DISS. ETH NO 29539

OPTIMIZATION-BASED TRAJECTORY PLANNING FOR PRECISION MOTION SYSTEMS AND AUTONOMOUS ROBOTIC INSPECTION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zürich)

presented by

ANTÓNIO SAMUEL ÁVILA BALULA

M.Sc., Instituto Superior Técnico,
University of Lisbon

born on 19.10.1993

accepted on the recommendation of

Prof. Dr. John Lygeros, examiner

Prof. Dr. Kira Barton, co-examiner

Prof. Dr. Dominic Liao-McPherson, co-examiner

2023

For those who never cease to be curious.

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. John Lygeros and Dr. Alisa Rupenyan for their guidance and support over the course of my doctoral studies. I am also deeply grateful to Prof. Dominic Liao-Mc Pherson, to Dr. Alex Liniger, and to Dr. Efe Balta who have contributed extensively with their valuable insights and guidance. I would also like to thank Prof. Kira Barton, for accepting to be part of the examination committee. This work benefited from the productive collaboration with Dr. Stefan Stevšić, and Dr. Christoph Gebhardt. Besides their scientific contributions, I would like to thank their invaluable support regarding the experimental setup used in Chapter 4. I would also like to acknowledge Dr. Natanael Lanz for his support with the *Andromeda* experimental setup, used in Chapters 2 and 3. I thank the institutional and financial support of the Automatic Control Laboratory of ETH Zürich, Inspire AG, Tinamu Labs, InnoSuisse, and NCCR Automation.

I would like to thank all of the students I supervised during their group, semester or master thesis, for their commitment, creativity, and for all that I have learned with and from them in so many different topics. This was an extraordinarily enriching experience: Alexander Almeida, Andreas Schlaginhaufen, Danila Merola, Divya Guruswamy, Eugenio Chisari, Gianni Pasini, Ivan Gehri, Jacob Clarysse, Jente Clarysse, Jonas Holzem, Kathrin Schleicher, Luzian Hug, Martin Baumann, Martin Vahlensieck Michel Schubiger, Silvia Ritsch, Tim Liechti, Timo Laaksonlaita, Yingjie Jiang, and Zheng Sheng. A special word goes to the colleagues with whom I had the opportunity of co-supervising these students with: Dr. Alex Liniger, Prof. Andrea Iannelli, Anil Parsi, Aren Karapetyan, Dr. Christoph Gebhardt. Prof. Dominic Liao-Mc Pherson, Dr. Eva Ahbe, Dr. Goran Banjac, Dr. Joe Warrington, Sandeep Menta, Dr. Stefan Stevšić, and Xavier Guidetti. I would also like to thank Sabrina Baumann and Tanja Turner, who keep the lab running so smoothly. A word of thanks to my colleagues, who have kindly accepted to proofread this thesis, Verena Häberle, Dr. Marta Zagorowska, Marta Fochesato, and Anil Parsi. To my office mates, who have been true companions in this journey, Dr. Yvonne Stürz, Dr. Jianzhe (Trevor) Zhen, and Dr. Efe Balta.

To my colleagues, with whom I had the privilege to work with at the Automatic

Control Laboratory (IfA). You made, undoubtedly, the best of my experience during my doctoral studies. Many of you have also become close friends. I am deeply indebted to you, both scientifically and personally. You have shown me how to see the world from new perspectives, helped me in times of struggle and shared the joy in times of celebration. I could not have imagined better colleagues and friends. A special word goes to Dr. Ahmed Aboudonia, Andrea Martinelli, Anil Parsi, Aren Karapetyan, Marta Fochesato, and Sandeep Menta. I lack the eloquence to thank you properly.

To my friends, who have made me call Switzerland home, and with whom I shared so many adventures with, over the last five years. A special word goes to Dr. Ricardo Peres, Dr. André Coroado, Dr. Raquel Calçada, Catarina Lozano, Dr. Rita Afonso, Dr. Raúl Penaguião, Leonor Lopes, Pragnya Sharma, Marta Ostini, and António Coelho as well as to my old friends, who I consider like family: Manuel Barjona, Telmo Pires, João Duarte, Rita Tomaz, and Ricardo Meyrelles. I am profoundly grateful for your friendship.

Lastly, and most importantly, I am deeply grateful to my parents, António and Gilda Balula. You instilled in me the importance of perseverance, the value of knowledge, and the courage to pursue my dreams. You believed in me, even when I doubted myself. Thank for contributing more than anyone else to make me who I am today.

Abstract

In industrial environments economic savings are a primary driver for constant quality and productivity improvements. For motion systems in particular, improvements can be achieved with a better choice of motion trajectories. This thesis proposes a set of methods to plan trajectories with applications in precision motion systems and autonomous robotic inspection.

First, we propose an optimization-based pre-compensation method to improve the contour tracking performance of precision motion systems. The approach modifies the reference trajectory, while leaving unaltered the built-in low-level controller. The position of the precision motion system is simulated with two data-driven models of different fidelity. A linear low-fidelity model is employed to optimize path traversal time, by manipulating the path velocity and acceleration profiles. Next, a non-linear high-fidelity model is used to refine the previously computed time-optimal solution. Second, we propose an algorithm for the nonlinear iterative learning control problem based on sequential quadratic programming. We iteratively solve quadratic subproblems built by combining approximate nonlinear models and process measurements to find an optimal input for the original system. We demonstrate our method in a trajectory optimization problem for a precision motion system. We present simulations and experimental results to validate the performance of the proposed method, comparing the achieved performance for linear and nonlinear gradient models. We demonstrate experimentally that both methods are capable of simultaneously improving the productivity and the accuracy of a precision motion system. Given the data-based nature of the models, they can easily be adapted to a wide family of precision motion systems.

We then address the problem of planning trajectories for autonomous robotic inspection. Volume estimation in large indoor spaces is an important challenge in robotic inspection of industrial facilities. We propose an approach for volume estimation for autonomous systems using visual features for indoor localization and surface reconstruction from 2D-LiDAR measurements. A Gaussian Process-based model incorporates information collected from measurements given statistical prior information about the surface. The volume is measured from the surface model reconstruction. Our algorithm finds feasible motion trajectories for quadcopters which minimize the uncertainty of the volume estimate. We show simulation and experimental results for the surface reconstruction of topographic and industrial data.

Zusammenfassung

In den heutigen industriellen Umgebungen besteht zunehmend mehr wirtschaftlicher Druck, die Qualität und Produktivität von Produkten und/oder Prozessen zu maximieren. Insbesondere in Bewegungssystemen können Verbesserungen durch eine bessere Auswahl von Trajektorien erreicht werden. Diese Dissertation stellt eine Reihe von Methoden zur Planung von Trajektorien vor, welche für Anwendungen in Präzisionsbewegungssystemen und autonomer robotergesteuerter Inspektion ausgelegt sind.

Zunächst wird eine optimierungsbasierte Vor-Kompensationsmethode präsentiert, um die Konturverfolgungsleistung von Präzisionsbewegungssystemen zu verbessern. Hierbei wird die Referenztrajektorie modifiziert, während der eingebaute Low-Level-Controller unverändert bleibt. Die Position der Präzisionsbewegungsplattform wird mit zwei datengetriebenen Modellen unterschiedlicher Genauigkeit simuliert. Als erstes wird ein lineares Low-Fidelity-Modell verwendet, um die Pfad-Durchlaufzeit zu optimieren, indem die Pfadgeschwindigkeits- und Beschleunigungsprofile manipuliert werden. Im Anschluss wird ein nicht-lineares High-Fidelity-Modell eingesetzt, um die zuvor berechnete zeitoptimale Lösung weiter zu verfeinern. Experimente bestätigen, dass die vorgeschlagene Methode in der Lage ist, die Produktivität und die Genauigkeit einer hochpräzisen Bewegungsstufe gleichzeitig zu verbessern. Als ein zweites Resultat dieser Arbeit wird ein Algorithmus für das nichtlineare iterative Lernkontrollproblem vorgestellt. Dieser basiert auf sequentieller quadratischer Programmierung, einer weit gebräuchlichen und anerkannten Methode für nichtkonvexe Optimierung. Es werden wiederholt quadratische Unterprobleme gelöst, die mit Hilfe von annähernden nichtlinearen Modellen und Prozessmessungen erstellt werden, sodass ein optimaler Eingang für das ursprüngliche System gefunden werden kann. Die neue Methode wird in einem Trajektorienoptimierungsproblem für ein Präzisionsbewegungssystem demonstriert. Simulations- und Experimentalergebnisse validieren die hohe Leistungsfähigkeit der Methode, wobei sowohl das lineare als auch das nichtlineare Gradientenmodell verglichen wird. Aufgrund der datenbasierten Natur beider Modelle können die entwickelten Methoden auch auf andere Präzisionsbewegungssysteme angewandt werden.

Der letzte Teil dieser Arbeit befasst sich mit der Planung von Trajektorien für die autonome robotergesteuerte Inspektion. Heutzutage stellt insbesondere die Volumenschätzung in grossen Innenräumen eine wichtige Herausforderung bei der robotergesteuerten Inspektion von Industrieanlagen dar. In dieser Arbeit wird daher ein neuer Ansatz zur Volumenschätzung für autonome Systeme präsentiert, der visuelle Merkmale zur Innenraumlokalisierung und zur Oberflächenrekonstruktion aus 2D-LiDAR-Messungen verwendet. Ein auf dem Gaussschen Prozess basierendes Modell beinhaltet Informationen, die aus Messungen unter Berücksichtigung statistischer Vorinformationen über das Gelände gesammelt wurden. Das Volumen wird aus der Rekonstruktion des Oberflächenmodells gemessen. Der vorgeschlagene Algorithmus findet machbare Trajektorien, die die Unsicherheit der Volumenschätzung minimieren. Die Oberflächenrekonstruktion aus topographischen und industriellen Daten wird simulativ und experiementel validiert.

Contents

List of Figures

List of Tables

Glossary

ANN	Artificial Neural Network
CNC	Computer Numeric Control
GP	Gaussian Process
ILC	Iterative Learning Control
LiDaR	Light Detection and Ranging
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
OB-ILC	Optimization-Based Iterative Learning Control
PID	Proportional Integral Derivative [Controller]
PMS	Precision Motion System
QP	Quadratic Program
SfM	Structure from Motion
SQP	Sequential Quadratic Programming

CHAPTER 1

Introduction

The aim of this thesis is to design trajectories for complex nonlinear systems that are required to operate at the limits of their capabilities, while executing tasks of industrial relevance in real-world scenarios.

This Chapter is organized as follows. In Section 1.1 we present the motivation of this work, followed in Section 1.2 by the objectives and in Section 1.3 by the contributions. In Section 1.4 we list the publications in which this thesis is based, and in Section 1.5 we list other work that resulted from the supervision of students, during the course of the doctoral studies. Finally, in 1.6 we present an outline of the rest of this thesis.

1.1 Motivation

We are surrounded by dynamical systems of different scales and complexity, both natural and engineered, that play a crucial role in our daily lives. An important subset of these systems can be instrumented and controlled to function in desirable ways. This work focuses on controlling such systems, taking into account their dynamics and limitations, and improving their efficiency. Industrial processes, in particular, stand to benefit from these improvements, as even small gains in productivity are multiplied in scale and can have a large impact. Potential benefits include reduced environmental footprint of the produced goods, as well as lower manufacturing cost.

The task of designing an effective controller often requires or benefits from access to a simplified mathematical representation of the complex system at hand. For example, an effective model allows the evaluation of different control schemes in a simulated environment. Furthermore, it can be used directly in the context of optimization, such as with a Linear Quadratic Regulator (LQR) or Model Predictive Control (MPC). Traditionally, different fields of science, for example physics, chemistry or biology, provide mathematical models that describe the system's behavior. First-principles modeling relies on expert understanding of the fundamental scientific and engineering principles, generalizes well and requires little or no experimental data. However, models based on first principles analysis are, in some cases, either cumbersome, time-consuming or impossible to derive. In those cases data-driven models, built from experimental data, are advantageous. In fact, with the widespread availability of sensors and connected embedded computing devices, it is now feasible to record large amounts of data about a variety of systems, enabling high-fidelity data-driven models to be built.

In this thesis, we consider two classes of problems where such a data driven approach offers such benefits. The first are Precision Motion System (PMS). These are essential in many industrial applications, including Computer Numeric Control (CNC), manufacturing of semiconductors, precision machining of parts, and inspection. Trajectory planning is a critical component for PMS, enabling efficient high accuracy motion. For PMS, the precision requirements of the task mean stringent requirements of the model prediction accuracy if it is to be used in control. Data-driven modelling can account for complex system dynamics, capturing details that could otherwise be neglected by experts, and delivers high accuracy. Furthermore it offers adaptability to machine modifications, thus reducing the need for continuous expert input and intervention.

The second class of problems we consider is Autonomous robotic inspection. This is an area of growing importance, particularly in industrial settings. It allows for the evaluation of assets in complex and hazardous environments, mitigating the risks posed to humans. Due to limits on resources, notably in battery life, effective planning of the inspection task is important to guarantee successful completion of the task. We focus in surface reconstruction and volume estimation with a quadcopter platform. Here we leverage statistical information about the surface properties, representing it with a efficient Gaussian Process (GP) model, and combining it with sensor data.

In summary, this thesis is motivated by the high potential that automation and control has for a more efficient society through advancements in industrial processes, and enabled by the growing availability of data and computational resources.

1.2 Objectives

The main objective of this thesis is to propose and validate practical algorithms for the control of nonlinear systems, more specifically tailored for PMS and Autonomous Robotic Inspection using quadcopters. The algorithms presented in this thesis are validated both with high-fidelity simulation and empirically.

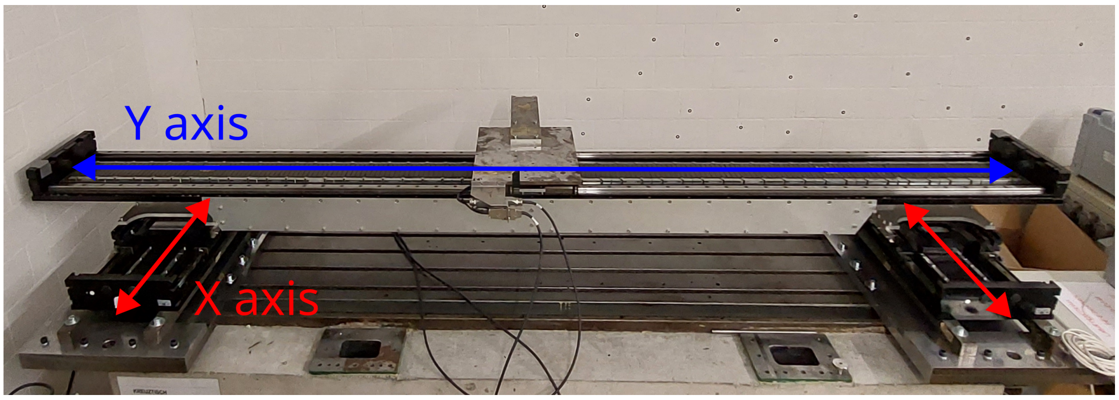
Trajectory planning for PMS

A PMS is a robotic setup designed to precisely position an end-effector, typically in the micro- to nanometer range. PMSs are used in lithography [1], micro-assembly [2], precision metrology [3], and precision engineering [4], and many other applications [5, 6]. They are often used as components in machine tools for the production of parts, such as metal laser cutting or grinding processes where the goal is to trace a desired target geometry as quickly and accurately as possible.

