}}(t) = k_\theta \int_0^t \theta_\tau\, d\tau, \tag{10}$$

and similarly for $\delta_{\text{on}}(t)$. The gains $k_\theta$, $k_A$, $k_\delta$ are chosen to ensure converge of the online correction terms to the steady-state values $\theta_{\text{on},\infty}$, $A_{\text{on},\infty}$ and $\delta_{\text{on},\infty}$, respectively. Note again that the above online parameter adaptation strategy is implemented for each direction separately.

## 5.2 Results

Using the proposed online parameter adaptation strategy, the errors in amplitude, phase and center position are effectively regulated to zero, see Fig. 7. We observe a substantial transient phase before the online correction terms attain steady state, see Fig. 8. This is mainly due to

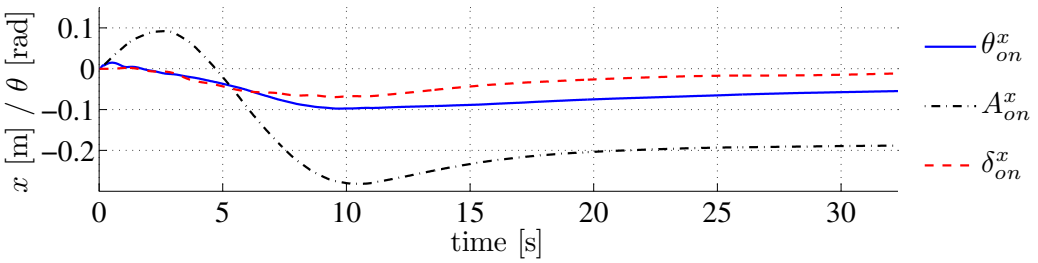



**Figure 8.** Side-to-side motion: convergence of the online correction terms.

the fact that the error identification scheme (8)-(9) only provides reliable values after several periods.

To draw further conclusions, we consider the steady-state values in the following form: the amplitude-normalized amplification factor,

$$\alpha^i_{\text{on},\infty} = (A^i_d + A^i_{\text{on},\infty})/A^i_d\,, \qquad i \in \{x,y,z\}\,, \tag{11}$$

and the steady-state phase and offset, $\theta^i_{\text{on},\infty}$ and $\delta^i_{\text{on},\infty}$, as before.

We found that, when executing the *same* motion primitive multiple times, the standard deviations of the corresponding steady-state values are small. We call this the *intra-class variability*, which is a measure of the repeatability of the experiments. We then investigated the *inter-class variability*: the standard deviation of the steady-state values $\alpha^i_{\text{on},\infty}, \theta^i_{\text{on},\infty}$, and $\delta^i_{\text{on},\infty}$ is evaluated for *different* motion primitives. We found that the inter-class variability of the obtained steady-state values for a given translational direction at a given directional frequency is of the same order of magnitude as the intra-class variability. Supported by the considerations in Sec. 4.2 and by experiments shown below, this leads to the following conclusions:

**Decoupled directions** The steady-state values $\alpha^i_{\text{on},\infty}, \theta^i_{\text{on},\infty}$ in each direction are independent of the motion's components in the other directions. Moreover, as expected from the quadrocopter's symmetry, the $x$ and $y$ directions exhibit the same behavior.

**Linear behavior** Considering the motion component in one direction $i$, the steady-state values depend only on the motion's frequency in this direction $\omega^i_d$.

Fig. 9 depicts the amplification factor $\alpha^l_{\text{on},\infty}$ and steady-state phase $\theta^i_{\text{on},\infty}$ against the motion's directional frequency $\omega^i_d$ for the two directions $i \in \{x,y\}$. Plots are shown for ten different periodic motion primitives in 1D, 2D and 3D of various amplitudes up to 0.6 m and different relative phase shifts. The standard deviation of the steady-state terms is indicated by the vertical labels. In particular, the variability of the amplification factors translates, for the largest amplitude, to a residual deviation of $\pm 1.5$ cm, which lies within the TFC hover accuracy of $\pm 2$ cm. The variability of the phase translates to a residual time shift of $\pm 25$ ms (at maximum), which is within the human audiovisual synchrony perception limits [16], and therefore likewise negligible.

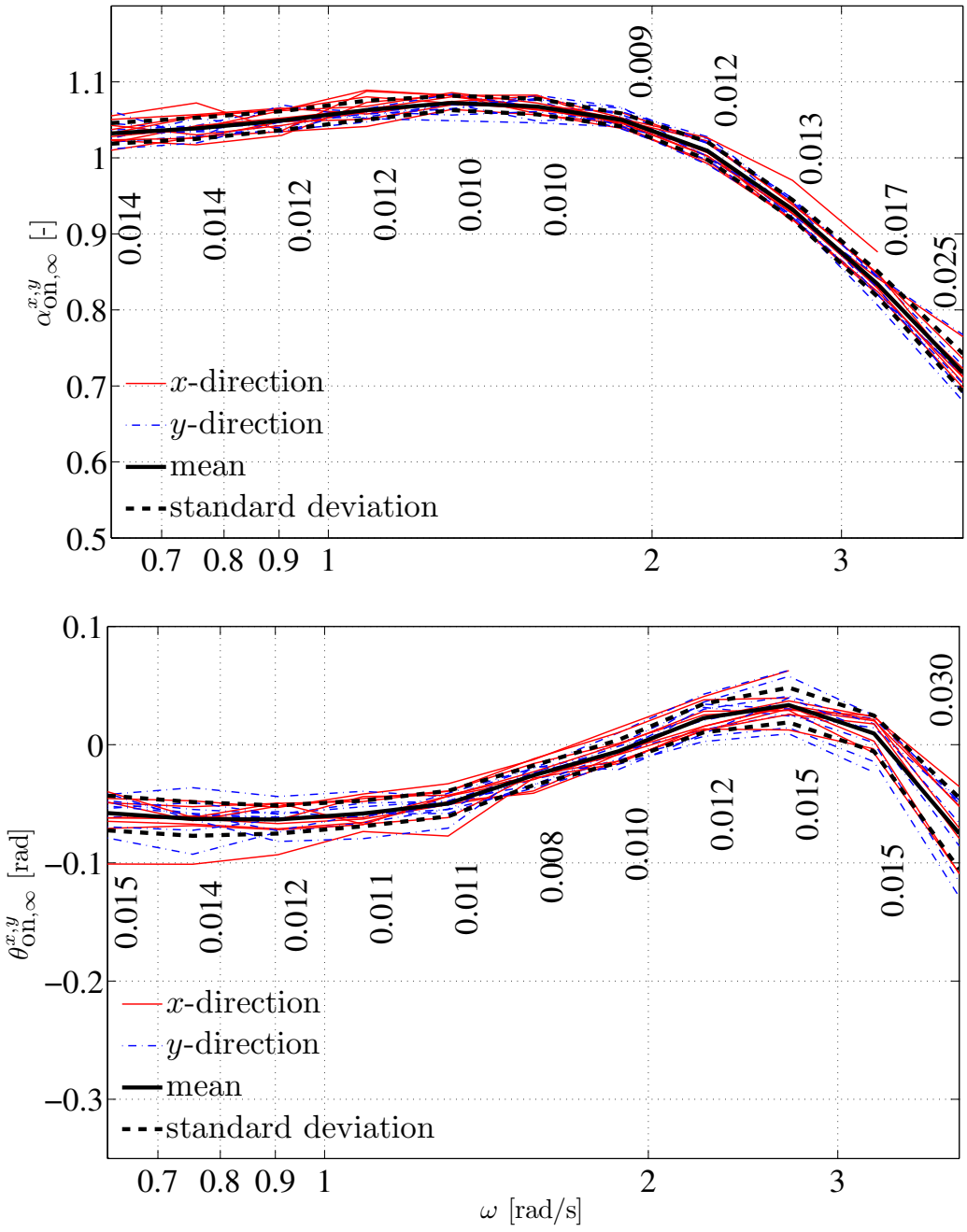



**Figure 9.** Steady-state correction values in $x$ and $y$ for ten different motions primitives in 1D, 2D and 3D. Here, $\omega = \omega_d^x = \omega_d^y$. The vertical labels provide the standard deviation for the samples. **Top:** steady-state amplification factor $\alpha^{x,y}_{\text{on},\infty}$. **Bottom:** steady-state phase $\theta^{x,y}_{\text{on},\infty}$.