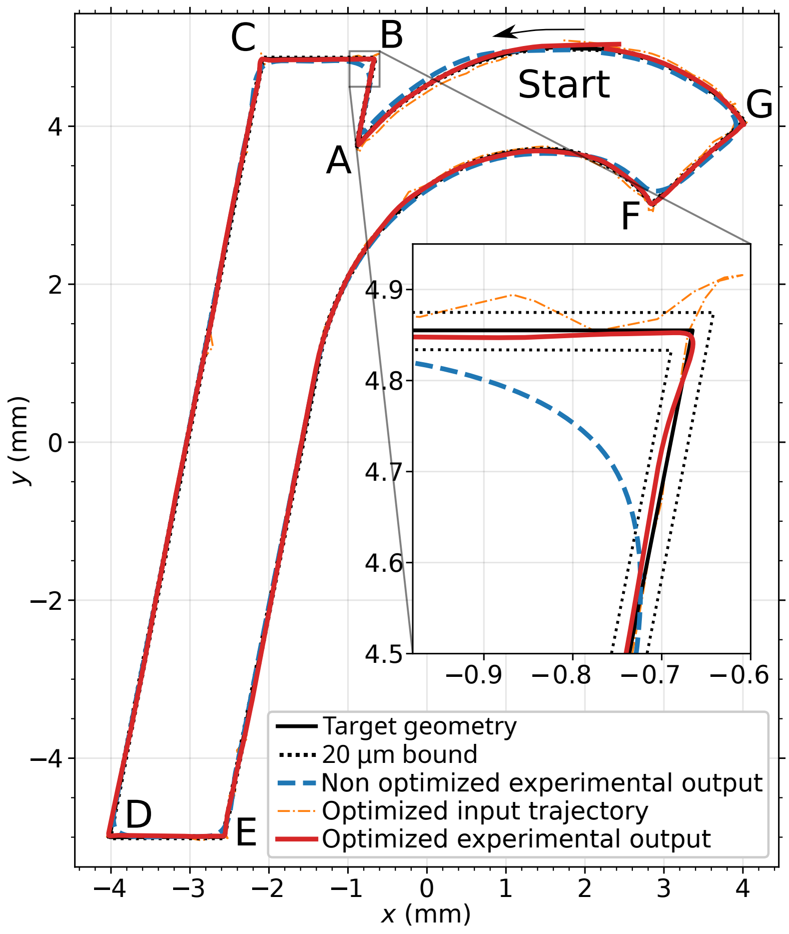
The accuracy of PMSs is critical as the parts they produce must satisfy tight engineering tolerances. At the same time, economic considerations dictate rapid production of parts to boost productivity. The physical inertia of PMSs dictates a trade-off between productivity and accuracy. In practice, the quality of the control system is crucial for managing this trade-off. Typical industrial precision systems achieve high-precision using well-tuned “classical” feedback controllers, such as Proportional Integral Derivative [Controller] (PID), to track a given target trajectory. These low-level controllers are designed by the manufacturer and users are generally not able to modify or tune the control algorithm.

Our goal is to improve the precision and productivity of a PMS, without any changes to the existing controller structure. We aim to achieve such improvements relying exclusively on experimental data, or data-based models, thus avoiding the need for a expert-designed first-principles model. The trajectories should respect system and engineering constraints, as well as system dynamics.

Experimental setup Figure 1.1a shows a picture of the experimental setup used as a test bed in Chapters 2 and 3. It consists of an `ETEL DXL-LM325` two stage motion machine. The machine is instrumented with quadrature encoders that measure the position of both axes and is actuated with `ETEL LMS15-100` linear motors. The system is driven by a feedback controller that adjusts the motor voltages to track a supplied input trajectory r. The controller is implemented on a `Speedgoat` rapid prototyping unit using `MATLAB/Simulink` software. We consider this loop as a black-box and the 2D input trajectory r as the only input. The machine enforces limits on the acceleration, velocity, and position of the input



(a) Experimental apparatus.



(b) Example trajectory.

Figure 1.1: Panel (a) shows the 2D PMS used as a test case in this thesis. Experimental data from this experimental setup is shown in panel (b), where the reference trajectory is optimized such that the output tracks a target shape more precisely than the existing controller.

Property	Value	Unit
Absolute Maximum acceleration	40	$\mathrm{m\,s^{-2}}$
Recommended Maximum acceleration	3	$\mathrm{m\,s^{-2}}$
Recommended Maximum velocity	1.5	$\mathrm{m\,s^{-1}}$
Low-level control loop frequency	10	kHz
Repeatability (standard deviation)	2	μm

Table 1.1: Software-limited values and operational specifications of the experimental setup.

trajectory to prevent accidental damage. The main characteristics of the machine are collected in Table 1.1.

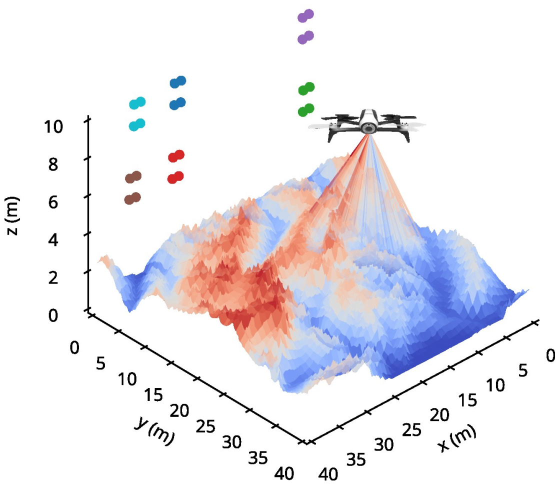
A highlight of this setup is that for the same provided input trajectory, the distribution of the outputs over several repetitions exhibits a standard deviation of approximately $2\,\mu$m. This fact allows us to design input trajectories offline. Figure 1.1b shows an example of an optimized trajectory.

Trajectory planning for Autonomous Robotic Inspection

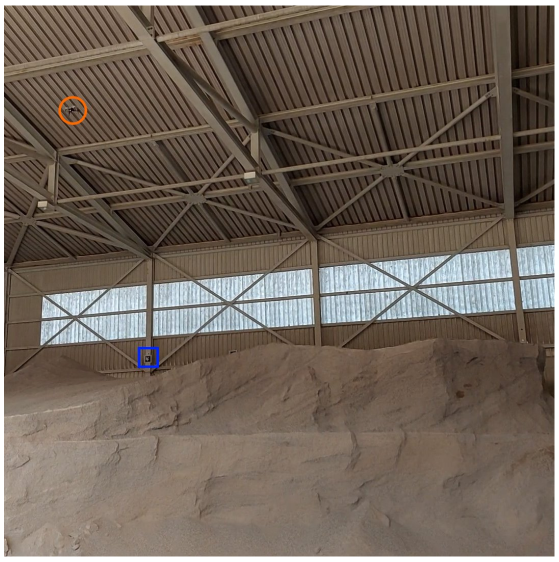
The process of inspection is a complex and often critical task that traditionally required direct human intervention. Employing autonomous robotic platforms can enhance the safety, accuracy and quality of the inspection process, with the potential to reduce cost. Quadcopters in particular can access difficult to reach areas, carrying sensors adapted to the task at hand.

In the second part of the thesis, we consider the problem of designing feasible trajectories that maximize the amount of information gathered with a Light Detection and Ranging (LiDaR) scanner. The scanner is attached to a quadcopter, which is flying in a large indoor space in order to estimate the volume of bulk material, such as sand or ore, contained in it. In particular we are interested in estimating the volume of material, with the lowest possible uncertainty.

Experimental setup Figure 1.2 shows a picture of the experimental setup and simulation environment where a quadcopter flies in an indoor, GPS-deprived, environment. The drone is equipped with a 2D LiDaR scanner, obtaining distance measurements to the landscape below it. Localization is inferred from visual features, with known 3D coordinates.



(a) Simulation environment.



(b) Experimental environment.

Figure 1.2: A quadcopter flies above a landscape, collecting depth data with a LiDaR scanner and inferring localization from a previously mapped set of features. Panel (a): Simulated environment. The surface is a segment of a scaled topographic map of the Swiss Alps. The features used for localization are marked as dots located approximately on the $y = 0$ plane. Panel (b): A picture from an experimental flight. The drone is highlighted with an orange circle, and one of the features used for localization is highlighted with a blue square.

1.3 Contributions

The main contributions of this thesis are

1. A high-fidelity model of a 2D PMS, obtained exclusively from experimental data;
2. A method for reference trajectory design using exclusively data-driven models' information;
3. An Optimization-Based Iterative Learning Control (OB-ILC)-based method for reference trajectory design, mixing model gradient information and error correction from experimental data;
4. A modeling environment for drone flight in an indoor GPS-deprived setting, where localization is inferred from features and incorporating LiDaR scanner data;
5. An efficient method for surface reconstruction and volume computation from a point cloud obtained from LiDaR data.

1.4 Publications

The work presented in this thesis is based on the published and submitted articles listed below.

Chapter 2

[7] S. Balula, A. Liniger, A. Rupenyan, and J. Lygeros, “Reference design for closed loop system optimization,” in *2020 European Control Conference (ECC)*, pp. 650–655, IEEE, 2020

[8] S. Balula, D. Liao-McPherson, A. Rupenyan, and J. Lygeros, “Data-driven reference trajectory optimization for precision motion systems,” *Under submission*, 2023

Chapter 3

[9] S. Balula, E. Balta, D. Liao-McPherson, A. Rupenyan, and J. Lygeros, “Sequential quadratic programming-based iterative learning control for nonlinear systems,” in *2023 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2023

Chapter 4

[10] S. Balula, D. Liao-McPherson, S. Stevšić, A. Rupenyan, and J. Lygeros, “Drone-based volume estimation in indoor environments,” in *IFAC world congress*, 2023

1.5 Other work

This thesis does not address the following published and unpublished works, the result of Master and Bachelor thesis work by students I had the privilege of supervising.

Magnetic levitation: A control experiment for the General Laboratory course by Danila Merola. Supervised by Aren Karapetyan, Xavier Guidetti, Samuel Balula, Prof. John Lygeros.

A digital controller is developed for a Magnetic Levitation device from Quanser. The device is used to levitate a small steel ball with the magnetic field force generated by passing current through a coil. First the system is analyzed theoretically:

two subsystems are identified, the electrical and the electro-mechanical subsystem. The electrical subsystem generates a coil voltage depending on the desired coil current, while he electro-mechanical subsystem generates a desired coil current depending on the desired ball position. The modeling and the control design for both subsystems are presented. Second, a PI controller for the electrical subsystem and a PIV controller with Feed-Forward for the electro-mechanical subsystem are introduced. The control system is designed as a cascade control loop, where the electrical system represents the inner loop and the electro-mechanical system represents the outer loop. Simulations are carried out in the MATLAB-Simulink environment. As signal conditioning is needed to map the signals into the desired voltage ranges, circuits with inverting operational amplifiers are designed and implemented on a breadboard. A PCB for a future implementation is also presented. For the digital controller an Arduino Mega 2560 microcontroller is chosen. In conclusion a possible structure to implement this project as a control laboratory is presented.

Optimizing Quadrotor Trajectories for Robust Vision-based Flight by Zheng Shen. Supervised by Samuel Balula, Prof. Dominic Liao-Mc Pherson, Dr. Christoph Gebhardt, Prof. John Lygeros.

Autonomous mobile robots are increasingly being used for exploration and information gathering tasks, such as environmental monitoring, aerial surveillance, and tactile object exploration. We propose an algorithm in the continuous space based on Gaussian process and Traveling Salesman Problem and apply it to the task of volume estimation via vision-based flight. We formulate the problem of path planning for volume estimation as an informative path planning problem based on Bayesian quadrature. We also introduce an efficient multi-query planner for vision-based flight. Simulation results show that our approach estimates volumes accurately and travels a significantly smaller distance compared to other algorithms.

Online control of quadrotor and camera with MPC for robust vision-based flight by Yingjie Jiang. Supervised by Dr. Stefan Stevsic, Dr. Christoph Gebhardt, Samuel Balula, Prof. Dominic Liao-McPherson, Prof. John Lygeros.

In this project a model predictive controller is designed for a microaerial robot used for indoors inspection, relying on visual tags for localization. The drone navigates between way points, predefined by a higher level planner, while minimizing its state estimation uncertainty, a fundamental aspect of the monitoring task. By introducing a Kalman Filter into the optimization problem of model predictive control, the proposed real-time method brings a better state estimation quality for the quadrotor during a vision-based flight.

Control of a paramotor I by Ivan Gehri. Supervised by Samuel Balula, Dr. Eva Ahbe, Prof. Florian Dörfler.

An autopilot has been designed for a radio control model paramotor, with the aim of performing waypoint navigation. The microcontroller and sensors: GPS, gyroscope, compass, accelerometer, barometer, (indirect) force measurement on the breaking lines, and a Pitot tube have been embedded in the flying apparatus, contained in a 3D printed casing. The control can be taken over by the remote control at any time to ensure safety of operation. A 2D model of the dynamic system allows for the initial controller design in a simulated environment. Experimental results, using solely the GPS sensor, have validated the setup, but haven't yet achieved autonomous navigation.

Control of a paramotor II by Timo Laaksonlaita. Supervised by Dr. Eva Ahbe, Samuel Balula, Prof. Florian Dörfler.

This project deals with the control of a paramotor in straight level flight. The paramotor is a two-body system connected through cords, making it a challenging system to model and control compared to other UAVs. A simplified six degrees of freedom model is derived, and the unknown model parameters are determined through flight experiments. By linearizing the model around a trim equilibrium, the longitudinal and lateral dynamics of the paramotor get decoupled, allowing for separate control design of these two subsystems. Based on the linearized models, MPC controllers for longitudinal and lateral dynamics are developed and tested in simulation. The performance of the controllers to perturbed plants is analyzed and possible improvements for the overall system are suggested.

Control of a paramotor III by Gianni Pasini. Supervised by Samuel Balula, Dr. Eva Ahbe, Prof. Roy Smith.

The aim of this semester project is to control a paramotor, so that it can autonomously fly and follow a path in 3D space. In order to do so, PID, LQR and MPC controllers are tested in a simulation environment, using a previously developed 6DoF paramotor model. In addition, PID and LQR controllers are implemented on a Pixhawk Autopilot and tested during experimental flights with a real paramotor. The results are presented together with some short videos. The comparison between simulation and experiments shows that it is possible to successfully control the paramotor, but some future improvements are necessary in order to achieve fully autonomous flight.

Optimizing Control for CodinGame racing bot by Silvia Ritsch. Supervised by Anil Parsi, Samuel Balula, Prof. Florian Dörfler.

The online coding challenge Coders Strike Back is a competition to implement the fastest podracer, a hovercraft-type vehicle. The objective is to pass through a series of checkpoints in minimum time. The focus of this project is to design controllers for this problem, and two controllers were designed. The first is a PID controller which projects the distance to the current checkpoint onto the line of sight. Additionally, an MPC controller was designed which minimizes the distance to the current checkpoint. The knowledge of the next checkpoint was also encoded by modifying the objective when close to the current checkpoint. It was observed that in several randomly generated arenas, the MPC controller consistently outperforms the PID controller.

Modeling of a Traveling Wave Thermoacoustic Engine by Tim Liechti. Supervised by Prof. Andrea Iannelli, Samuel Balula, Prof. Roy S. Smith.

This semester project considers the problem of modeling thermoacoustic machines. They have significant potential for environmentally friendly and low maintenance solutions for electricity generation from waste heat or solar heat, and can also be employed for refrigeration and air conditioning. An overview of existing modeling methods is presented together with advantages and drawbacks, and their area of application is explored. Two approaches are then considered in more detail in the project. The transmission matrix method is used to get a first estimation of the onset temperature and the operating frequency. The solver deltaEC is instead used to model the steady-state operation. The modeling environment deltaEC is a well established tool to investigate new prototypes and to inform design choices. It is also a good tool for learning the modeling process and gaining some intuition on the acoustic field generated by these machines. deltaEC was used to model and analyze a thermoacoustic engine found in the literature (and for which experimental data are available), consisting of a two-stage looped traveling wave engine that employs a push-pull linear alternator for electric power generation. After the modeling procedure with deltaEC is explored, the validity of the model is investigated and possible improvements are suggested.

Machine Learning Modelling for Path-following Control by Jonas Holzem. Supervised by Samuel Balula, Dr. Alisa Rupenyan, Prof. John Lygeros.

Path-following with positioning systems often requires tracking accuracy in the micrometer range that cannot always be achieved by the built-in controllers. Enhancing the accuracy can be achieved by optimizing the controller reference inputs to compensate for deviations, using the available information from previous runs. The optimization procedure requires an accurate model predicting these deviations for new geometries that have not been processed previously on the system. The goal of this project is to achieve a model with sufficient accuracy to be able to predict the system dynamics for new geometries, using representative trials on the system as training data. The approach aims to explore one step, as well as multiple step trajectory predictions for a given geometry. Different neural network architectures were tested and optimized and sufficient accuracy was achieved for the one-step prediction case. Furthermore, it is demonstrated that the trained network can be integrated in the optimization framework.

Learning from Simulation, Racing in Reality: Sim2Real Methods for Autonomous Racing by Eugenio Chisari. Supervised by Dr. Alisa Rupenyan, Dr. Alex Liniger, Samuel Balula, Prof. John Lygeros.

Reinforcement Learning (RL) methods have been successfully demonstrated in robotic tasks, however, their application to continuous state-action systems with fast dynamics is challenging. In this work, we investigate RL solutions for the autonomous racing problem on the ORCA miniature race car platform. When training a deep neural network policy using RL methods only using simulations, we observe poor performance, due to model mismatch also known as reality gap. We propose three different methods to reduce this gap, first we propose a policy regularization in the policy optimization step, second, we use model randomization. These two methods allow learning a policy that can race the car without any real environment interactions. Our third method improves this policy, by running the RL algorithm online while driving the car. The achieved performance on the ORCA platform is comparable to that achieved previously by a model-based controller, in terms of lap time, and improved with respect to track constraint violations.

Spin Shots for a Robotic Billiard Player by Luzian Hug. Supervised by Samuel Balula, Sandeep Menta, Dr. Joe Warrington, Prof. John Lygeros .

Beside pocketing balls, a vital part of the game of snooker is also bringing the cue ball into an advantageous position after the shot. Building upon several preceding projects, we develop a robotic snooker player to the point of being able to not only pocket a ball successfully, but also controlling the trajectory of the cue ball over its whole movement. In order to achieve this, the cue strike, the motion of the ball on the table, the collision between two balls and between a ball and a cushion are modeled physically. By inverting the physical model, we find the shot velocity and amount of spin needed in order to place the cue ball in the desired location after the shot. We then test the model on the actual robot. Furthermore, the robot's state estimator and controller are improved in order to generate meaningful results. The time required to line up a shot with the robot is reduced from several minutes to between 15 and 30 seconds. With the updated system, we are able to place the cue ball within 324 mm of the desired position in 68 percent of shots.

Real-time Artificial Intelligence for a Robot Billiard Player by Andreas Schlaginhaufen. Supervised by Dr. Joe Warrington, Dr. Marcello Colombino, Sandeep Menta, Samuel Balula, Prof. John Lygeros.

With its continuous state and action space and stochastic outcomes snooker provides major challenges for an artificial intelligence player. Building upon previous projects, we break the decision making problem down using tree search methods for stochastic games. In addition, an efficient action filter is implemented to restrict the set of possible actions. Alongside with this, we use the *-Minimax algorithm to implement two different tree search players. One of them follows the standard approach of evaluating a heuristic evaluation function at leaf nodes and for the other one we propose a rollout according to some rollout policy to play until a terminal state is reached. Then these players are combined with the two different utilities of maximizing the probability of winning vs maximizing the expected lead. Simulating different scenarios we show that the rollout based players are performing best, while still being capable of playing a whole game in less than 10 minutes time.

Design and deployment of a control board for thermoacoustic experiments by Martin Stefan Baumann. Supervised by Prof. Andrea Iannelli, Samuel Balula, prof. Roy Smith

We design an in-house control board tailored for thermoacoustic experiments. Thermoacoustic systems pose special requirements on the hardware such as a high sampling rate, compatibility with different types of sensors signals and voltage levels, and low input to output latency. The board includes signal conditioning, microcontroller, and output amplifier units, while a Raspberry Pi is used to collect and store the data and to provide a web interface to the hardware platform. To demonstrate the capabilities of the developed hardware, experiments and system identification of the Rijke tube, a prototype of nonlinear thermoacoustic instabilities, have been performed. Open loop data acquisition has been used to preliminary characterize the onset and main features of the limit cycle oscillations arising due to the dynamic coupling between pressure and heat release. A proportional feedback controller, sensing pressure measurements and actuating a speaker, has then been implemented with a twofold aim. Firstly, the branch of periodic orbits emanating from the Hopf bifurcation point has been experimentally constructed by using the controller gain as bifurcation parameter. This has led to observe hysteresis, which in turns indicates the presence of a subcritical Hopf bifurcation. Secondly, the controller has been used to stabilize the limit cycles and thus identify the underlying LTI system. Two different closed loop system identification methods have been compared to obtain the transfer function between input speaker and pressure measurement at one location of the tube. This work resulted in the publication:

[11] A. Iannelli, M. S. Baumann, S. Balula, and R. S. Smith, "Experiments and identification of thermoacoustic instabilities with the rijke tube," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 757–763, IEEE, 2020 .

Machine learning of billiard ball dynamics by Divya Guruswamy. Supervised by Dr. Joe Warrington, Sandeep Menta, Samuel Balula, Prof. John Lygeros.

From the strike of the cue stick to the final positions on the table, billiard balls have a number of equations involved. There are multiple parameters that influence the behavior of the balls thus making it difficult to represent it in terms of analytical equations. Machine learning models allow prediction of the behaviors without handling these complex relations. This project aims to model the behavior of the ball in case of spin shots, to allow decision making in future projects. Initially, the ball collisions in case of 3D simulations having spin and 2D model having no spin are observed to understand the impact of spin on the collision. For a complete spin shot, model of cue strike with different parameters and ball collision model are trained separately and then combined. Finally, two models, one for predicting the final positions on table given initial parameters, and second for predicting the required initial parameters, given final positions are built.

Implementation of a Convex Optimization Algorithm on a Graphics Processing Unit by Michel Schubiger. Supervised by Dr. Goran Banjac, Samuel Balula, Prof. John Lygeros.

Large-scale convex optimization problems arise in various practical applications. Even though there exist many efficient methods for solving these problems, such as the alternating direction method of multipliers (ADMM), they may take minutes or even hours to compute solutions of very large problem instances. In this thesis we explore the possibilities of using a graphics processing unit (GPU) to accelerate ADMM. We use OSQP as a state-of-the-art implementation of ADMM to analyze the potential to parallelize the algorithm. We identify several parts of the implementation that could be accelerated by using a GPU, such as the direct linear system solver, which we replace with an iterative conjugate gradient (CG) method implemented on a GPU. Our implementation written in CUDA C has been tested on many medium- to large-scale problems in applications ranging from engineering to statistics and finance. The GPU-accelerated algorithm outperforms the CPU implementation by up to 2 orders of magnitude for problems that take more than 15 minutes to solve by the standard OSQP implementation.

Building a Control System for the Hydroponic Gardening System by Alexander Baltazar Almeida, Jacob Clarysse, Jente Clarysse, Kathrin Schleicher, Martin Vahlensieck. Supervision by Samuel Balula, and Prof. John Lygeros.

Design and implementation a control system for a hydroponic gardening system where vegetables grow in an automated hydroponic environment. Hydroponics is a growing technique where crops are cultivated on water instead of soil. The nutrients in the water, the pH-level and the temperature of the water are measured and adapted during the growing process. In our implementation the water flows continuously, driven by a pump in the reservoir, a method known as the nutrient flowing technique (NFT).

System identification and control of a hydroponic based gardening system by Jacob Clarysse, Alexander Almeida, and Martin Vahlensieck. Supervison by Samuel Balula, Prof. John Lygeros.

In this project a hydroponic NFT gardening system is modeled, identified and controlled, using a PI and LQG controllers. A clustering filter is used for the noisy measurements of the level sensors. Finally, the manually adjustable outflow of the system is estimated dynamically and incorporated in the controller.

1.6 Thesis Outline

In Chapter 2 we propose an optimization-based method for the design of reference trajectories with applications to PMS, based on different models built from experimental data, and computed exclusively offline. In Chapter 3 we propose an OB-ILC method for the same type of PMS, where we use model derivative information as well as successive evaluations of the system, both in simulation and experimentally. In Chapter 4 we propose a surface reconstruction method, as well as a greedy algorithm for the planning of trajectories for a quadcopter equipped with a LiDaR scanner. Finally, in Chapter 5 we summarize the conclusions of this work and list possible future research directions.

CHAPTER 2

Trajectory optimization for precision motion systems

In this Chapter, we propose an optimization-based reference (input trajectory) design (pre-compensation) methodology for simultaneously improving the precision and productivity of PMS by leveraging the widespread availability of system data. Our methodology is guided by the following requirements: (i) it does not require modifications to the existing hardware and low-level controllers, making it suitable for application in production machines; (ii) it does not require a physics-based model of the machine, relying exclusively on experimental data; (iii) it can be applied to individual machines with diverse technical characteristics; and (iv) it can produce good results for previously unseen parts at the first attempt.

These requirements are motivated by the requirements of “Industry 4.0” where machines are no longer standalone setups and are instead connected in a network, making data more easily available [12]. This data can be exploited to improve performance; for example, it can be leveraged to create data-driven models, avoiding the labour intensive process of developing and maintaining first principles based ones [13]. Future manufacturing is also expected to be more distributed and personalized (e.g., “Manufacturing as a service”) [14], leading to smaller volumes for any given part, possibly produced across a pool of non-homogeneous machines. This makes it more important to exploit data to ensure that machines can execute designs with minimal calibration.

Chapter Outline

This Chapter is organized as follows. In Section 2.1 we summarise the state-of-the-art and contributions, in Section 2.2 we precisely define the problem tackled, while in Section 2.3 we describe the modeling structure, the model fitting strategy, and the accuracy achieved. In Section 2.4 we detail the input trajectory design strategy proposed, specifying the optimization problems. In Section 2.5 we show experimental validation for the method with a set of test cases.

2.1 State-of-the-art

Several methods for improving PMS performance have been proposed in literature. One method is to re-designing the feedback controller. A common approach is Model Predictive Contouring Control (MPCC) [15], wherein a model of contouring error is minimized in a receding horizon fashion, drawing on methods from autonomous racing [16]. However these methods have high online computational loads, rely on accurate process models, and require modifying the inner-loop controllers.

Another common approach in manufacturing and precision motion systems is *learning-based control* [17, 18]. These methods assume that the processes is repeatable and adjust the input trajectories, using the error from the previous repetition as feedback. The input trajectories can be modified between iterations (Iterative Learning Control, and Run-to-Run control), or periods (Repetitive Control). These methods can exploit models [19], include constraints [20] , and are effective for repeatable processes. However, the computed trajectory is specific for a particular part and machine, with multiple trials required to achieve the desired accuracy. While this may be acceptable in large production runs, it may be inefficient for smaller runs e.g., in "manufacturing as a service" applications.

The final major class of methods is pre-compensation, which aim to modify the input trajectory offline to reduce the contouring error for arbitrary parts [21]. A common approach is to drive a model of the contouring error to zero by manipulating the input trajectory using a classical controller (e.g., a PID) [22]. These methods cannot include look-ahead or constraints, motivating the use of offline MPCC algorithms based on physics-based nonlinear [23], identified linear models [24] or combined with a reference governor [25]. In [26] the compensation is computed online using previous measurements and a linear model. In [27] the contour error is predicted with an Artificial Neural Network (ANN), and the pre-compensation computed with reinforcement learning. This approach does not allow constraints to be included and uses only zeroth-order information about the ANN model, despite the ready availability of ANN derivatives. In [28, 29] a simultaneous feed rate optimization and error pre-compensation method is proposed for a system described by a linear model. This approach is analogous to the first stage optimization described in 2.4.

Therefore, to the best of our knowledge, there is a gap in the existing literature on pre-compensation methods for improving PMS performance/ productivity that are data-driven, capable of generalizing to novel geometries, and applicable to a broad variety of machines that can only be accurately described by a nonlinear model.

Contributions

In this Chapter we tackle the challenge of simultaneously improving the performance/ productivity of a PMS using only data from the system. Our main contribution is a novel pre-compensation method based on data-driven modeling and trajectory optimization. Our approach does not require modification of the design or control software of the system (i) and is fully data-driven (ii). Moreover, it enables adaptation to changing conditions or system degradation through retraining the underlying data-driven model (iii). Finally, the approach is independent of the target geometry and significantly shifts the precision/ productivity trade-off curve. Relative to a method without pre-compensation, our proposed method is 30-60% more accurate for new geometries for same trajectory time budget or alternatively, trajectories can be traced in 25-50% of the time for the same contouring accuracy. (iv).

2.2 Problem Statement

Consider a tooltip, e.g., the laser in a laser cutting machine, which traverses a two-dimensional (2D) workspace $\mathcal{W} \subset \mathbb{R}^2$. The position of the tooltip as a function of time is denoted by $\gamma(t) = [x(t) \;\; y(t)]^\top$. The ultimate goal in precision motion planning is to have the tooltip trace a spatial target geometry $\Xi : [0, S] \to \mathcal{W}$, where S is the path length. The target geometry Ξ is assumed to be a continuous function parameterized by the path progress variable s. Typical design specifications require the target geometry to be traced with a precision of tens of micrometers; for example, in industrial laser cutting systems a typical precision specification is 20 μm

Most industrial precision motion systems come with a built-in low-level controller designed by the manufacturer to track an input trajectory $r : [0, T] \to \mathcal{W}$, where $T > 0$ is the time necessary for the tooltip to complete tracing the target geometry. When r is given as an input to the low-level controller, its actions and the dynamics of the machine give rise to an output trajectory $\gamma : [0, T] \to \mathcal{W}$; note that for simplicity we assume that both the input and the output trajectory lie in the workspace $\mathcal{W}$. We denote the mapping between the commanded input

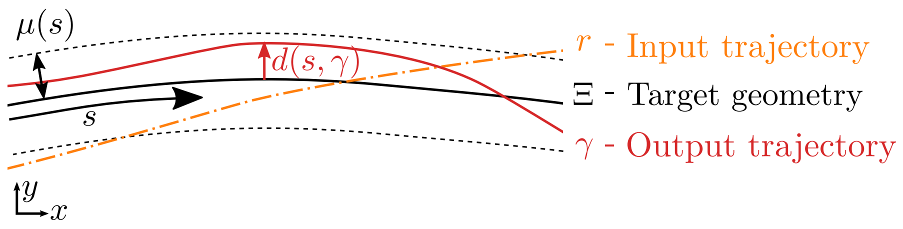


Figure 2.1: A low-level controller tracks the input trajectory r, producing the output trajectory γ. The deviation $d(s,\gamma)$ measures the distance from γ to the target geometry Ξ as a function of the path progress s. We desire that the deviation is smaller than the tolerance $\mu(s)$ at all points.

trajectory r and the resulting output trajectory γ by

$$\gamma = F(r) + e, \tag{2.1}$$

where the function $F : \mathcal{W}^{[0,T]} \to \mathcal{W}^{[0,T]}$ encapsulates the closed-loop dynamics of the machine. The noise e is zero-mean and quantifies the repeatibility of the system, a typical standard deviation is $\approx 2\,\mu$m. Ideally, the low-level controller is well designed so that $\gamma \approx r$ and F is approximately the identity function.

We are interested in the following inverse problem: Given the target geometry Ξ, how do we generate an input trajectory r for the machine, so that the resulting output $\gamma = F(r)$ matches Ξ as closely as possible. In addition to the usual difficulties associated with inverse problems, for precision motion systems the function F is often poorly known, partly due to lack of information about the low-level controller.