These results affirm the linearity and directional independence property. Note that when performing the identification run with the same motion primitives several times, the variability is of the same order of magnitude. Consequently, we do not lose accuracy when identifying $\theta^i_{\text{on},\infty}$, $A^i_{\text{on},\infty}$ for one motion primitive and later using it for another one as described in the next section.

The steady-state correction terms for the center position $\delta^i_{\text{on},\infty}$ lie within the hover accuracy with a variability of the same magnitude. Thus, the value $\delta^i_{\text{on},\infty}$ cannot be called repeatable and may either be neglected because of its small average size or identified each time when flying.

## 6. Offline Identification

### 6.1 Approach

The previous section showed that the steady-state values obtained from the online correction are repeatable. Consequently, relevant steady-state values can be extracted once, and later applied to improve the transient performance. We employ offline identified parameters in addition to the online adaptation. Again, we drop the superscripts indicating the direction. The parameters of the commanded trajectory are set to

$$A_c(t) = \alpha_{\text{off}} A_d + A_{\text{on}}(t), \;\; \delta_c(t) = \delta_d + \delta_{\text{on}}(t), \;\; \theta_c(t) = \theta_d + \theta_{\text{off}} + \theta_{\text{on}}(t),$$

where the subscript 'off' indicates the offline motion parameters identified prior to the experiment. Note that there is no offline parameter for the center point $\delta_c$, because the errors are small in size and less repeatable and, thus, more efficiently handled by the online adaptation strategy. The offline parameters are selected at the start of a motion on the basis of the desired motion primitive and stay constant for the duration of the executed motion. For a given motion, the offline correction terms are set to the steady-state values obtained from the online correction, see Sec. 5.2:

$$\alpha_{\text{off}} = \alpha_{\text{on},\infty}, \qquad \theta_{\text{off}} = \theta_{\text{on},\infty}.$$

We make use of the directional independence and linearity property derived above to efficiently identify the offline correction terms for all periodic motions that can be expressed in our framework (1). We perform a *single* identification run with fixed amplitudes $A^i_d$, phases $\theta^i_d$ and center point $\delta^i_d$ and vary only the frequency $\omega^i_d = \omega$. Moreover, since the $x$ and $y$ direction exhibit the same dynamics, an identification run with a 2D motion primitive in $x$ or $y$, and $z$ is sufficient to completely identify all necessary feed-forward parameters.

Conceptually, the offline identification strategy results in a pair of maps

$$\Gamma_{xy} : \omega_d^{x,y} \mapsto (\alpha_{\text{off}}^{x,y}, \theta_{\text{off}}^{x,y}), \qquad \Gamma_z : \omega_d^z \mapsto (\alpha_{\text{off}}^z, \theta_{\text{off}}^z),$$

where the superscript $x,y$ indicates that this map is used for both, the $x$ and $y$ direction. The values are stored in a table with rows $[\omega_d^{x,y} \;\; \omega_d^z \;\; \theta_{\text{off}}^{x,y} \;\; \theta_{\text{off}}^z \;\; \alpha_{\text{off}}^{x,y} \;\; \alpha_{\text{off}}^z]$. We use linear interpolation between the offline parameters, which are only obtained at a discrete set of frequencies.

### 6.2 Results

As compared to using only online parameter adaptation, the proposed offline identification substantially decreases the transient phase, see Fig. 7. The offline parameters are effective from

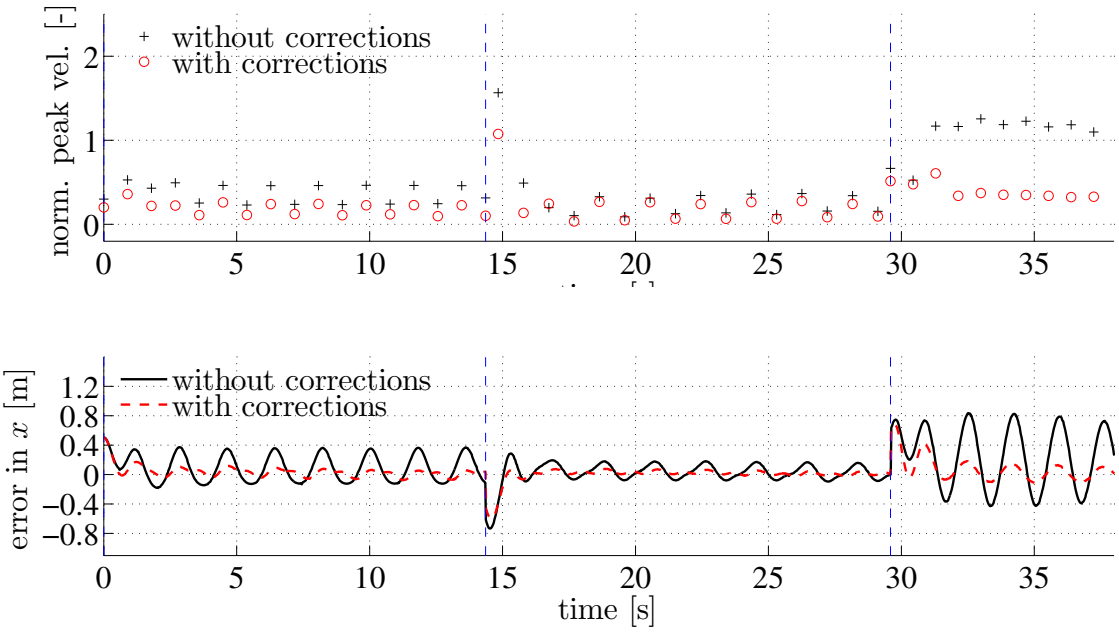



**Figure 10.** Sequence of motions (comprising a circular motion in 3D, a swing motion in 3D and a horizontal circle): *with and without feed-forward corrections*. Offline correction terms were obtained from a *reduced identification*. **Top:** absolute value of velocities at the peaks of the desired trajectory normalized by dividing by $A_d^x \omega_d^x$. **Bottom:** errors between desired trajectory and vehicle response ($s_d(t) - s(t)$) are plotted.

the start of the motion primitive. When combining online and offline correction, the former is used only after several periods.

In order to show the effectiveness of the reduced identification scheme, we perform a sequence of periodic 3D motions with offline parameters obtained from an oscillatory motion in 2D ($A_d^x = A_d^z = 0.4\,\text{m}$, $\omega_d^x = \omega_d^z = \omega$). Fig. 10 shows that the quadrocopter's deviation from the desired trajectory is clearly reduced when using the offline parameter adaptation strategy. As a consequence, the corresponding peak velocities (cf. Fig. 5) are also small indicating that the phase error is reduced. Note that the transient performance can be further improved by starting the vehicle with the appropriate velocity and acceleration.

The videos at `www.tiny.cc/MusicInMotion` show examples of quadrocopter choreographies timed to music.