In standard practice, input trajectories are generated by traversing the path at a constant speed $v > 0$, i.e. setting $r(t) = \Xi(vt)$. However, due to sensor noise, actuator limits, the inertia of the machine, and other factors we observe experimentally a significant error between γ and r, especially when v is high. This induces a natural trade-off between speed and precision that is characteristic of precision motion systems. Our aim is to improve this trade-off by learning the function F through preliminary experiments, then inverting it using optimization.

Formally, let $d : [0,S] \times \mathcal{W}^{[0,T]} \to \mathbb{R}$ be the deviation function, defined as the projection of the output into the target geometry $d(s,\gamma) = \min_{t\in[0,T]} \|\Xi(s) - \gamma(t)\|_2$. Our goal is then to solve the optimization problem

$$\begin{aligned} \min_{r} \quad & T(\gamma) + \lambda \int_0^S d(s,\gamma) ds \\ \text{s.t.} \quad & \gamma = F(r), \\ & d(s,\gamma) \leq \mu(s),\ \forall s \in [0,S]\,, \end{aligned} \tag{2.2}$$

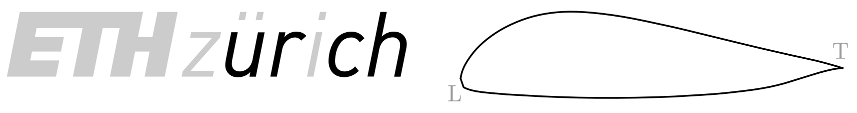


Figure 2.2: The two test cases used. The left panel shows Letters test case, where the target geometry is the contour of the letters marked in black. The right panel shows the airfoil test cases. The outline is labeled with the leading (L) and trailing (T) edges locations.

where $T(\gamma)$ is the duration of the output trajectory, $\lambda > 0$ governs the trade-off between speed and accuracy, and $\mu(s) : [0, S] \to \mathbb{R}$ is a tolerance bound around the target geometry; for the typical industrial laser cutting systems mentioned above, one would use $\mu(s) = 20\ \mu\text{m},\ \forall s \in [0, S]$. The different quantities are illustrated in Figure 2.1.

The optimization problem (2.2) is stated in a trajectory space and is not computationally tractable as written. In the following Sections we present and experimentally validate a practical methodology for approximately solving (2.2) using only data gathered from the process. Because the methodology is data-driven and noninvasive (in the sense that it does not require any knowledge of the underlying controllers but only the ability to run experiments on the machine) it is applicable to a wide range of practical industrial applications.

While our proposed methodology is applicable for general PMS, we use the experimental setup described in Section 1.2 as a running example throughout this work and to experimentally validate our approach.

Test cases

We selected two test cases, a series of letters and an airfoil, as target geometries to validate our proposed methodology. These geometries contain a rich set of features that are representative of industrial applications. To avoid "cherry picking", and to demonstrate generalization to new geometries, neither test case was used to generate training data or during the development of the method.

The **letters** test case, as can be seen in Figure 2.2, consists of four letters ("u","r","c", and "h") from the ETH Zürich logo. These letters were selected due to the rich set of relevant and challenging features they contain, including sharp corners, straight segments, and curves with various curvatures. The limiting contour of each letter is used as the target geometry.

The **airfoil** test case, shown in Figure 2.2, is based on a high-lift Eppler e344 airfoil [30]. Geometries of this kind are typically laser cut out of e.g., balsa wood, and are used in the primary structural sub-assembly of an aircraft. The main

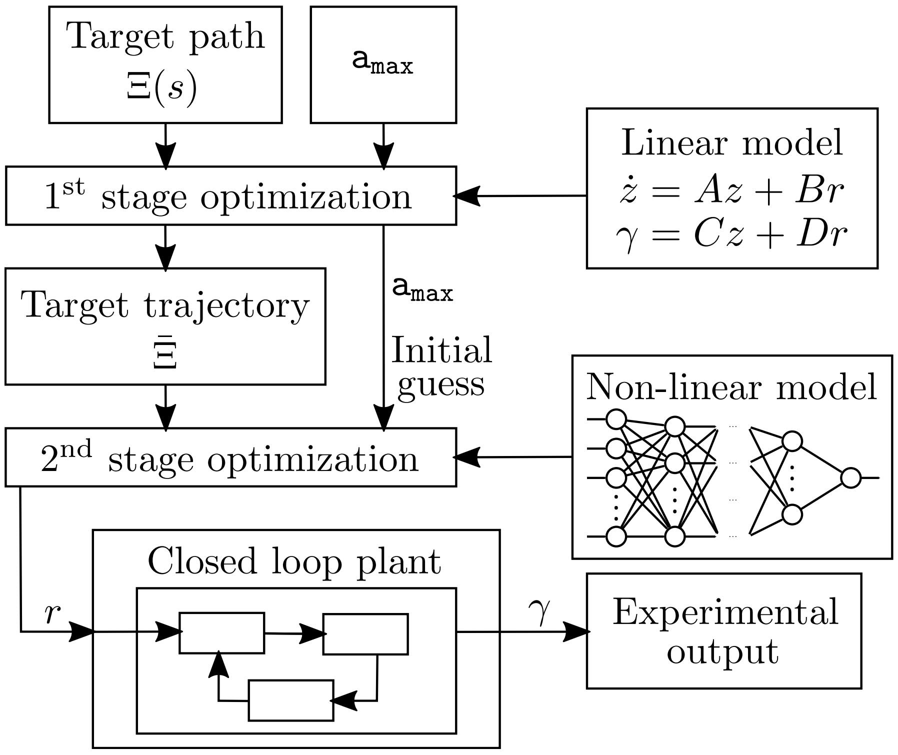


Figure 2.3: In the first stage optimization (2.8) a minimum time trajectory for tracking the target path $\Xi(s)$ is computed. The speed and acceleration profile from the first stage is used to sample the target geometry at constant sample rate of 400 Hz, creating a surrogate target discrete time trajectory $\bar{\Xi}$. In the second stage optimization (2.10) nonlinear model is used to find the best input trajectory r to track $\bar{\Xi}$. The maximum allowable acceleration $\mathtt{a_{max}}$ governs the accuracy/trajectory time trade-off. The optimized input trajectory r is given to the experimental apparatus.

challenging features of the airfoil are a cut-out at the leading edge (L) used to attach a structural spar, and the trailing edge (T) whose geometry has a strong effect on the aerodynamics of the aircraft.

Architecture overview

We approach the trajectory design problem in several stages. First, we design experiments to generate informative input-output data for the unknown mapping F and use the resulting data to identify two models for the system, a low-fidelity linear one and a high-fidelity one based on an ANN. Throughout, we incorporate machine learning best practices, such as independent training and validation data sets, normalization, and diverse training data to avoid over fitting and to ensure that the resulting models are able to generalize to trajectories outside the training data set, enabling optimization of new geometries.

Using these models, we propose a two-stage optimization-based approach for offline optimization of references as schematized in Figure 2.3. In the first stage, we compute an input trajectory by solving a contouring control problem using the linear model of the system. This first stage solution yields a fast trajectory that respects the problem constraints, by reducing the speed in intricate features of the path, such as sharp corners, and accelerating through long smooth segments. The output of the first-stage is used as the initial condition for the second stage, which employs the high-fidelity neural network model to correct errors from the first stage. The resulting input trajectory is then given to the low-level controller as an input trajectory.

2.3 Data driven modelling

As a first approximation, we model the system using a continuous-time linear model which captures the dominant dynamics. The structure of this model allows for efficient computations of a time-optimal solution in the first stage, but lacks the high-precision required by the application. This shortcoming is alleviated by a second high-fidelity neural network model, used to refine the initial solution.

Low-fidelity linear model

The closed-loop system maps trajectories to trajectories and is of the form $\gamma = F(r)$. We approximate this mapping using a Linear Time Invariant (LTI) state space model of the following form

$$\dot{z}(t) = Az(t) + Br(t), \tag{2.3a}$$
$$\gamma(t) = Cz(t) + Dr(t). \tag{2.3b}$$

The model (2.3) uses a non-physical hidden state $z \in \mathbb{R}^4$; the matrices A, B, C, and D are identified from experimental data using the `MATLAB` function `n4sid` from the system identification toolbox. The dimension of the state was chosen empirically, noting that increasing the model order further has diminishing returns.

High-fidelity Nonlinear Model

To obtain a higher precision model, the 2D-positioning stage is modeled using two independent causal ANN, one for each axis. Unlike (2.3) the nonlinear model operates in discrete time, at a sample frequency of 400 Hz and it has the following form

$$\gamma_i = f_{nlm}(r_{i-h}, \ldots, r_{i-1}), \tag{2.4}$$

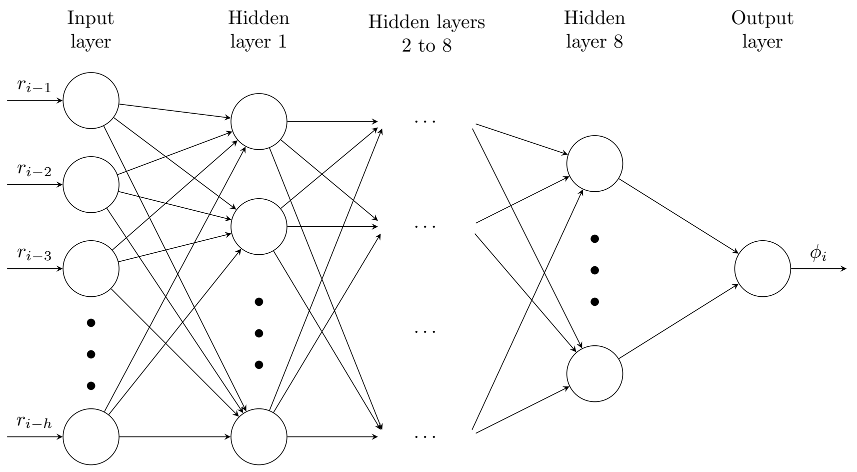


Figure 2.4: Neural network structure. Each layer is fully connected to the next. The input layer has 200 neurons, there are 8 hidden layers, with $\{200, 200, 100, 100, 50, 25, 12, 6\}$ neurons, and the output layer has 1 neuron. The activation function of the hidden layers is a LeakyReLU function with slope 0.01, and a bias term. There are total of 117322 weights to be adjusted in the training. Two independent networks with identical structure were trained, one for each axis.

where γ is the output, f_{nlm} is the nonlinear function defined by the ANN, r is the input trajectory given to the closed loop system, and h is the length of the input history considered for the prediction. The two ANN are structurally identical, but are trained separately, leading to different weights. Each takes as input the most recent 500ms of the input trajectory for the corresponding axis, subsampled at 400Hz, leading to 200 inputs. Each network has a single output, namely the predicted position at the subsequent time step. The networks have 8 fully connected hidden layers of LeakyReLU [31] activation functions with a slope of 0.01, and a triangular structure, as illustrated in Figure 2.4. The length of the time history was selected to be approximately 25 times the timescale of the slowest mode of the identified linear model. The sample rate was selected by analyzing how much distance can be covered at the highest expected speeds, and comparing it to the desired range of precisions. The number of hidden layers and their dimensions were selected empirically, to achieve good performance with the lowest possible complexity. The choice of feed-forward ANN was made to ensure that the model prediction accuracy does not degrade over time due to compounding errors. We also note that by training separate ANN for each axis we are implicitly ignoring coupling effects between the two axes; this choice is supported by experimental data that suggests coupling effects are negligible, as will be demonstrated in Section 2.5.

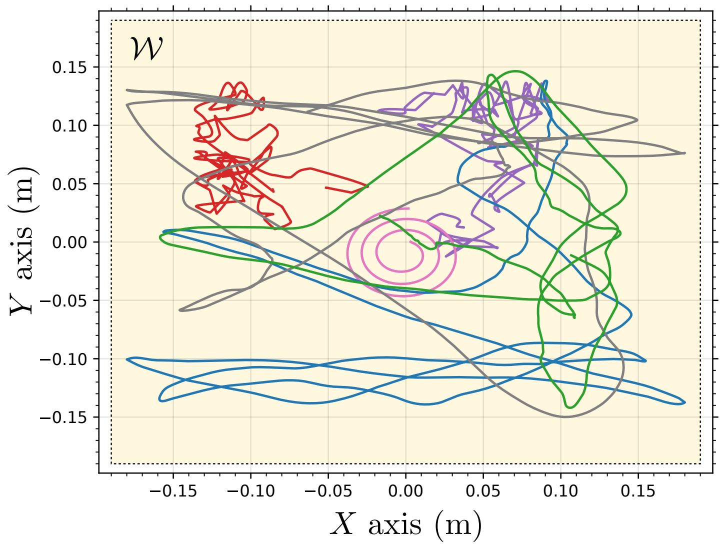


Figure 2.5: Sample of the data collected to train and test the nonlinear neural network model. Six trajectories are shown, five randomly generated and one regular shape (a spiral). The trajectories are contained in the workspace $\mathcal{W}$, shown in yellow background.

Training data generation

Choosing input trajectories that cover a wide range of operational regimes is essential for generating informative data to train models that can generalize to previously unseen trajectories.

Linear Model

For the identification of the linear model we opted for randomly generated trajectories, yielding 6×10^6 points in total. By randomly changing the velocity, acceleration, and jerk of the input trajectory we made sure that the trajectories represent features seen in real parts. This included, for example, curves of different radii, traversed at different speeds and appearing in different locations within the workspace, as well as sharp transitions in acceleration and jerk. The trajectories were not designed for a specific part, but for a particular set of constraints on velocity and acceleration.

Nonlinear Model

Having sufficient high-quality data is necessary to ensure that the ANN is accurate and generalizes well. We started by training a first generation ANN with the random trajectories used to fit the linear model, and a set of varied regular geometries including circles, polygons, and spirals. Each shape was traversed at varied speeds, constant for each trial. To enrich the data set further, the trained first generation ANN was used in the optimization methodology of Section 2.4 applied to the same regular geometries. This resulted in optimized input trajectories and, after applying these to the system, additional experimental data. This data was used to train the ANN further obtaining a second generation ANN. This training-optimization-retraining cycle can be executed recursively until the model prediction accuracy is acceptable. These data sets lead to a targeted reduction in over fitting in areas that are favored by the optimization. Overall, our strategy is to include sufficient random (exploratory) and structured (exploitative) data to ensure the ANN is both accurate and generalizes well in the areas of interest.

A representative training data sample is shown in Figure 2.5. In section 2.5 we present the prediction quality of a third generation ANN to generate the results of Section 2.5. The data was divided 80%/20% for training and testing purposes. This results in a ratio of ≈ 34 data points per parameter of the ANN. About 60% of the data comes from the randomly generated trajectories, and the remaining from regular shapes both before and after optimization. The experimental data is divided in segments of 500 ms, subsampled to achieve an effective sample rate of 400Hz, and scaled. For the loss function we use the mean square of the prediction error. We use batches of 16k segments which take maximum advantage of the GPU computation power. *Adam* [32] was used for the optimizer, with decaying learning rate. Modeling and training were done in *PyTorch* [33] with *NVIDIA CUDA* support.

Model Validation

The quality of the model predictions is evaluated over different data sets and acceleration limits. The results are summarized in Table 2.1.

The linear model training data set includes trajectories with a range of accelerations up to 5 $\mathrm{m\,s^{-2}}$. However, due to its simple structure, the linear model cannot overfit the data as it does not have the representative power to provide accurate predictions everywhere in its training set. We observed that it performs better when the acceleration is restricted to a lower value, even for trajectories not seen during the fit, such as the letters test case. The linear model accuracy is insufficient for the demanding requirements of the application.

Regarding the nonlinear model, the prediction accuracy is best for data with the same characteristics as the one used for the training. When presented with data coming from shapes which have not been used for the training of the model, the prediction accuracy degrades. However, the accuracy is still adequate for the application, as the standard deviations remain below 20 μm for each axis for accelerations below 3 $\mathrm{m\,s^{-2}}$.