## 7. Conclusions

In this paper we studied a feed-forward parameter tuning strategy that improves the tracking performance of periodic motion primitives, as compared to pure feedback control, especially during transients. With pre-identified correction terms, the tracking converges to a level virtually imperceptible to a human observer within a single period. The parameter correction terms depend only on the motion's 3D directional frequencies. The translational directions are inde-

pendent, allowing for an efficient offline identification of correction values. Due to the directional independence, the approach presented in this paper can be applied even to non-periodic 3D motions that are composed of periodic motions in each translational direction.

## References

[1] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 2247–2252.

[2] S. Al-Hiddabi, "Quadrotor control using feedback linearization with dynamic extension," in *Proceedings of the International Symposium on Mechatronics and its Applications (ISMA)*, 2009, pp. 1–3.

[3] T. Lee, M. Leoky, and N. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[5] C. Castillo, W. Moreno, and K. Valavanis, "Unmanned helicopter waypoint trajectory tracking using model predictive control," in *Proceedings of the Mediterranean Conference on Control Automation*, 2007, pp. 1–8.

[6] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2459–2464.

[7] J.-J. Aucouturier, "Cheek to chip: dancing robots and AI's future," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 74–84, 2008.

[8] D. Grunberg, R. Ellenberg, I. H. Kim, J. H. Oh, P. Y. Oh, and Y. E. Kim, "Development of an autonomous dancing robot," *International Journal of Hybrid Information Technology*, vol. 3, no. 2, 2010.

[9] P. Allen, "Feed-forward compensated high switching speed digital phase-locked loop frequency synthesizer," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 371–374, 1999.

[10] A. P. Schoellig, M. Hehn, S. Lupashin, and R. D'Andrea, "Feasibility of motion primitives for choreographed quadrocopter flight," in *Proceedings of the American Control Conference (ACC)*, 2011, pp. 3843–3849.

[11] A. P. Schoellig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3355–3360.

[12] A. P. Schoellig, F. Augugliaro, and R. D'Andrea, "A platform for dance performances with multiple quadrocopters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) - Workshop on Robots and Musical Expressions*, 2010, pp. 1–8.

[13] P. C. Hughes, *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

[14] A. Isidori, *Nonlinear control systems*. Springer Verlag, 1995.

[15] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *IFAC World Congress*, vol. 18, 2011, pp. 1485–1491.

[16] R. Arrighi, D. Alais, and D. Burr, "Perceptual synchrony of audiovisual streams for natural and artificial motion sequences." *Journal of Vision*, vol. 6, no. 3, pp. 260–268, 2006.

Paper VIII

# Feasibility of Motion Primitives for Choreographed Quadrocopter Flight

Angela P. Schoellig · Markus Hehn · Sergei Lupashin · Raffaello D'Andrea

**Abstract**

This paper describes a method for checking the feasibility of quadrocopter motions. The approach, meant as a validation tool for preprogrammed quadrocopter performances, is based on first principles models and ensures that a desired trajectory respects both vehicle dynamics and motor thrust limits. We apply this method towards the eventual goal of using parameterized motion primitives for expressive quadrocopter choreographies. First, we show how a large class of motion primitives can be formulated as truncated Fourier series. We then show how the feasibility check can be applied to such motions by deriving explicit parameter constraints for two particular parameterized primitives. The predicted feasibility constraints are compared against experimental results from quadrocopters in the ETH Flying Machine Arena.

Published in *Proc. of the 2011 American Control Conference (ACC),* 2011.

## 1. Introduction

Our goal is to derive motion primitives for quadrocopter flight choreography, where we define 'choreography' as the design and arrangement of expressive sequences of movements.

Motion primitives are short and fairly simple basic motion elements; when concatenated, they can describe complex behavior and are often used to represent repetitive movements such as, for example, human hand-writing [1] or human body gestures [2]. Motion primitives are also used as a tool for simplifying complex problems, including motion planning [3–5], the control of humanoid robots [6–8], object recognition in video [9] or motion extraction from large data sets [10]. In particular, dance movement is often described by motion primitives because of its repetitive and rhythmic form [11–14].

In this paper, we introduce motion primitives as basis elements for choreographed dance-like quadrocopter movements. The design of these primitives is guided by the four key variables of dance as described by professional dancers and choreographers: time, space, energy, and structure. We present motion primitives that are adjustable in their temporal characteristics as well as in their spatial features. A wide spectrum of movements and motion segments are included in the library of motion primitives, ranging from sharp and energetic movements to soft and smooth ones. With this library, we aim at providing a choreographer with degrees of freedom for creating an expressive choreography comparable to a human dance performance.

Like humans, whose range of motion and speed of movement is limited, not all motions are feasible for a quadrocopter. Thus, a large part of the analysis below is devoted to determining the set of parameters that represents motion sequences that can be realized with a quadrocopter. The resulting library of feasible motion primitives allows for multifaceted choreographies that could eventually be synchronized to music, in order to create a novel visual musical experience as described in [15].

This paper is organized as follows: Sec. 2 introduces a general description for motion primitives that define the translational dynamics of the quadrocopter and can be related to the four key elements of dance. In Sec. 3, the equations governing the dynamic behavior and constraints of the quadrocopter are stated. This allows us to derive inequalities for determining the feasibility of trajectories in Sec. 4. To illustrate the effectiveness of this procedure, feasible parameter sets are explicitly calculated for two motion primitives and validated by experimental data (Sec. 5). We conclude the paper with a summary in Sec. 7.

## 2. Motion Primitives

Our goal is to develop basic motion elements that, when combined into sequences, allow for a multifaceted, meaningful quadrocopter choreography. We specify motion primitives on the quadrocopter's translational position $s(t) = (x(t), y(t), z(t))$ measured in the inertial coordinate system $\mathbf{O}$, see Fig. 2. The remaining degrees of freedom, namely the vehicle's attitude $\mathbf{V}$, are not considered in the description of the motion primitive, but are partly defined by the quadrocopter's dynamics, see Sec. 3. Motion primitives are introduced as

$$s_d(t) = s_d(p,t), \tag{1}$$

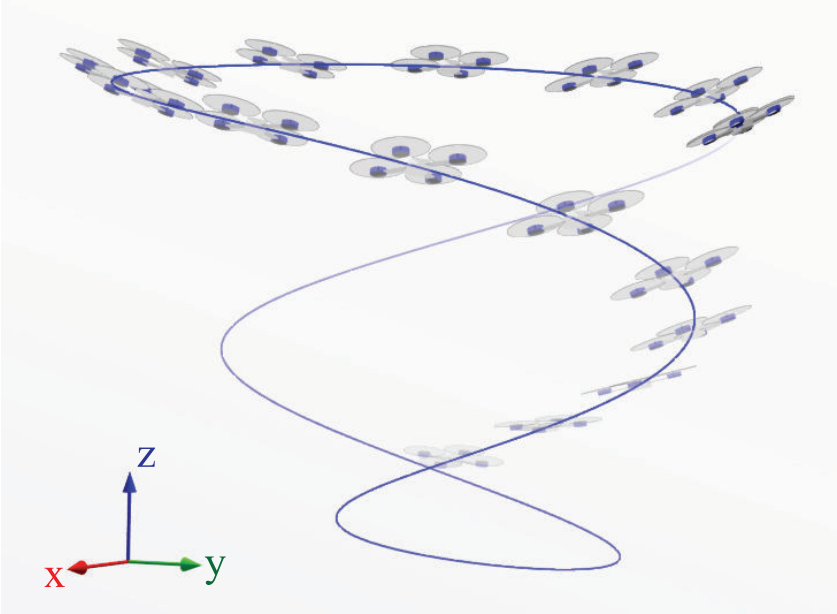



**Figure 1.** An example of a periodic motion primitive studied in this paper.

over a finite time interval $t \in [t_0, t_f] \subset \mathbb{R}$, $t_f < \infty$, where $p$ denotes the set of adjustable motion parameters. Parameterized motion primitives allow for variety and expressiveness in the choreography design and provide choreographers with intuitive tools for the design of performances.

Our objective is to offer a similar range of motions as is used in human dance composition. In this context, we ask: Which choices does a professional dance choreographer have when creating a performance? How can we provide the tools and degrees of freedom necessary for implementing an expressive performance on the quadrocopter?

Four fundamental choreographic elements – time, space, energy, and structure – are commonly used by professional dancers, choreographers and dance teachers to build choreography with interest, dynamics and aesthetic appeal, cf. [16, 17]. These parameters provide a framework for meaningful quadrocopter choreography, and are described as follows:

**Space** refers to the area the dancer is performing in. It also relates to how the dancer moves through the area, as characterized by the direction and path of a movement, as well as its size, level, and shape.

**Time** encompasses rhythm, tempo, duration, and phrasing of movements. Using time in different combinations can create intricate visual effects. Ideas such as quick-quick, slow or stop movements are examples.

**Energy** relates to the quality of movement. This concept is recognizable when comparing ballet and tap dance. Some types of choreography are soft and smooth, while others are sharp and energetic.

**Structure** represents the organization of movement sequences into larger concepts: the combination and variation of movements using recurring elements, contrast, and repetition. Movements can even follow a specific story line to convey certain information through a

dance.

Examples illustrating the four elements of dance are found in [16, 17].

One way of introducing parameterized motion primitives that capture a wide range of different movements is as a Fourier series [18],

$$s_d(t) = a_0 + \sum_{k=1}^{N} a_k \cos(k\Omega t) + b_k \sin(k\Omega t), \tag{2}$$

where $\Omega = 2\pi/T$ represents the fundamental angular frequency corresponding to a period of $T$. Additional design parameters are the constant vectors $a_0, a_k, b_k \in \mathbb{R}^3,\ k \in \mathscr{K} = \{1,2,\ldots,N\}$, and $N \geq 1$; that is, $p = \{\Omega, N, a_0, a_k, b_k \,|\, k \in \mathscr{K}\}$. The parameters $p$ allow us to express the key choreographic elements:

**Space** The parameters $a_0$ and $a_k, b_k,\ k \in \mathscr{K}$ define the amplitudes of the periodic motion and, thus, the spacial dimension of the movement. These vectors also specify the direction of the motion and the overall three-dimensional shape of the curve.

**Time** The underlying rhythm is given by the frequency $\Omega$. When the choreography is set to music, the frequency $\Omega$ can be related to the music's tempo. Different tempos are combined when choosing $N > 1$. The overall duration of the motion can be adjusted via $t_f$.

**Energy** The higher the value of $N$, the more energetic and sharp are the possible motions, cf. [18].

**Structure** The motion primitives described in (2) can be combined into sequences, which can in turn be combined to create an overall choreographic performance. Endless permutations are possible, much the way individual words can be combined into a variety of sophisticated stories.

In short, the general motion description (2) reflects the fundamental choregraphic elements and allows for a multi-dimensional choreography. Out of the variety of motions captured by (2), Fig. 1 illustrates the one with $N = 3$, $T = 10$, $a_0 = (0,0,3)$, $a_1 = (0,0,1)$, $a_2 = (1,0,0)$, $b_3 = (0,1,0)$ and $a_3, b_1, b_2$ being zero. A Matlab file for generating arbitrary motion primitives of the proposed type are available online at `www.idsc.ethz.ch/Downloads/QuadDance`.

In order to make (1) and (2) a useful tool for choreographers, we need to specify which motion primitives can be realized on the vehicle. The dynamics and physical limits of the quadrocopter define the feasible sets of parameters $p$.

## 3. Quadrocopter Dynamics and Constraints

The quadrocopter dynamics and constraints are derived from first principles models:

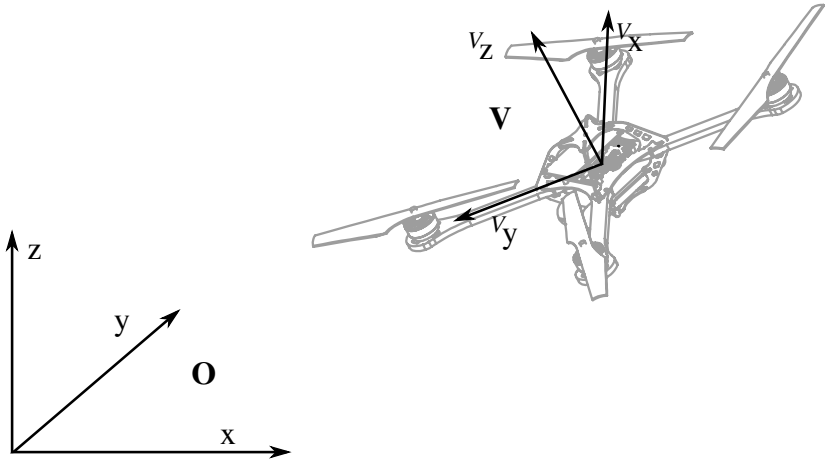



**Figure 2.** The inertial coordinate system **O** and the vehicle coordinate system **V**.

### 3.1 Dynamics

The quadrocopter is described by six degrees of freedom: The translational position $s = (x, y, z)$ is measured in the inertial coordinate system **O** as shown in Fig. 2. The vehicle attitude is defined by the body-fixed frame **V** and represented by the Euler angles yaw, pitch and roll, $(\alpha, \beta, \gamma)$. The rotation matrix ${}^{O}_{V}R(\alpha, \beta, \gamma)$ for transforming coordinates from **V** to **O** is

$$ {}^{O}_{V}R(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_x(\gamma), \tag{3} $$

where

$$ R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}, \qquad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{4} $$

$$ R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5} $$

The vector $s$ describes the center of mass of the vehicle in the inertial coordinate system **O**. The translational acceleration of the vehicle is dictated by the attitude of the vehicle and the total thrust produced by the four propellers. The translational dynamics in the inertial frame are given by

$$ \ddot{s} = {}^{O}_{V}R(\alpha, \beta, \gamma) \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \mathrm{g} \end{bmatrix}, \tag{6} $$

where $g$ is the acceleration due to gravity and $f$ is the sum of the rotor forces $F_i$ normalized by the vehicle mass $m$,

$$ f = \sum_{i=1}^{4} f_i \qquad \text{with} \quad f_i = F_i/m. \tag{7} $$

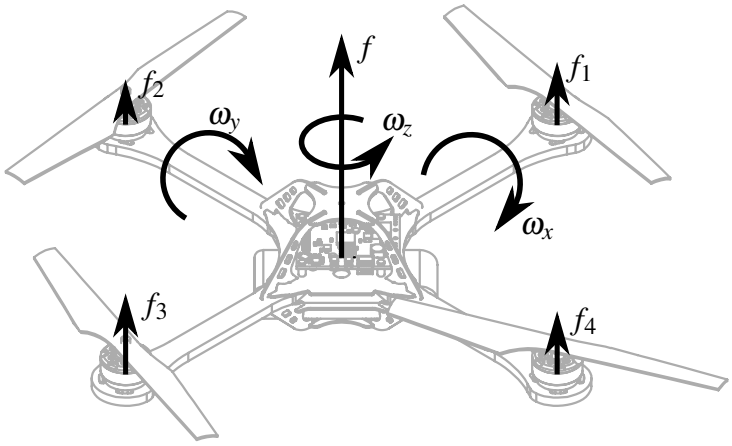



**Figure 3.** The control inputs of the quadrocopter are the body rates $\omega_x$, $\omega_y$, and $\omega_z$ and the collective thrust $f$. These inputs are converted by an onboard controller into motor forces $f_i$, $i \in \{1,2,3,4\}$.