		Training / Validation		$\mathtt{a}_{\mathtt{max}}$ 1.0 $\mathrm{m\,s^{-2}}$		$\mathtt{a}_{\mathtt{max}}$ 3.0 $\mathrm{m\,s^{-2}}$	
Model	Axis	μ	σ	μ	σ	μ	σ
–	–	μm	μm	μm	μm	μm	μm
Linear	x	3.0	48.0	4.7	30.1	5.4	57.2
	y	-13.2	156.3	0.0	108.5	0.5	229.4
Non-linear	x	0.0	3.2	-0.3	8.2	1.0	11.1
	y	0.0	6.0	-2.0	13.4	-2.3	18.7

Table 2.1: Prediction error mean μ and standard deviation σ of the linear and non-linear models. Both models are evaluated in the Letters test case after optimization, with different maximum acceleration values $\mathtt{a}_{\mathtt{max}}$. The linear model is also evaluated on its training data set, and the non-linear model on its validation data set.

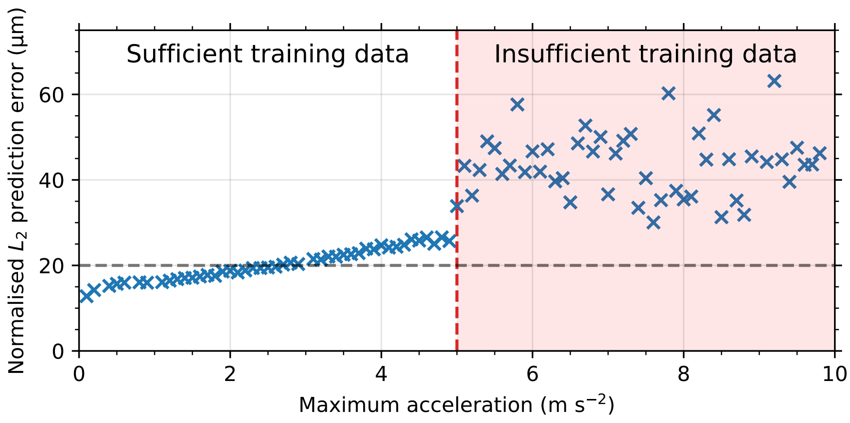


Figure 2.6: Prediction error of the nonlinear model as a function of the maximum acceleration $\mathtt{a}_{\mathtt{max}}$ of the input trajectory. The error is evaluated for the letters and airfoil test cases. Above 5 $\mathrm{m\,s^{-2}}$ the training data do not contain enough information to train the model properly.

Figure 2.6 shows how the standard deviation of the prediction evolves with the maximum acceleration $\mathtt{a_{max}}$ that was set as a constraint in the optimization of Section 2.4. For acceleration values where there is sufficient training data, the model prediction error increases gradually with the acceleration. If we try to use the model outside of the acceleration values it was trained for, the predictions rapidly deteriorate.

2.4 Trajectory Optimization

With the the data-driven models in hand, we now discuss the two-stage optimization architecture used to compute input trajectories. The decision to divide the optimization problem in two stages is motivated by the need to include the duration T of the trajectory in a computationally tractable manner. This is accomplished by using a variable time-discretization in the first stage, enabling inclusion of T as a decision variable. The second stage then fixes the duration T and refines the accuracy of the first-stage solution using the high-fidelity neural network model.

Our experiments suggest that treating the time discretization directly with the nonlinear model is intractable. A key difficulty is that real machines typically generate noisy output measurements in a sampled, time series format. These cannot be readily used to train the high-fidelity continuous time model needed for a computationally tractable variable-time discretization, due the challenges associated with numerical differentiation of noisy data. Trying to solve the inverse problem in one shot using the resulting model leads to low quality solutions; often the solver fails to even return a feasible solution in a reasonable amount of time.

On the other hand, skipping the second stage and inverting using only the linear model is computationally tractable, but typically leads to low quality solutions; this observation is supported by the model prediction errors shown in Table 2.1. Training a fixed sampling time nonlinear model and seeding with the solution of the first stage optimization substantially reduces the overall computation time and improves the quality of the solutions.

A key advantage of our methodology is that, unlike iterative learning control, once the models are trained and we are given a target geometry, the input trajectory is computed off-line, without the need for additional experiments. This is especially important for small batch manufacturing where it is crucial to minimize failed parts. However we can still use the information collected from experimental runs to improve the model.

First-stage Optimization Formulation

The purpose of the first stage is to fix the duration of the trajectory, T, by trading off speed and precision. We employ a contouring control approach with a fixed spatial discretization. This involves sampling the target geometry at N equally spaced points in space $\{\Xi_k \doteq \Xi(s_k) : k \in \{0, 1, \ldots, N-1\}\}$, where $s_k = k\frac{S}{N-1} = k\Delta s$.

We use the identified linear model (2.3) to approximate the process dynamics. As our approach involves a fixed spatial discretization Δs, the time-discretization must be variable. Starting with $t_0 = 0$, we denote by t_k as the time at which we would like the trajectory to reach the point Ξ_k and consider the non-uniform time steps $\Delta\tau_k = t_{k+1} - t_k, k = \{0, \ldots, N-1\}$ as decision variables; note that the time $T = t_N = \sum_{k=0}^{N-1} \Delta\tau_k$ at which the trajectory is completed is implicitly also a decision variable. We then obtain a discrete time model by applying the 4th order Runge-Kutta (RK4) [34] time-marching method to (2.3)

$$z_{k+1} = f_{\text{lm}}(z_k, r_k, \Delta\tau_k) \tag{2.5a}$$

$$\gamma_k = Cz_k + Dr_k, \tag{2.5b}$$

where f_{lm} is the RK4 function, and $\Delta\tau$ will be treated as decision variables.

To evaluate the contouring error d_k we start with the distance $\nu_k = \gamma_k - \Xi_k$ and introduce a local coordinate frame as shown in Figure 2.7. The linear transformation

$$T_k = \begin{bmatrix} \cos\alpha_k & \sin\alpha_k & \Xi_{k,x} \\ -\sin\alpha_k & \cos\alpha_k & \Xi_{k,y} \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

where α_k is the rotation angle of the local frame, encodes a translation and rotation between the local coordinate frame k and the global coordinate frame. With

$$\begin{bmatrix} \gamma_k \\ 1 \end{bmatrix} = T_k \begin{bmatrix} \nu_k \\ 1 \end{bmatrix}, \tag{2.7}$$

the output can be transformed from local coordinates ν_k to global coordinates γ_k. Since time discretization is variable, we can always ensure that the x^l component of ν_k is zero by adding a constraint to the optimization problem. The contouring error is then simply the y^l component of ν_k. The tolerance bound condition in (2.2) can then also be computed for the corresponding discretization point $\mu_k = \mu(s_k)$.

The aim of the optimization problem is to minimize the total time needed to complete the target geometry subject to the tolerance bounds, velocity and acceleration limits. To this basic formulation, we introduce an additional term for the input to promote smoothness; this is used to suppress high frequency content

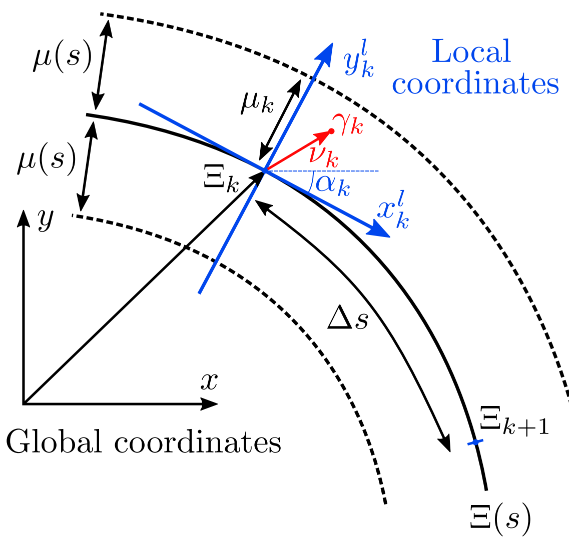


Figure 2.7: For every point Ξ_k of the target geometry $\Xi(s)$ a local coordinate frame is placed. x^l is tangent to the target geometry at Ξ_k and pointing in the direction of increasing s whereas y^l is orthogonal to x^l pointing to the port side. α_k is the angle between the global x axis and x^l. The maximum allowed deviation μ_k is computed from $\mu(s)$. Δs is the path distance between points Ξ_k and Ξ_{k+1}.

on the input trajectory, which is filtered out by the linear model. This results in the following optimization problem

$$\min_{\Delta\tau, r, \gamma, z, \nu} \quad \sum_{k=1}^{N-1} \Delta\tau_k + \frac{1}{N-2}\sum_{k=2}^{N-1} \|\partial^2 r_k\|_2^2 \tag{2.8a}$$

s.t.

$$z_{k+1} = f_{\mathrm{lm}}(z_k, r_k, \Delta\tau_k), \quad k = 1, \dots, N-1, \tag{2.8b}$$

$$\gamma_k = Cz_k + Dr_k, \quad k = 1, \dots, N, \tag{2.8c}$$

$$\Delta\tau_k \geq 0, \quad k = 1, \dots, N, \tag{2.8d}$$

$$\gamma_k \in \mathcal{W}, \quad k = 1, \dots, N, \tag{2.8e}$$

$$\begin{bmatrix} \gamma_k \\ 1 \end{bmatrix} = T_k \begin{bmatrix} \nu_k \\ 1 \end{bmatrix}, \quad k = 1, \dots, N, \tag{2.8f}$$

$$\nu_{k1} = 0, \quad k = 1, \dots, N, \tag{2.8g}$$

$$|\nu_{k2}| \leq \mu_k, \quad k = 1, \dots, N, \tag{2.8h}$$

$$\|\partial\gamma_k\|_\infty \leq \mathtt{v_{max}}, \quad k = 1, \dots, N-1, \tag{2.8i}$$

$$\|\partial^2\gamma_k\|_\infty \leq \mathtt{a_{max}}, \quad k = 2, \dots, N-1, \tag{2.8j}$$

where N is the number of discretization points, selected based on the total path length and tolerance bound, $\Delta\tau = \{\Delta\tau_k\}_{k=1}^{N-1} \subseteq \mathbb{R}$, $\partial\gamma_k = \frac{\gamma_{k+1}-\gamma_k}{\Delta\tau_k}$ and $\partial^2\gamma_k = \frac{\gamma_{k+1}-2\gamma_k+\gamma_{k-1}}{\Delta\tau_k\Delta\tau_{k+1}}$ are discrete derivative operators, $\nu = \{\nu_k\}_{k=1}^{N} \subseteq \mathbb{R}^2$ is the output trajectory in local coordinates, $\gamma = \{\gamma_k\}_{k=1}^{N} \subseteq \mathbb{R}^2$ is the output trajectory in global coordinates, $r = \{r_k\}_{k=1}^{N-1} \subseteq \mathbb{R}^2$ is the input trajectory, f_{lm} is the RK4 approximation of the linear model (2.5), with identified matrices C and D, T_k is the transformation matrix between global and local coordinates in (2.6), μ_k is the discretized maximum allowed deviation from the target geometry, $\mathcal{W}$ are the limits of the working space of the device, $\mathtt{v_{max}}$ is the maximum allowable speed, and $\mathtt{a_{max}}$ is the acceleration limit.

The cost (2.8a) contains two terms. The first penalizes the total time to perform the trajectory while the second penalizes the acceleration of the input, promoting fast trajectories with a smooth input, (2.8b) and (2.8c) imposes the RK4 discretization of the linear dynamics (2.5), (2.8d) requires the variable time step to be non-negative, (2.8e) imposes that the output stays within the working space of the device, (2.8f) the transformation between local and global coordinates in (2.7), (2.8g) requires that the x_k^l component of ν_k is zero, in which case the y_k^l component is the deviation, (2.8h) bounds the y_k^l component of ν_k to be less than the maximum allowed deviation, (2.8i) and (2.8j) impose component-wise limits on velocity and acceleration of the output. This formulation is analogous to [28] where the feedrate and contour error are optimized using a linear model.

Second-stage Optimization Formulation

Experiments applying the trajectory r generated by (2.8) directly to the machine suggest that this generally results in large deviations from the target geometry, as can be seen in Figure 2.11, which we attribute to the low fidelity of the linear model. To improve performance, we refine the trajectory generated by the first stage using the high-fidelity ANN model (2.4). In principle, it is possible to skip the first stage and directly embed the ANN model in an optimization problem similar to (2.8). However numerical experiments suggest that this often leads to poorer performance even compared to the linear model, as the solver tends to converge to poor local minima, or fails to return a feasible solution. Here we take advantage of the output of (2.8) to fix the time discretization and further improve precision through a second optimization problem based on the nonlinear model. In practice this leads to a more reliable and scalable overall method.

Given the structure of the nonlinear model, which is trained using time series data, we sample the target geometry at M equally spaced points in time

$$\bar{\Xi}_i \doteq \Xi(s(t_i)) : i \in \{0, 1, \dots, M\} \tag{2.9}$$

$t_{i+1} = t_i + \Delta t$, where Δt is the fixed sample rate of the time series data used to train the nonlinear model. Given this fixed discretization, the target geometry of the second-stage optimization problem is simply to minimize the deviation from the target trajectory while satisfying the tolerance, speed, and acceleration bounds. This leads to the optimization problem

$$\begin{aligned}
\min_{r,\gamma} \quad & \sum_{i=1}^{M} \|\gamma_i - \bar{\Xi}_i\|_2^2 && && \text{(2.10a)}\\
\text{s.t.} \quad & \gamma_i = f_{\text{nlm}}(r_{i-h}, \dots, r_{i-1}), && i = 1, \dots, M, && \text{(2.10b)}\\
& |\gamma_i - \bar{\Xi}_i| \leq \mu_i && i = 1, \dots, M, && \text{(2.10c)}\\
& \gamma_i \in \mathcal{W} && i = 1, \dots, M, && \text{(2.10d)}\\
& |\partial\, \gamma_i| \leq \mathtt{v_{max}} && i = 1, \dots, M-1, && \text{(2.10e)}\\
& |\partial^2 \gamma_i| \leq \mathtt{a_{max}} && i = 2, \dots, M-1, && \text{(2.10f)}
\end{aligned}$$

where M is the number of discretization points, $\gamma = \{\gamma_i\}_{i=1}^{M} \subseteq \mathbb{R}^2$ is the output, $r = \{r_i\}_{i=1}^{M} \subseteq \mathbb{R}^2$ is the input trajectory, $\bar{\Xi} = \{\bar{\Xi}_i\}_{i=1}^{M} \subseteq \mathbb{R}^2$ is the sampled target geometry, f_{nlm} is the nonlinear ANN model (2.4), μ_i is the tolerance bound evaluated for each discretization point, $\mathcal{W}$ are the limits of the working space of the device, $\mathtt{v_{max}}$ is the maximum allowable speed, and $\mathtt{a_{max}}$ is the acceleration limit.

The cost (2.10a) penalizes deviations of the output from the target geometry. The constraint (2.10b) imposes the nonlinear system dynamics modeled by the ANN, (2.10c) constrains the output to be within the tolerance bound for each discretization point, While (2.10d), (2.10e) and (2.10f) ensure that the output, its velocity and acceleration stay within working space and operational limits of the device.

In practice, to ensure feasibility, the tolerance constraint is replaced with an exact L_1 penalty which is implemented using slack variables [35]. For longer trajectories, due to memory constraints, the optimization problem can be solved with a receding horizon strategy. In this case we use an horizon of 11 steps. The maximum acceleration limit $\mathtt{a_{max}}$ is the main parameter determining the total trajectory time, tuning the solution along the accuracy/trajectory time trade-off curve.

Implementation Details

The optimization problems defined in (2.8) and (2.10) are both nonlinear and are solved using `IPOPT` [36], with the JuMP [37], and Casadi [38] interfaces in Julia and Python respectively. The first stage optimization problem (2.8) is initialized by taking as r the target geometry traversed at a constant speed. The solution of (2.8) is then used to initialize the second-stage (2.10).