The control inputs to the vehicle are the mass-normalized collective thrust $f$ and the desired rotational rates about the vehicle body axes, $\omega = (\omega_x, \omega_y, \omega_z)$, see Fig. 3.

The relationship between the body-fixed angular velocity vector $\omega$ and the rate of change of the Euler angles is

$$\omega = \begin{bmatrix} \cos\beta\cos\gamma & -\sin\gamma & 0 \\ \cos\beta\sin\gamma & \cos\gamma & 0 \\ -\sin\beta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix}. \tag{8}$$

Each rotor of the quadrocopter produces not only a force $F_i$, $i \in \mathscr{I} = \{1,2,3,4\}$, in the positive ${}^{V}z$ direction, but also a reaction torque $M_i$ perpendicular to the plane of rotation of the blade, see Fig. 3, where

$$M_i = kF_i, \qquad k = \text{const}, \tag{9}$$

describes the relationship between the motor force $F_i$ and the associated reaction torque $M_i$. The parameter $k$ is given by the motor characteristics, see [19] for details. Rotors 1 and 3 rotate in the negative ${}^{V}z$ direction, producing a moment that acts in the positive ${}^{V}z$ direction; while rotors 2 and 4 rotate in the opposite direction resulting in reaction torques in the negative ${}^{V}z$ direction. Given the inertia matrix $I$ with respect to the center of mass and the vehicle frame $\mathbf{V}$, the rotational dynamics of the body-fixed frame are given by

$$I\dot{\omega} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ k(F_1 - F_2 + F_3 - F_4) \end{bmatrix} - \omega \times I\omega, \tag{10}$$

where $L$ is the distance from each motor to the center of the quadrocopter. The vehicle's principal axes coincide with the vehicle frame axes, resulting in a diagonal inertia matrix with entries $(I_x, I_y, I_z)$, where $I_x = I_y$ because of symmetry.

### 3.2 Constraints

The agility of the quadrocopter is constrained by the minimum and maximum force of a single motor,

$$f_{i,min} \leq f_i \leq f_{i,max}, \qquad i \in \{1,2,3,4\}, \tag{11}$$

where we use the mass-normalized representation of the forces. The forces are always positive, $f_{i,min} \geq 0$, since the motors can spin only in one direction. Assuming identical motors, the collective thrust is bounded by

$$f_{min} \leq f \leq f_{max} \qquad \text{with} \quad f_{min} = 4f_{i,min}, \quad f_{max} = 4f_{i,max}. \tag{12}$$

In the following feasibility analysis, we study the nominal dynamics of the vehicle under the assumption that we can control the vehicle body rates directly, and ignore rotational acceleration dynamics. We also assume that the collective thrust can be changed instantaneously.

We justify the above assumption based on experimental results that show a fast response time to changes in the desired rotational rates (time constants are on the order of 20 ms). A high-bandwidth controller on the vehicle tracks the desired rates using feedback from gyroscopes. Because the quadrocopter has very low rotational inertia and can produce high torques due to the outwards mounting of the propellers, see Tab. 1, high rotational accelerations in the order of 200 rad/s$^2$ are achievable. The true thrust dynamics are as fast as the rotational dynamics, with propeller spin-up being faster than spin-down.

This allows us to calculate the motor forces $f_i(t)$ for a desired motion primitive from the nominal inputs, $\omega_d(t)$ and $f_d(t)$, using equations (7) and (10).

## 4. Feasibility of Motion Primitives

For the subsequent feasibility analysis, we assume that motion primitives, cf. (1), are twice-differentiable in time. Feasibility is formulated in terms of the motor limits $c = \{f_{i,min}, f_{i,max}\}$ and the motion parameters $p$. The objective is to derive a set of inequalities that specify feasible parameter sets $p$ given the limits $c$,

$$\overline{h}(p,c,t) \leq 0 \qquad \forall t \in \left[t_0, t_f\right]. \tag{13}$$

In other words, given the vehicle limits $c$, a parameter set $p$ is feasible if (13) is satisfied over the whole time interval $t \in \left[t_0, t_f\right]$.

The vehicle is constrained by the minimum and maximum force of a single rotor, cf. (11), which in turn results in a minimum and maximum collective thrust (12). Each of the aforementioned constraints must be satisfied in order to guarantee the feasibility of a given motion primitive,

$$\overline{h}(p,c,t) = \begin{bmatrix} h(p,c,t) \\ h_1(p,c,t) \\ h_2(p,c,t) \\ h_3(p,c,t) \\ h_4(p,c,t) \end{bmatrix} \leq 0, \qquad \forall t \in \left[t_0, t_f\right], \tag{14}$$

where $h(p,c,t)$ represents the constraint on the collective thrust (12) and $h_i(p,c,t),\ i \in \mathscr{I}$ the motor limits (11). The above inequality is defined componentwise. Note that the inequalities on the single motors $h_i(p,c,t),\ i \in \mathscr{I}$, alone would be sufficient for investigating the feasibility of a motion primitive. However, we keep $h(p,c,t)$, since this inequality can be can be derived directly from (6) and checked easily, as shown below.

### 4.1 Collective Thrust Limit

To derive $h(p,c,t)$ for a desired motion primitive $s_d$, we re-write (6),

$$ {}^{O}_{V}R(\alpha,\beta,\gamma)\, n\, f_d = \ddot{s}_d + n\, g\,, \tag{15} $$

where $n = [0,0,1]$ and $f_d$ is the nominal thrust input associated with $s_d$. Taking the 2-norm, we can solve for $f_d$, $f_d \geq 0$,

$$ \left\| {}^{O}_{V}R(\alpha,\beta,\gamma)\, n\, f_d \right\| = \|\ddot{s}_d + n\, g\| \quad \Leftrightarrow \quad f_d = \|\ddot{s}_d + n\, g\|\,. \tag{16} $$

Recalling that $s_d = s_d(p,t)$ and (12), the constraint guaranteeing the maximum and minimum bound of the collective thrust, is

$$ h(p,c,t) = \begin{bmatrix} \|\ddot{s}_d(p,t) + n\, g\| - f_{max} \\ f_{min} - \|\ddot{s}_d(p,t) + n\, g\| \end{bmatrix} \leq 0\,. \tag{17} $$

This feasibility requirement can be checked for any given desired motion primitive $s_d(p,t)$ by calculating its second time derivative. No further calculations are necessary. In particular, the nominal input associated with $s_d(p,t)$ need not be determined in advance. The constraint (17) on the collective thrust guarantees that the translational dynamics (6) are satisfied. Most importantly, it excludes the majority of infeasible parameters $p$. For a more detailed discussion on this topic see Sec. 4.3 and the examples in Sec. 5.

### 4.2 Motor Saturation

In order to evaluate the motor constraints (11), we must determine the (nominal) rotational inputs $\omega_d(t)$ of the given motion primitive $s_d(t)$. Given $\omega_d(t)$, we can, for the motor forces $f_{i,d}(t),\ i \in \mathscr{I}$, solve a linear system of equations, (7) and (10), and check their feasibility based on (11). Note that in specifying a trajectory by its translational degrees of freedom, we are free to choose the rotational rate $\omega_z(t)$. For the general case (1), the rotational inputs $\omega_{x,d}(t)$ and $\omega_{y,d}(t)$ are obtained by numerically integrating the dynamic equations (6), (8) and (10), and using the result (16). Below, we propose problem-specific analytic solutions for two simple examples and state $h_i(p,c,t)$ explicitly.

### 4.3 Discussion

The collective thrust constraint $h(c,p,t)$ and the motor constraints $h_i(c,p,t)$ differ in the computational effort necessary for evaluating the corresponding inequalities as well as in the information they provide. The collective constraint $h(c,p,t)$ can be explicitly stated, see (17), is easy to evaluate, and provides quick insight into the dynamic behavior of the quadrocopter by

excluding the majority of infeasible parameter sets. In contrast, for the motor constraints, we first need to calculate the nominal inputs $\omega_{x,d}(t)$ and $\omega_{y,d}(t)$. An explicit equation for $\omega_{x,d}(t)$ and $\omega_{y,d}(t)$ can be derived only in simple cases; in the general case, the rotational inputs are found numerically. The effects of both types of constraints are evident in the following two examples.

## 5. Examples

We consider two simple periodic motions that fall into the framework introduced in (2): a side-to-side motion and a circular motion in the horizontal plane. For the side-to-side motion, experimental results are shown in Sec. 6.

### 5.1 Side-to-Side Motion

The desired motion is a planar side-to-side movement,

$$s_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ z_d(t) \end{bmatrix} = \begin{bmatrix} A\cos(\Omega t) \\ 0 \\ 0 \end{bmatrix}. \tag{18}$$

The objective is to determine feasible combinations of amplitudes $A$ and frequencies $\Omega$. The side-to-side motion is a special case of the general motion primitive description (2), where $N = 1$, $a_1 = (A,0,0)$ and $a_0, b_1 = (0,0,0)$.

Calculating the second derivative of (18) and inserting it into (17), gives us the inequalities resulting from the collective thrust limit,

$$h(p,c,t) = \begin{bmatrix} \sqrt{A^2\Omega^4\cos^2\Omega t + g^2} - f_{max} \\ f_{min} - \sqrt{A^2\Omega^4\cos^2\Omega t + g^2} \end{bmatrix} \leq 0. \tag{19}$$

Given a pair $(A,\Omega)$, these inequalities must be satisfied for all $t \in [0,T]$. Therefore, it is enough to consider

$$\max_{t\in[0,T]} h(p,c,t) \leq 0, \tag{20}$$

which in the above case is simply

$$A\Omega^2 \leq \sqrt{f_{max}^2 - g^2}. \tag{21}$$

The second inequality is $f_{min} \leq g$ and must be satisfied in order for a quadrocopter to land. In brief, all parameter pairs $(A,\Omega)$ satisfying the inequality (21) represent side-to-side motions that stay within the collective thrust limits (12).

However, to guarantee feasibility of the trajectory, the required motor forces must satisfy (11). In order to evaluate the motor constraints $h_i(p,c,t),\ i \in \mathscr{I}$, we solve the dynamic equations. Note that in the following calculations, the subscript $(\cdot)_d$ is dropped to simplify notation.

For the side-to-side motion, the roll angle is zero, $\gamma(t) = 0$, for all $t \in [0, T]$. In order to fully determine the trajectory (cf. comments in Sec. 4.2), we set the rotational rate $w_z(t)$ to zero, resulting in $\dot{\alpha}(t) = 0$, see (8). The initial yaw angle is set to zero and, thus, $\alpha(t) = 0$ for all $t \in [0, T]$. With this, the translational dynamics (6) reduce to

$$\ddot{x} = f \sin\beta \tag{22}$$

$$\ddot{z} = f \cos\beta - g\,. \tag{23}$$

Recalling the above results, the nominal rotational inputs (8) are $\omega = [0, \dot{\beta}, 0]$. The rotational rate $\omega_y(t)$ and the fourth input, the collective thrust $f$, are obtained from (22), (23) and (18), where the latter implies $\ddot{z} = 0$. From (23),

$$f = \frac{g}{\cos\beta}\,, \tag{24}$$

and with (22) and the second derivative of $x_d(t)$,

$$\ddot{x} = g\tan\beta \quad \Leftrightarrow \quad \beta(t) = \tan^{-1}\left(-\frac{A\Omega^2}{g}\cos(\Omega t)\right). \tag{25}$$

Equations (24) and (25) yield the inputs $\omega_y(t) = \ddot{\beta}$ and $f(t)$. Once the nominal inputs are determined, the nominal motor forces are a direct consequence of (7) and (10),

$$\begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \frac{1}{ml} \begin{bmatrix} 0 \\ 0 \\ I_y\ddot{\beta} \\ fl \end{bmatrix}, \tag{26}$$

where the right vector is obtained from the previous analysis and the matrix is invertible. Solving this linear system of equations results in

$$f_1 = \frac{1}{2}\left(\frac{f}{2} + \frac{I_y}{ml}\ddot{\beta}\right), \quad f_3 = \frac{1}{2}\left(\frac{f}{2} - \frac{I_y}{ml}\ddot{\beta}\right), \quad f_2 = f_4 = f/4\,. \tag{27}$$

The collective thrust constraint (21) guarantees the feasibility of $f_2$ and $f_4$, while the motor constraints (11) of $f_1$ and $f_3$ narrow down the set of feasible pairs $(A, \Omega)$ compared to the collective thrust inequality (21).

For the vehicle parameters in Tab. 1, Fig. 4 illustrates the feasible set of side-to-side trajectories $(A, \Omega)$ for both cases. The dark gray region contains parameter sets that are infeasible due to the collective thrust limit, cf. (21). The light gray area represents the additional infeasible parameter sets obtained by taking into account the limits on the motors. Matlab files for creating the plots are available at `www.idsc.ethz.ch/Downloads/QuadDance`.

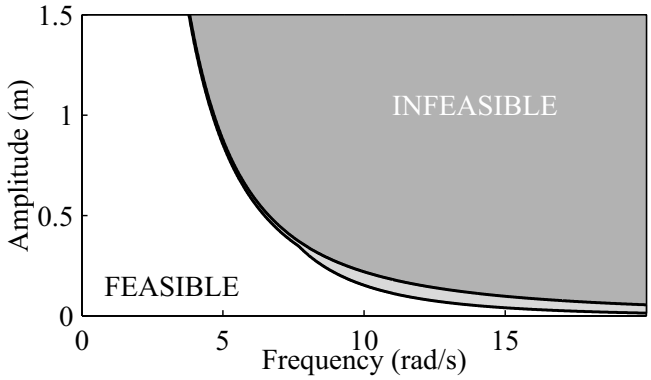



**Figure 4.** Feasible parameter sets for the side-to-side motion primitive. The dark gray region denotes parameter sets that are infeasible due to collective thrust limits; light gray denotes sets that are infeasible due to additional single motor constraints.

## 5.2 Circular Motion

As a second periodic motion primitive, we require a quadrocopter to fly a circle in the horizontal plane at a constant rotational rate $\Omega$ with radius $A$,

$$s_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ z_d(t) \end{bmatrix} = \begin{bmatrix} A\cos(\Omega t) \\ A\sin(\Omega t) \\ 0 \end{bmatrix}. \tag{28}$$

The circle is represented by the general motion primitive description (2) with $N = 1$, $a_0 = (0,0,0)$, $a_1 = (A,0,0)$ and $b_1 = (0,A,0)$. We study the feasibility of the circle primitive depending on the parameters $(A,\Omega)$ and follow the same procedure as for the side-to-side motion in Sec. 5.1.