The computer used throughout this work runs Arch Linux, with Linux kernel version 5.15, and it is equipped with a `GeForce RTX 2080 Ti` GPU with 12 GB of dedicated memory, an `Intel(R) Core(TM) i9-9900K` CPU @ 3.60 GHz and 64 GB of RAM.

For the problem sizes we considered, the first-stage optimization takes approximately 1 minute to complete ($N \approx 10^3$), and the second-stage takes approximately between 1 to 4 hours depending on the trajectory time ($M \in \{200, \ldots, 1000\}$). Lower $\mathtt{a_{max}}$ results in slower trajectories which take more time to compute due to the increased number of function evaluations required. Training of the ANN model takes approximately 24 h per axis.

2.5 Experimental Results

In this Section, we apply our proposed data-driven optimization methodology from Section 2.4 to the experimental apparatus in 1.2, using the test cases in 2.2. Our experiments demonstrate that the methodology is capable of improving system performance relative to the baseline – the trajectory obtained with the low-level controller following a not optimized input trajectory – by tracing desired geometries faster and more precisely.

Individual trajectories

We first focus on the letter "r" as a case study and consider a scenario with an acceleration limit of 1 $\mathrm{m\,s^{-2}}$; the individual results for the other letters in the test case are comparable and will be discussed collectively in Section 2.5. Applying our method results in an optimized input trajectory with a total time of $T = 0.807$ s. The experimental result of this new input trajectory is compared with a baseline performed in the same total time. For this the original, non-optimized shape (in this case the letter "r") is sampled at a constant progress speed $r(t) = \Xi\left(t\frac{S}{T}\right)$, where S is the total path length, which for a closed shape corresponds to the perimeter.

Figure 2.8 compares the output and input trajectories for the optimized and baseline cases in the $x - y$ plane while Figure 2.10a displays the same data as a function of progress.

Analyzing Figures 2.8 and 2.10a, we see that the optimized output effectively stays within the selected $\mu = 20$ μm while the baseline output deviates by more than $100\mu m$. The optimized output trajectory speeds up in areas with low curvature and slows down near corners or other intricate features. Figure 2.10a also illustrates that the optimized input trajectory deviates aggressively from the target geometry near areas where the baseline output performs poorly. This can be interpreted as an attempt to "cancel out" the error; The detail in Figure 2.8 illustrates this behavior in the $x - y$ plane.

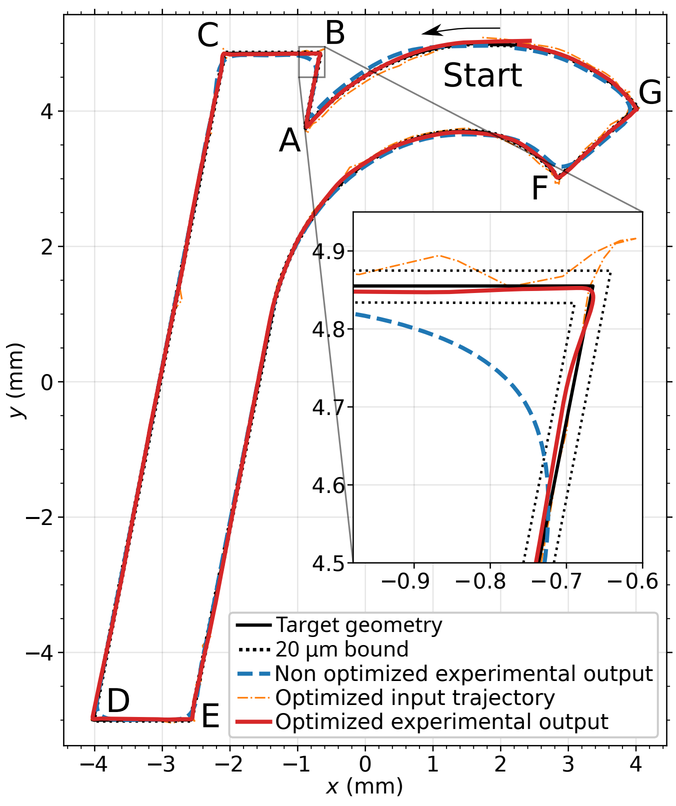


Figure 2.8: Letter “r” of the ETH Zürich Logo. Optimized with $\mathtt{a}_{\mathtt{max}} = 1\ \mathrm{m\,s^{-2}}$, including a detail view of station B. Note how the optimized input trajectory significantly deviates from the target geometry near the corner in B to compensate for the dynamics of the machine. This leads to significantly smaller error between the output and the target geometry compared to using the state-of-practice (non-optimized) input trajectory.

Figure 2.10b display the same information for the airfoil test geometry with a maximum acceleration of $2\ \mathrm{m\,s^{-2}}$. The baseline and optimized trajectories complete the part in 1.081 s. Similarly to the letter “r” example, the optimized trajectory slows down near intricate features, in this case the leading and trailing edges of the airfoil, and accelerates between them. Figures 2.9 and 2.10b show that the optimized input again seeks to “compensate” for deviations in the baseline trajectory.

To summarize, we observe that the optimizer exploits two main mechanisms to improve performance of the machine relative to the baseline system with the same trajectory time: 1. It reduces the speed of the input trajectory near intricate geometric features and increases it in areas with low curvature. 2. It “compensates” for errors in the baseline trajectory by moving the optimized input trajectory in an opposing direction. In effect, the optimizer exploits information encapsulated

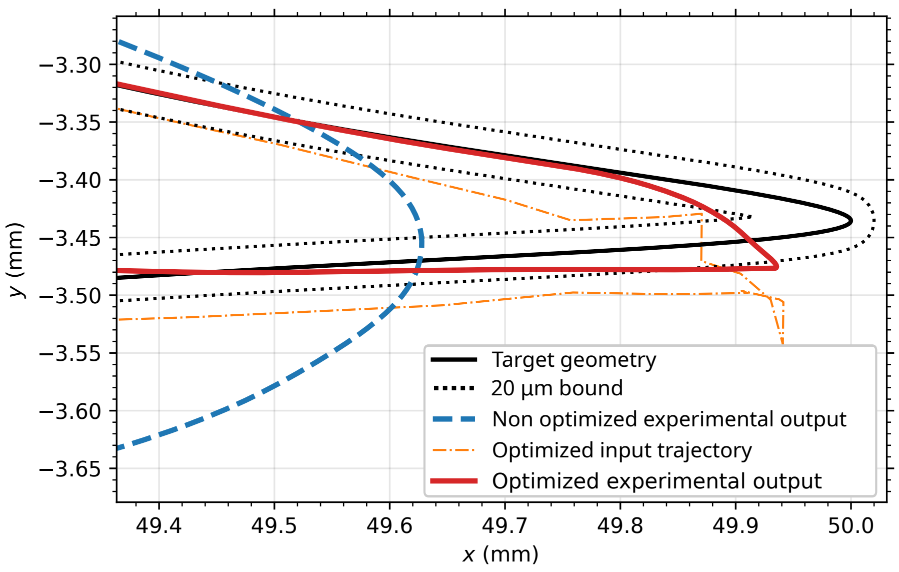


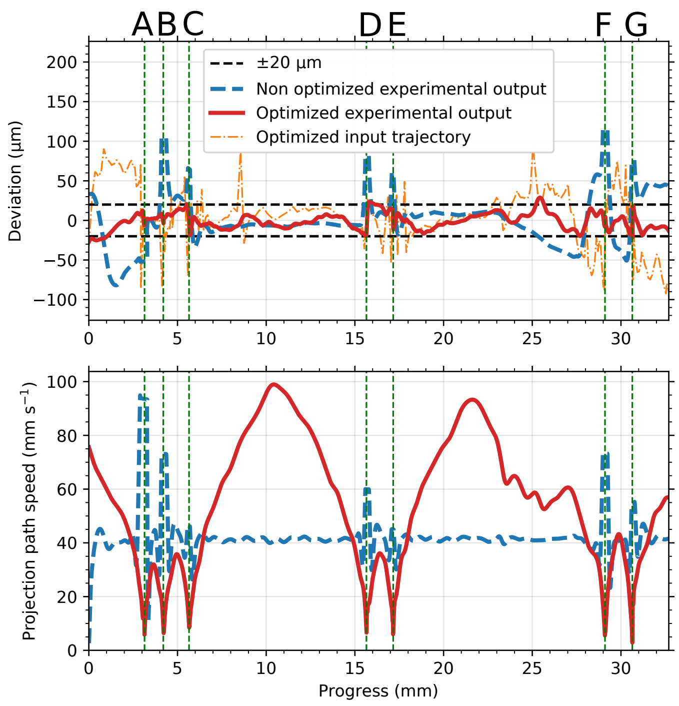
Figure 2.9: A close-up of the trailing edge of the airfoil test case with $\mathtt{a}_{\mathtt{max}} = 2\ \mathrm{m\,s^{-2}}$. The optimized trajectory is able to track the desired geometry more precisely than the baseline, though still not within the 20 μm tolerance band. The optimized input trajectory is not constrained by the tolerance band.

in the data-driven models to push the limits of system performance and adapt the input trajectory to the machine capabilities, for the selected maximum acceleration value.

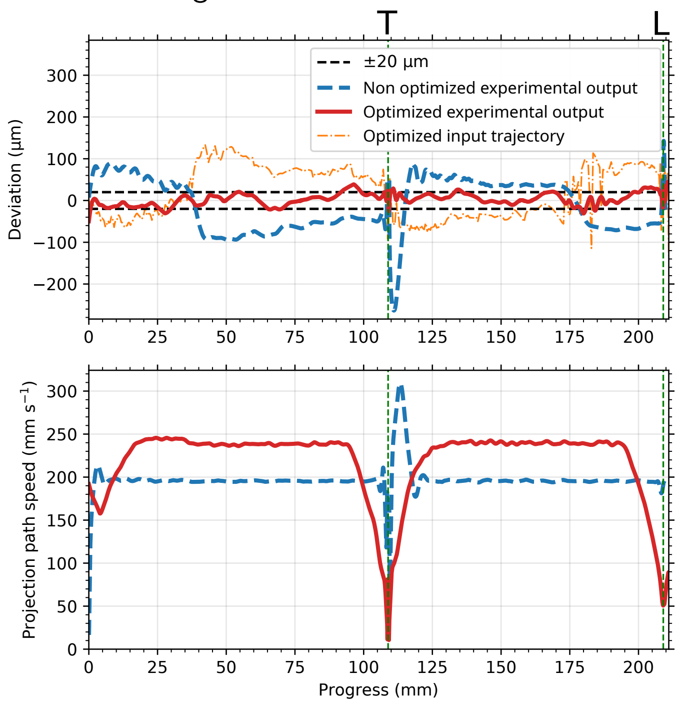
Precision-accuracy Trade-off

In precision motion systems there is a fundamental trade-off between speed and accuracy caused primarily by the inertia of the machine. In this Section, we demonstrate that our proposed methodology is able to improve system performance by shifting this trade-off.

In our formulation (2.8)-(2.10), the trade-off between speed and accuracy is controlled by the parameter $\mathtt{a}_{\mathtt{max}}$ which limits the maximum acceleration of the input trajectory. We generated trade-off curves for both the letters and airfoil test geometries by conducting experiments for different values of this parameter between $0.1\ \mathrm{m\,s^{-2}}$ and $3.3\ \mathrm{m\,s^{-2}}$ for the letters and between $0.1\ \mathrm{m\,s^{-2}}$ and $10\ \mathrm{m\,s^{-2}}$ for the airfoil, and tested each resulting input trajectory experimentally. Similarly to the two individual test cases presented above, the non-optimized input trajectories are subsequently run with the same total time as the one found for the optimized input trajectories.



(a) Letter "r" of the ETH Zürich logo optimized with $a_{\mathtt{max}} = 1\ \mathrm{m\,s^{-2}}$. The letters A-G refer to the points labeled in Figure 2.8.



(b) Airfoil test case, optimized with $a_{\mathtt{max}} = 2\ \mathrm{m\,s^{-2}}$. The letters L and T refer to the leading and trailing edges labeled in Figure 2.2.

Figure 2.10: Top panels show the deviation from the target geometry for the optimized and non-optimized output, as well as the optimized input trajectory. The non-optimized input trajectory by construction does not deviate from the target geometry. Bottom panels shows the velocity of the optimized and non-optimized outputs projected onto the target geometry.

We use the normalized norms

$$L_1 = \frac{1}{M}\sum_{i=1}^{M} \min_s ||\gamma_i - \Xi(s)||_2 \,, \tag{2.11a}$$

$$L_2 = \sqrt{\frac{1}{M}\sum_{i=1}^{M} \min_s ||\gamma_i - \Xi(s)||_2^2} \tag{2.11b}$$

$$L_\infty = \max_{i\in\{1,\dots,M\}} \min_s ||\gamma_i - \Xi(s)||_2 \tag{2.11c}$$

to quantify the deviation between the output γ and target geometry Ξ, both in simulation and experimentally.

The results are shown in Figure 2.11. In all cases the optimized experimental output results in significantly more precise trajectories for a given part completion time than the non-optimized trajectories. The input trajectories optimized in the first stage exhibit higher deviations than the non-optimized trajectories. This is to be expected given the prediction error of the linear model, as quantified in Table 2.1. Moreover, the simulated deviation is lower than the experimental one due to model mismatch. Figure 2.11 also show that for the same deviation values the optimized trajectories take less time to complete. For example, in the letters test case the non-optimized version takes 8.1 s and achieves a deviation of 26 μm, while the optimized version takes only 2.2 s for an identical deviation. In this case the optimized trajectory reduces the time needed to trace the shape in 73%. For the airfoil test case a similar analysis yields a reduction of 57% for a deviation of 47 μm. In Table 2.2 we show the results for 5 different test cases, each at two different $\mathtt{a}_{\mathtt{max}}$ scenarios. For all cases our method improves system performance in L_1, L_2 and L_∞ norms.

Effect of horizon length

In Figure 2.12 we study the effect of optimizing the reference trajectory with a receding horizon strategy on the first stage optimization. We observe that the trajectory time is the smallest for the one-shot optimization, and increases with a reduction of the horizon length. The increase is more pronounced for the Airfoil test case, which can be traced at higher speeds compared to the Letters test case. The computation time is also the smallest for the one-shot optimization. Short horizons are fast to compute, but require the largest number of optimization problems to be solved. The maximum computation time is reached for long horizons, where each optimization problem takes some time to compute and many are required.