First, the collective thrust constraint (17) is evaluated. For the circle, the nominal collective thrust is constant, cf. (16),

$$f_d = \sqrt{A^2\Omega^4 + g^2}, \tag{29}$$

resulting in the inequalities

$$A\Omega^2 \leq \sqrt{f_{max}^2 - g^2}. \tag{30}$$

and $f_{min}^2 - g^2 \leq A^2\Omega^4$. The latter is true for $f_{min} < g$, see Sec. 5.1. Note that the same inequality, cf. (21), describes the collective thrust limit of the side-to-side primitive.

Second, we study the feasibility with respect to the motor force limits (11). For deriving the nominal rotational inputs, we transform the equations of motion into different coordinate systems, such that the flight dynamics can be described in a time-invariant manner. The subscript $(\cdot)_d$ is omitted in order to simplify notation. To describe the vehicle position, we introduce the following coordinate system **C** with $(u,v,w)$ describing the quadrotor position in **C**:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} := R_z(\Omega t) \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \cos\Omega t & -\sin\Omega t & 0 \\ \sin\Omega t & \cos\Omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{31}$$

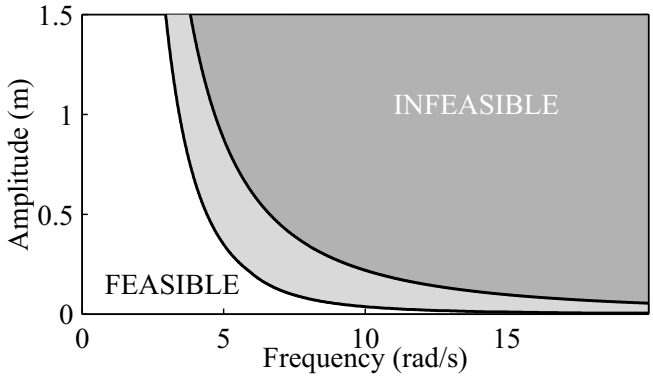



**Figure 5.** Feasible parameter sets for the circular motion primitive. The dark gray region denotes parameter sets that are infeasible due to collective thrust limits; light gray denotes sets that are infeasible due to additional single motor constraints.

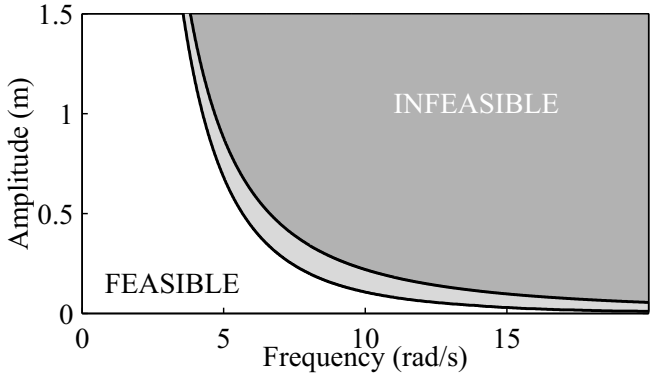



**Figure 6.** Feasible parameter sets for the circular motion primitive when yaw control needs no additional control effort ($k = \infty$). The dark gray region denotes parameter sets that are infeasible due to collective thrust limits; light gray denotes sets that are infeasible due to single motor constraints.

The attitude of the vehicle is represented by a second set of Euler angles $(\eta, \mu, \nu)$, describing the 'virtual vehicle attitude' **W**:

$$ {}^{O}_{W}R(\eta,\mu,\nu) = R_z(\eta)\,R_y(\mu)\,R_x(\nu)\,, \tag{32} $$

where

$$ {}^{O}_{V}R(\alpha,\beta,\gamma)\begin{bmatrix}0\\0\\1\end{bmatrix} = {}^{O}_{W}R(\eta,\mu,\nu)\begin{bmatrix}0\\0\\1\end{bmatrix}. \tag{33} $$

As every column of a rotation matrix has a unit norm, this equation defines only two of the angles $(\eta, \mu, \nu)$. With (31), the derivatives of (31) and (33), the quadrotor's equations of motion

(6) simplify to

$$\begin{bmatrix} \ddot{u} \\ \ddot{v} \\ \ddot{w} \end{bmatrix} = \begin{bmatrix} f \sin\mu \cos\nu + \Omega^2 u + 2\Omega\dot{v} \\ -f \sin\nu - 2\Omega\dot{u} + \Omega^2 v \\ f \cos\mu \cos\nu - \mathrm{g} \end{bmatrix}, \tag{34}$$

when setting the free parameter $\eta$ to $\eta = \Omega t$. The circular trajectory is described by $u = A$ and $v = 0$. Again, we have an additional design parameter to choose, see Sec. 4.3. For the circle, the vehicle rotation around its vertical axis is set to zero, i.e. $\dot{w} = 0$. Using these values and the nominal thrust (29), the Euler angles $\mu$ and $\nu$ can be calculated from (34):

$$\mu = \arctan\left(-\frac{A\Omega^2}{\mathrm{g}}\right), \quad \nu = 0. \tag{35}$$

Knowing the values for $(\eta, \mu, \nu)$, we solve for $(\alpha, \beta, \gamma)$ using (33). We choose $\alpha = 0$, simplifying (33) to

$$\begin{bmatrix} \sin\beta\cos\gamma \\ -\sin\gamma \\ \cos\beta\cos\gamma \end{bmatrix} = \begin{bmatrix} \cos\Omega t \sin\mu\cos\nu + \sin\Omega t \sin\nu \\ \sin\Omega t \sin\mu\cos\nu - \cos\Omega t \sin\nu \\ \cos\mu\cos\nu \end{bmatrix}, \tag{36}$$

which can be solved for $\beta$ and $\gamma$. To calculate the rotational rate inputs in (8), we take the first derivative of (36). It can be shown that

$$\dot{\beta} = \frac{A\Omega^3 \cos^{-1}\gamma (\tan\beta \tan\gamma \cos(\Omega t) + \cos^{-1}\beta \sin(\Omega t))}{\sqrt{\mathrm{g}^2 + A^2\Omega^4}} \tag{37}$$

$$\dot{\gamma} = \frac{A\Omega^3 \cos^{-1}\gamma \cos(\Omega t)}{\sqrt{\mathrm{g}^2 + A^2\Omega^4}}. \tag{38}$$

Combining this result with the results from (8) and (10), one can solve for the nominal control inputs $(\omega_x, \omega_y, \omega_z)$, similar to the side-to-side motion in the previous section, Sec. 5.1. The equations for the motor forces are not explicitly stated here, however Matlab files showing the relevant equations and creating the corresponding plots, see Fig. 5 and 6, are available at `www.idsc.ethz.ch/Downloads/QuadDance`.

Fig. 5 illustrates the feasible set of circle trajectories $(A, \Omega)$ for the vehicle parameters in Tab. 1. The collective thrust limit, cf. (30), is identical to the side-to-side motion. However, the boundary that takes into account the single rotor limits is lower. One reason is that, for the circle motion, additional rotor force is needed to keep the yaw angle at zero. Choosing the motor constant $k = \infty$, meaning that no force is required for rotational accelerations around the vertical axis of the vehicle, cf. (10), the feasible set of parameters increases, see Fig. 6. In other words, the control effort for yaw (for the given quadrocopter, see Tab. 1) is large and has a noticeable effect when studying the feasibility of trajectories.

## 6. Preliminary Experimental Results

We now compare the predicted feasible region of the side-to-side motion with experimental results. Experiments were conducted in the ETH Flying Machine Arena on our customized quadrocopters. The Flying Machine Arena is an indoor research space built specifically for the study of autonomous systems and aerial robotics. Details on the testbed, the vehicles, and the communication and control infrastructure are found in [15, 20].