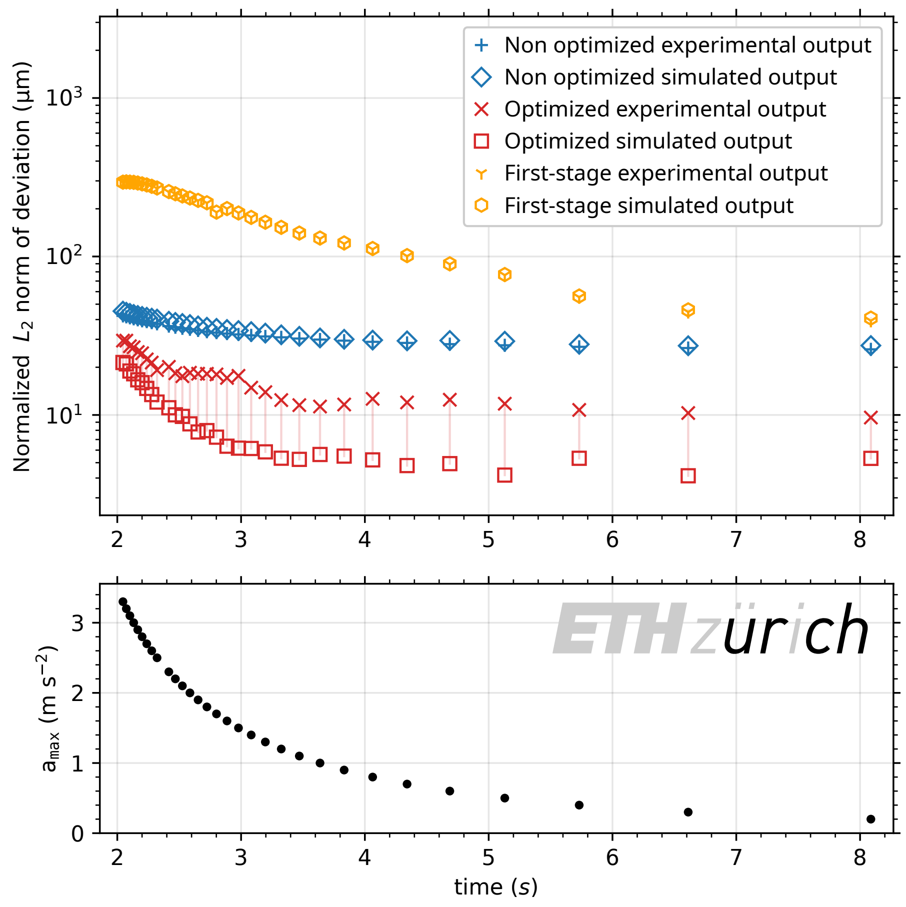


Figure 2.11: Precision-speed trade-off curve for the letters test case. The top panel shows the normalized L_2 deviation of the outputs from the target geometry. The bottom panel shows the total time to perform the trajectory and the $\mathtt{a}_{\mathtt{max}}$ used in optimization. To determine the performance at a given acceleration value, one can use the lower panel to determine the time needed to complete the trajectory for a given acceleration, then retrieve the deviation for the corresponding time from the top plot.

2.6 Discussion

In this Chapter we proposed a method that improves the precision vs. productivity trade-off of a PMS. We use models built exclusively from experimental data, and only modify the input trajectory provided to the closed loop control system. Experimental data obtained for shapes outside of the training dataset corroborates simulation results and shows that the method can significantly improve system performance, reliably shifting the precision vs. productivity trade-off curve across a wide range of operating conditions. This is accomplished by exploiting variable speeds (possible since the full trajectory is optimized in one go), and error compensation while respecting system operational constraints.

	$a_{max} = 1.0\,m\,s^{-2}$				$a_{max} = 3.0\,m\,s^{-2}$			
Shape	time	L_1	L_2	L_∞	time	L_1	L_2	L_∞
–	s	%	%	%	s	%	%	%
Letter u	1.052	52.7	60.0	63.0	0.618	36.4	38.7	46.1
Letter r	0.807	58.9	64.3	75.1	0.473	32.8	29.7	42.9
Letter c	0.807	80.0	78.5	74.4	0.473	40.2	38.6	54.3
Letter h	0.971	26.6	32.8	31.0	0.568	22.6	27.0	43.1
airfoil	1.322	73.5	69.0	64.9	1.024	75.0	63.0	28.1

Table 2.2: Percent improvement in accuracy after applying the proposed input trajectory design method compared to the default control without input trajectory optimization. Experimental results for different shapes and a_{max}.

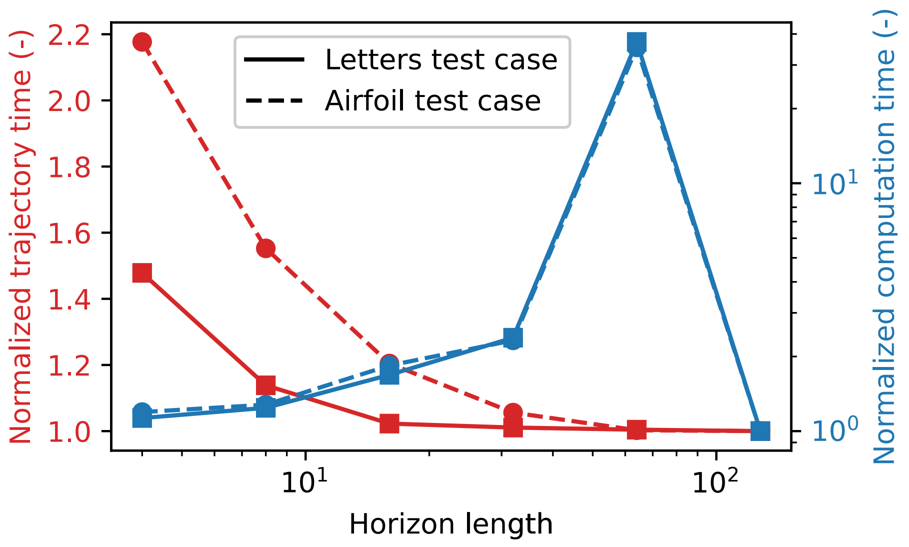


Figure 2.12: Trajectory time and computation time of the first stage as function of the horizon length using a receding horizon strategy. Both times are normalized by the one-shot optimization case, the right-most points where the horizon is equal to the number of points used in the discretization $N = 128$. In this study the tolerance band is set to 200 μm.

Future work will focus on reducing the computational load of the offline input trajectory optimization and further increasing the ANN complexity to improve the prediction accuracy.

CHAPTER 3

Optimization-Based iterative learning control

ILC is used in repetitive tasks to improve performance over iterations by learning from previous trials. In ILC, the control input is updated between iterations using the measured error, which is shown to ensure monotonic convergence to an approximate fixed point of the original problem under various assumptions [20, 19, 39, 40]. An important challenge with Iterative Learning Control (ILC) is to ensure convergence and constraint satisfaction, which is especially difficult when the underlying system is nonlinear.

In this chapter we extend existing OB-ILC methods to nonlinear system dynamics. Specifically, our goal is to leverage approximate process models to pose an optimization problem that we iteratively solve using the underlying nonlinear system while ensuring constraint satisfaction.

Chapter Outline

This Chapter is organized as follows. In Section 3.1 we summarise the state-of-the-art and contributions, in Section 3.2 we present the problem setting and the control approach, in Section 3.3 we present the optimization problem and the proposed OB-ILC approach, in Section 3.4 we present a detailed case study in precision motion control, in Sections 3.5 and 3.6 we present simulation and experimental results respectively, and in Section 3.7 we provide closing remarks with potential future directions.

Notation

We denote the Jacobian by ∇ and the Jacobian along a certain direction d by ∇_d. Similarly ∇^2 and ∇_d^2 are the Hessian and the Hessian along the direction d respectively. ∂^n denotes the discrete derivative of order n, defined by $\partial^n x = (\partial^{n-1}x(i+1) - \partial^{n-1}x(i))/\Delta t$, $\partial^0 x = x$, where Δt is the discrete time interval, between $x(i)$ and $x(i+1)$.

3.1 State-of-the-art

OB-ILC methods have been proposed in the literature to systematically study iteration-wise error dynamics and constraint satisfaction. Robust optimization-based methods [41, 39], interior point-based OB-ILC [42], and norm-optimal ILC methods [43, 19, 18, 44] are some of the common approaches in the literature for linear systems. While some of the works consider model mismatch and process constraints jointly, many of the existing works do not provide robust constraint satisfaction, and convergence results in the presence of measurement noise. Recently, OB-ILC has been extended to handle process constraints in linear processes while accounting for noise and model mismatch during all iterations [20].

A survey of the ILC method for nonlinear dynamics is given in [45]. In [40] robust convergence for a class of nonlinear systems is given, while a neural network-based nonlinear ILC method is presented in [46]. Linearization-based OB-ILC methods for nonlinear systems are studied in [47, 48]. Variants of Newton-based methods are used for nonlinear ILC problems [49, 50, 51]. In [52] a zeroth-order ILC for nonlinear processes is proposed. It requires solving a nonlinear program after each iteration and difficult-to-verify properties with approximate sensitivities.

ILC is used extensively in motion tracking problems and has been shown to improve the performance of gantry systems [53], wafer stages [42], precision motion systems [18, 19] and various related applications [54, 55].

Contributions

In this Chapter, we propose a novel nonlinear OB-ILC method based on the Sequential Quadratic Programming (SQP) method for nonconvex optimization [56]. Specifically, we consider model mismatch and constraints to form approximate subproblems, which are solved by using measurements from the nonlinear process. The main contribution of this Chapter is a nonlinear OB-ILC scheme based on the SQP framework, that requires solving convex quadratic subproblems after each trial and can handle constraints.

We illustrate our proposed OB-ILC method for nonlinear dynamics on a precision motion tracking problem. We present a detailed case study using a high-fidelity simulator of a precision motion system, and we compare the achieved tracking accuracy by using models with different fidelity (linear and neural network-based).

3.2 Problem Statement

We consider a noisy nonlinear repetitive process of the form

$$y = f(u) + w, \tag{3.1}$$

where $u \in \mathbb{R}^n$ is the input, $y \in \mathbb{R}^m$ is the output, $f : \mathbb{R}^n \to \mathbb{R}^m$ is the system response (input/output map), and $w \in \mathbb{R}^m$ is a non-repeating disturbance, assumed to be zero-mean. We focus on the response of a dynamical system over a finite interval where u and y define input/output trajectories of the underlying nonlinear system. Therefore, we have $u = (u(1), u(2), \ldots, u(N))$ for an input trajectory of N time steps, and similarly for y. For example, the input u could be a trajectory of actuator commands or a reference trajectory tracked by a low-level feedback controller, and the output y is the actual trajectory traced by the system. In Section 3.4 we show a case study with this configuration. The input and output must satisfy the constraints $u \in \mathcal{U} \subset \mathbb{R}^n$, $y \in \mathcal{Y} \subset \mathbb{R}^m$. For example, in a motion tracking problem, the sets $\mathcal{U}$ and $\mathcal{Y}$ may encode limits on actuation velocity, and acceleration.

Our control objective is to choose the input u such that the output y tracks a target trajectory Ξ as closely as possible. The ILC approach designs a learning policy $\pi = (x, \mathcal{T}, q)$ of the form

$$x_{k+1} = \mathcal{T}(x_k, y_k), \tag{3.2a}$$
$$u_k = q(x_k). \tag{3.2b}$$

where x is the internal state of the policy, $\mathcal{T}$ is the update function, and q is an output function that recovers the control input from x. The goal is to design x, $\mathcal{T}$, and q such that the (iteration domain) closed-loop system

$$u_k = q(x_k), \tag{3.3a}$$
$$y_k = f(u_k) + w_k, \tag{3.3b}$$
$$x_{k+1} = \mathcal{T}(x_k, y_k), \tag{3.3c}$$

converges to some $y_k \approx \Xi$, with $u \in \mathcal{U}$ and $y \in \mathcal{Y}$. Due to the dynamics and constraints y_k will, in general, not reach Ξ exactly. Subscript k indicates the iteration index of the ILC throughout the rest of the Chapter.

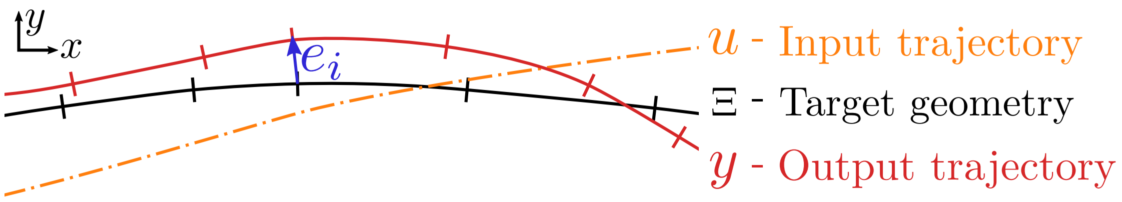


Figure 3.1: A realization of Target, Input, and Output trajectories. The error is the distance between the output and the target. In the figure we plot e_i, the error for time step i. This is not to be confused with the ILC iteration number k.

Here we propose to design the policy $(x, \mathcal{T}, q)$ using SQP. We assume access to a model that is used to derive gradient and hessian information about $f(u)$. In subsequent Sections we provide the details of the individual components in Figure 3.2. We initialize with a feasible input and take an SQP step after each experiment to evaluate the ILC policy (3.3) using the approximate model of the system, measurements, and past inputs. The objective of the approach is to minimize the output tracking error with respect to a target trajectory, illustrated in Figure 3.1.

3.3 Optimization Based ILC for non-linear systems

The process (3.1) is assumed to be nonlinear and unknown, as well its gradient and Hessian. We assume it to be possible to evaluate (3.1), by running an experiment, and to have access to a model of the process either from first principles, experimental data or combinations of both, from which gradient and Hessian models can be derived. Our goal is to efficiently improve the quality of the output trajectory by leveraging the model information to reduce the number of experiments needed.

Driving the output of the repetitive process to the target geometry can be encoded as an optimization problem, where the cost function encodes the objective of tracking the target trajectory Ξ and the limits on input and output are framed as constraints. The challenge is to incorporate process data into the optimization problem. We can use tools from optimization theory to solve the nontrivial nonlinear constrained ILC problem. In this work, we focus on adapting the SQP algorithm to compute the ILC updates.

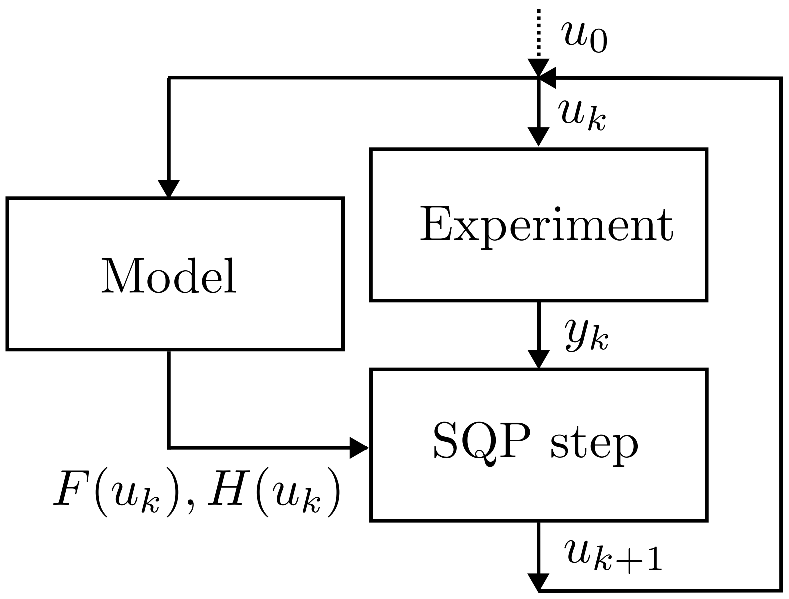


Figure 3.2: Scheme of the proposed approach. We start with some initial input u_0. The output is obtained from an experiment. The model is used for the gradient F and Hessian H information in the SQP step k, which is used for evaluating the next input $u(k+1)$.

We encode our control objective in the following optimization problem

$$\min_z \quad J(z) \tag{3.4a}$$

s.t.

$$h(z) \equiv p - f(u) = 0 \ , \tag{3.4b}$$

$$g(z) \leq 0, \tag{3.4c}$$

where $z = (u, p), \mathcal{U} \times \mathcal{Y} = \{z \,|\, g(z) \leq 0\}$, $J(z)$ is a function measuring the distance between the output and target trajectory (e.g., $J(z) = \|p - \Xi\|$), and p is an optimization variable constrained by (3.4b) to be equal to the noise-free system response $f(u)$.