The side-to-side motion was performed for various frequencies $\Omega$. The amplitude was increased in small steps of 1 to 2cm starting from 0m. We monitored the commands to the motors and determined the percentage of saturated motor commands per period, hitting either the lower or upper limit of the motor, $f_{i,min}$ or $f_{i,max}$, respectively. Fig. 7 shows the experimentally obtained feasibility limits with the corresponding predicted feasibility bounds, calculated as above with the vehicle parameters in Tab. 1. The vehicle parameters of our quadrocopter were determined experimentally and used before in [20, 21].

The feasibility bounds found experimentally support the predicted parameter limits. In our experiments, saturation occurs earlier than predicted. This can be explained by the fact that a simplified model was used when deriving the analytical bounds, see Sec. 3.1. Motor dynamics, effects caused by sampling of the inputs, slew rate limits on the motor commands etc. are not considered in our analytic derivations. Moreover, additional thrust is required to stabilize the vehicle.

## 7. Conclusions

In this paper we studied the feasibility of quadrocopter motions based on first principles models of the vehicle dynamics. We derived equations that ensure the feasibility of a desired trajectory (assumed to be twice-differentiable in time) with regard to the vehicle's collective thrust limits and the motor thrust limits. In particular, we considered motion primitives that are adjustable in their parameters. Such parameterized motion primitives will form the basis for a choreographed

**Table 1.** Vehicle Parameter

| | Definition | Value |
|---|---|---|
| $m$ | mass of vehicle | 0.468 kg |
| $L$ | vehicle arm length | 0.17 m |
| $I_x$ | inertia around vehicle ${}^V x$-axis | 0.0023 kg m$^2$ |
| $I_y$ | inertia around vehicle ${}^V y$-axis | 0.0023 kg m$^2$ |
| $I_z$ | inertia around vehicle ${}^V z$-axis | 0.0046 kg m$^2$ |
| $k$ | motor constant | 0.016 m |
| $f_{i,min}$ | normalized min. rotor force | 0.17 m/s$^2$ |
| $f_{i,max}$ | normalized max. rotor force | 6.0 m/s$^2$ |

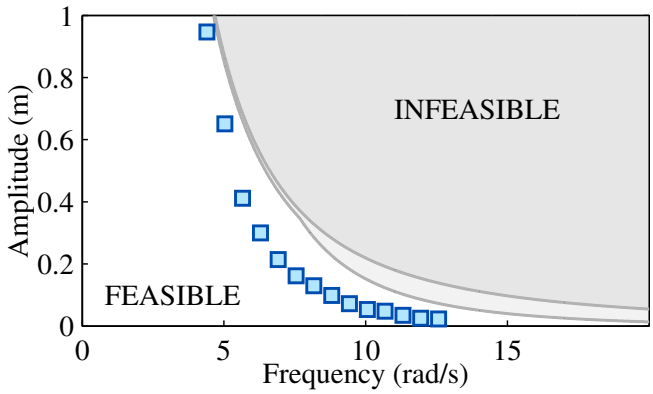



**Figure 7.** Experimentally determined feasibility limits for the side-to-side motion. The blue boxes mark amplitudes where motor commands are saturated 1% of the time. The predicted feasibility regions are shown in gray.

flight performance with quadrocopters. By adjusting the motion parameters, these primitives can capture fast/slow, smooth/sharp, and big/small motions. The goal of the feasibility analysis was to identify feasible parameter sets for these parameterized motion primitives a priori to flight experiments.

The first feasibility test used the collective thrust limit to effectively exclude most infeasible parameter combinations at little computational cost. A second feasibility test considered the thrust limits of each motor to obtain a more realistic approximation of the feasible set of trajectories. For determining the feasibility with respect to the single motors, the quadrocopter's dynamic equations must be solved for the corresponding nominal inputs. This was done for two simple examples, and the feasible parameter sets obtained from the first and second approach were compared. Our experiments validated the predicted feasibility bounds.

Ultimately, a library of these adjustable motion elements – together with their associated sets of feasible parameters – will serve as a basis for building a multifaceted choreography that is able to express different shapes, with different rhythms, in different spatial dimensions. First steps towards performing these choreographies in synchrony with music are shown in [15, 22] and accompanying videos.

## References

[1] B. Williams, M. Toussaint, and A. Storkey, "Extracting motion primitives from natural handwriting data," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2006, pp. 634–643.

[2] L. Reng, T. B. Moeslund, and E. Granum, *Finding Motion Primitives in Human Body Gestures*. Springer, 2006, pp. 133–144.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[4] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[5] E. Frazzoli, "Maneuver-based motion planning and coordination for multiple UAVs," in *Proceedings of the Digital Avionics Systems Conference*, vol. 2, 2002, pp. 8D3–1 – 8D3–12.

[6] D. Kulic, D. Lee, C. Ott, and Y. Nakamura, "Incremental learning of full body motion primitives for humanoid robots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 326–332.

[7] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 286–98, 2007.

[8] B. Hemes, D. Fehr, and N. Papanikolopoulos, "Motion primitives for a tumbling robot," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1471–1476, 2008.

[9] R. Cutler and L. Davis, "Robust real-time periodic motion detection, analysis, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 781–796, 2000.

[10] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGRAPH)*, 2004, pp. 559–568.

[11] R. Groten, J. Hoelldampf, M. Di Luca, M. Ernst, and M. Buss, "Motion Primitives of Dancing," *Journal of Neuroscience*, pp. 838–843, 2008.

[12] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Detecting dance motion structure through music analysis," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 857–862.

[13] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi, "Leg motion primitives for a dancing humanoid robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 610–615.

[14] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Rhythmic motion analysis using motion capture and musical information," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2003, pp. 89–94.

[15] A. Schöllig, F. Augugliaro, and R. D'Andrea, "A platform for dance performances with multiple quadrocopters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) - Workshop on Robots and Musical Expressions*, 2010, pp. 1–8.

[16] P. Sofras, *Dance composition basics: capturing the choreographer's craft*. Human Kinetics, 2006.

[17] S. C. Minton, *Choreography: a basic approach using improvisation*, 3rd ed. Human Kinetics, 2007.

[18] G. P. Tolstov and R. A. Silverman, *Fourier series*. Courier Dover Publications, 1962.

[19] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro UAV testbed," *IEEE Robotics and Automation Magazine*, 2010.

[20] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[21] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, to appear.

[22] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3355–3360.

# Curriculum Vitae

**Angela P. Schoellig**

born July 17th, 1983 in Miltenberg, Germany

| | |
|---|---|
| 2008 – 2012 | *ETH Zurich, Switzerland*<br>Doctoral studies in the group of Prof. Raffaello D'Andrea at the Institute for Dynamic Systems and Control, Department of Mechanical and Process Engineering; graduated with Dr. sc. ETH Zurich. |
| 2002 – 2008 | *University of Stuttgart, Germany*<br>Undergraduate and graduate studies; graduated with Dipl.-Ing. (equivalent to M.Sc.) in Engineering Cybernetics. |
| 2007 | *EADS Astrium GmbH (European Aeronautic Defense and Space Company), Friedrichshafen, Germany*<br>Internship in the group "Future Programmes & Missions, Science Missions & Systems." |
| 2006 – 2007 | *Georgia Institute of Technology, Atlanta, USA*<br>Graduate studies; graduated with M.Sc. in Engineering Science and Mechanics. |
| 2002 | *Dürr Systems GmbH, Bietigheim-Bissingen, Germany*<br>Internship in the group working on optimizing the use of robots for the serial painting of car and airplane bodies. |
| 1993 – 2002 | *Max-Born Gymnasium Backnang, Germany*<br>High-school education; graduated with Abitur (high school diploma for admission to higher education). |