This is a nonlinear program, that we aim to solve using SQP. In SQP, we construct and solve a sequence of QPs that eventually converge to a solution of the original nonlinear problem. The Lagrangian associated with (3.4) is

$$\mathcal{L}(z, \lambda, \sigma) = J(z) + \lambda^T h(z) + \sigma^T g(z), \tag{3.5}$$

and the standard quadratic subproblem is

$$\min_{\Delta z} \quad \frac{1}{2}\Delta z^{\mathsf{T}} B \Delta z + \nabla J(z)^{\mathsf{T}} \Delta z \tag{3.6a}$$

s.t.

$$\nabla h(z)^{\mathsf{T}} \Delta z + h(z) = 0, \tag{3.6b}$$

$$\nabla g(z)^{\mathsf{T}} \Delta z + g(z) \leq 0, \tag{3.6c}$$

where $B \approx \nabla^2 \mathcal{L}(z, \lambda, \sigma)$ is an approximation for the Hessian of the Lagrangian (3.5). We can use the primal-dual solution $(\Delta z^*, \lambda^*, \sigma^*)$ of the subproblem (3.6) to construct the SQP-based ILC policy

$$\mathcal{T}(z, \lambda, \sigma) = \begin{bmatrix} z + \Delta z^* \\ \lambda^* \\ \sigma^* \end{bmatrix} \tag{3.7}$$

It is known that the iteration

$$x_{k+1} = \mathcal{T}(x_k) \tag{3.8}$$

where $x = (z, \lambda, \sigma)$ converges locally at a quadratic rate to minimizers of the original problem (3.4) that satisfy appropriate regularity conditions (e.g., the linear independence constraint qualification and strong second order sufficient conditions [56]).

We modify the SQP algorithm to design an ILC policy by incorporating data. We assume that $f(u)$ is unknown but can be obtained for any given u_k by running an experiment leading to the output data $y_k = f(u_k) + w_k$ is corrupted by noise w_k. Further, we assume to have access to approximations of the Jacobian and Hessian of the process derived from a system model

$$F(u_k) \approx \nabla f(u_k) \tag{3.9a}$$

$$H(u_k) \approx \nabla^2 f(u_k) \tag{3.9b}$$

We adapt (3.6a) to deal with the fact that we do not have direct access to $f(u)$, replacing $h(z)$ with $p - y_k$, and $\nabla h = [I \quad -\nabla f]$ with $[I \quad -F]$.

$$\min_{\Delta u, \Delta p} \quad \frac{1}{2}\begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} R & S \\ S^{\mathsf{T}} & Q \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix} + \begin{bmatrix} \nabla_u J \\ \nabla_p J \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix} \tag{3.10a}$$

s.t.

$$\Delta p = F(u_k)\Delta u + (y_k - p_k), \tag{3.10b}$$

$$u_k + \Delta u \in \mathcal{U}, \tag{3.10c}$$

$$p_k + \Delta p \in \mathcal{Y}, \tag{3.10d}$$

where $R \approx \nabla_u^2 \mathcal{L}$, $S \approx \nabla_{y,u}^2 \mathcal{L}$, and $Q \approx \nabla_y^2 \mathcal{L}$. The first term in (3.10b) imposes a linearized version of the dynamic constraint, while the second term corrects the local estimate of the system output given the new measurement y_k. Additionally, R is constructed using the Hessian approximation $H(u_k)$.

Finally, we introduce a step size η_k as damping factor for the algorithm's iterates to encourage convergence; below we use a diminishing step size and provide simulation results on the effect of the decay rate. The resulting OB-ILC policy

$$\mathcal{T}(z, \lambda, \sigma, k) = \begin{bmatrix} z_k + \eta_k \Delta z_k^* \\ \lambda^* \\ \sigma^* \end{bmatrix} \tag{3.11}$$

where $(\Delta z^*, \lambda^*, \sigma^*)$ is the solution to the modified data-dependant subproblem (3.10) and $\Delta z^* = (\Delta u^*, \Delta p^*)$. Note that constraint (3.10b) explicitly incorporates data from the real unknown system into our SQP algorithm to compensate for model mismatch and improve robustness.

The overall algorithm is outlined in Algorithm 1. The ILC loop is terminated when $\|\Delta z^*\|$ goes below a certain threshold, or when a maximum number of iterations is reached. The SQP steps can be solved using a standard Quadratic Program (QP) solver. In our implementation, we use OSQP [57] via the Casadi [38] interface for Python.

Algorithm 1 OB-ILC with SQP steps

```
1: $u \leftarrow u_0$                                   ▷ Initialization
2: **repeat**
3:     $y_k \leftarrow f(u_k) + w_k$                    ▷ Measurement
4:     $\Delta z_k^* \leftarrow$ Solution of (3.10)
5:     $z_{k+1} \leftarrow z_k + \eta_k \Delta z_k^*$   ▷ Update
6:     $k \leftarrow k + 1$
7: **until** termination criteria are met
```

3.4 Case study with a 2D precision motion system

In this Section, we provide a detailed case study in precision tracking via a non-linear high-fidelity simulator of a physical system.

The system

As a case study we use a 2-axis high precision motion system depicted in Figure 1.1a. This system contains an internal closed-loop controller, it takes reference trajectories as inputs and produces tool-tip trajectories as outputs. In this work, we use three different models to instantiate Algorithm 1. All model the system response in discrete time, with a sample rate of 400 Hz. For each case, we report σ, the standard deviation of the prediction error of the model, for input trajectories with acceleration up to $3\,\mathrm{m\,s^{-2}}$, when compared to experimental data. The three models are summarized in Table 3.1. For a more detailed description of the system see Section 1.2, and for details on the model architecture see Section 2.3.

Model	Description	σ
LM	A discrete-time linear model, in a state space lifted representation.	$236.4\,\mu\mathrm{m}$
NL1	A nonlinear ANN model, with an input layer capturing 200 ms of input history, and LeakyReLu activation functions.	$16.50\,\mu\mathrm{m}$
NL2	A nonlinear ANN model, with an input layer capturing 500 ms of input history, and LeakyReLu activation functions. This model is, for the purposes of simulations, considered to be the ground truth.	$11.27\,\mu\mathrm{m}$

Table 3.1: The three different models of the system used for optimization and simulation. σ is the standard deviation of the prediction error of the model, for input trajectories with acceleration up to $3\,\mathrm{m\,s^{-2}}$, when compared to experimental data.

In the following numerical results, we use either LM or NL1 to derive the gradient (3.9a) and Hessian (3.9b) information. Since the structure of both models is known, one can take symbolic derivatives of the output in respect to the input to obtain the gradient and Hessian. The quality of the derivative information is not directly affected by the non-repeating disturbance, that is only used to determine the point where the derivatives are computed. Due to the structure of both models, the Hessian evaluates to zero. In all simulations, the model NL2 is used in lieu of the true system, i.e., for the evaluations of $f(u)$ but not its gradients.

Cost and constraints

The optimization problem (3.4) used in the case study is

$$\min_{u,p} \quad J(u,p) = \sum_{i=0}^{N} \|p(i) - \Xi(i)\|^2_{Q_a} + \|\partial^2 u(i)\|^2_{R_a} \tag{3.12a}$$

s.t.

$$p - f(u) = 0, \tag{3.12b}$$

$$p(i) \in \mathcal{W} \ , \quad i = 1, \dots, N, \tag{3.12c}$$

$$|\partial u(i)| \leq \mathtt{v_{max}}, \quad i = 1, \dots, N-2, \tag{3.12d}$$

$$|\partial^2 u(i)| \leq \mathtt{a_{max}}, \quad i = 1, \dots, N-3, \tag{3.12e}$$

$$|\partial^3 u(i)| \leq \mathtt{j_{max}}, \quad i = 1, \dots, N-4, \tag{3.12f}$$

where N is the number of points in the target trajectory $\Xi = \{\Xi_i\}_{i=1}^{N} \subseteq \mathbb{R}^2$, $p = \{p_i\}_{i=1}^{N} \subseteq \mathbb{R}^2$ and $\mathcal{W} \subseteq \mathbb{R}^2$ is the workspace. The first term of the cost function penalizes deviations of the output with respect to the target geometry, while the second term regularizes the input by penalizing the acceleration. We use $Q_a = 10^6 I$, $R_a = 10^{-2} I$ to reflect the different order of magnitude of the input acceleration and output deviation. For the constraints we use $\mathtt{v_{max}} = 2$, $\mathtt{a_{max}} = 2$, $\mathtt{j_{max}} = 500$, derived from the physical limits of the machine. The number of points N in the target trajectory (see Figure 3.7) is 314, which corresponds to 0.785 s time discretization between sample points given the sample rate of 400 Hz. As the target geometry we use the outline of the letter "r" from the ETH Zurich logo, shown in the inset of Figure 3.7. The values for the velocity, acceleration, and jerk limits are derived from the physical limits of the machine.

3.5 Simulation Results

We present results using LM and NL1 for evaluating the approximations in (3.9) and discuss the effect of the initial input u_0 and the decay rate of the form $\eta_k = \eta_0 k^{-c}$ on ILC performance.

Effect of the model

The results presented in Figure 3.3 and Figure 3.4 show that for the iterations taken using gradient information from both the LM and the NL1, the error converges to a value of the same order of magnitude. However, we see LM converge to a solution with lower deviation and at a faster rate compared to NL1. This result is at first surprising given that the prediction error of the LM is one order of magnitude higher than NL1. We note however that NL1 is built with LeakyReLu

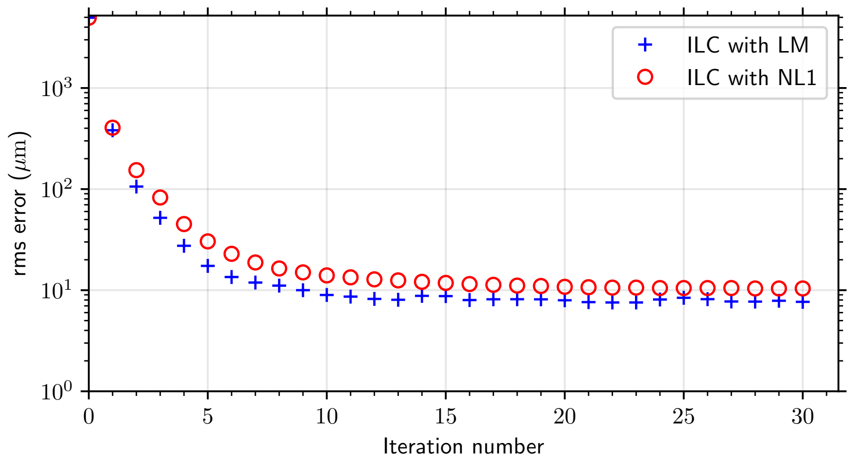


Figure 3.3: Simulation results. Output deviation as a function of the iteration number for SQP steps using the derivative information from either the LM or NL1 models, evaluated with NL2 as ground truth. The initial condition for both series is the target geometry traced at constant velocity. Step size updated with $c = 0.5$.

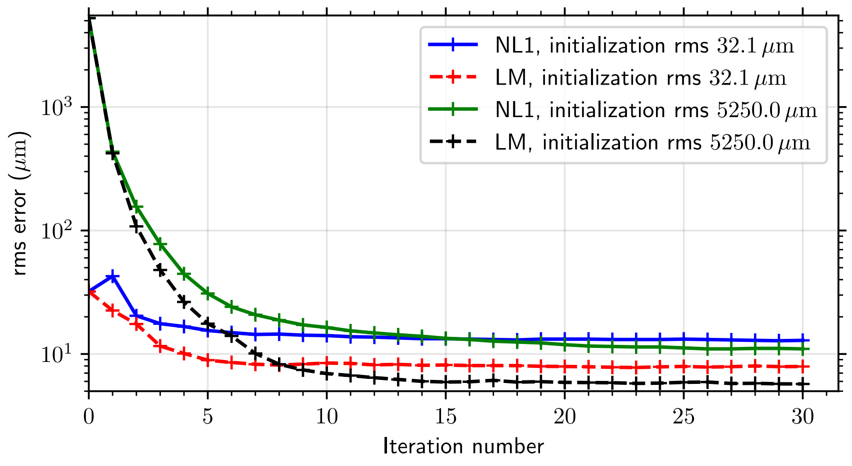


Figure 3.4: Simulation results. Output deviation as a function of the iteration number for SQP steps using the derivative information from either the LM or NL1 models, evaluated with NL2 as ground truth. Two different initialization are provided for each case. The initialization starting with $rms = 5250.0\,\mu\text{m}$ is the solution of an optimal time problem with dynamics modeled with LM, where the velocity along the path is adjusted, for example negotiating intricate features slower and straight segments faster. The initialization starting with $rms = 32.1\,\mu\text{m}$ takes the optimal time solution as a starting point and solves a global optimization problem with dynamics modeled by NL1 and additional constraints on deviation, velocity, and acceleration. Further details can be found in [8]. Step size $\eta_k = \eta_0 k^{-c}$ with $c = 0.5$.

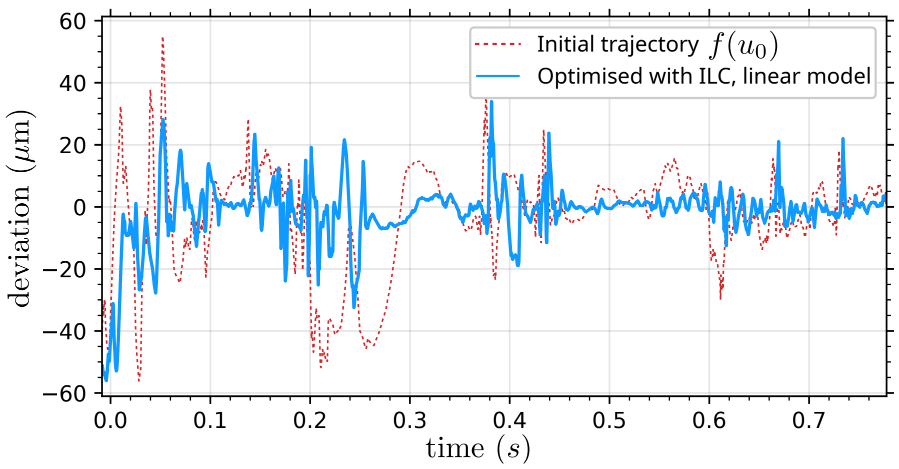


Figure 3.5: Simulation results. Output error as a function of time, after 20 ILC steps using the LM, and evaluated with NL2.

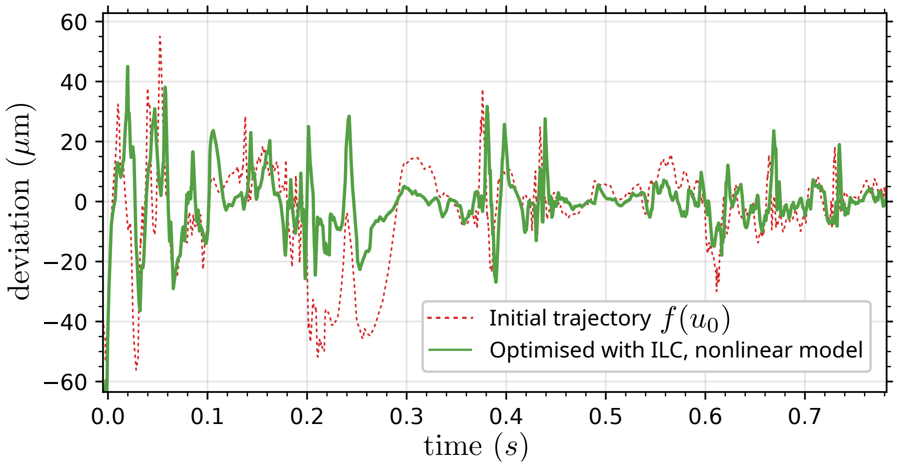


Figure 3.6: Simulation results. Output error as a function of time, after 20 ILC steps using the NL1, and evaluated with NL2.

activation functions, and thus its gradient is piecewise constant and discontinuous. A visualization of the nonlinear model response is presented in Figure 3.10. Despite its higher prediction error, the structure of the LM is found to provide more accurate gradient information. Different shapes and tunings of the cost function show mostly similar trends of the ILC loop using the linear and nonlinear models (data not shown). We have observed that the system is only mildly nonlinear, and since the ILC step relies on measurements that do not depend on the models used, the fidelity of the approximations (3.9) is apparently not of critical importance. We hypothesize that a system with more pronounced nonlinearities would experience faster convergence with ILC steps relying on the nonlinear model for the approximations. In Figure 3.5 and Figure 3.6, we show the output deviation as a

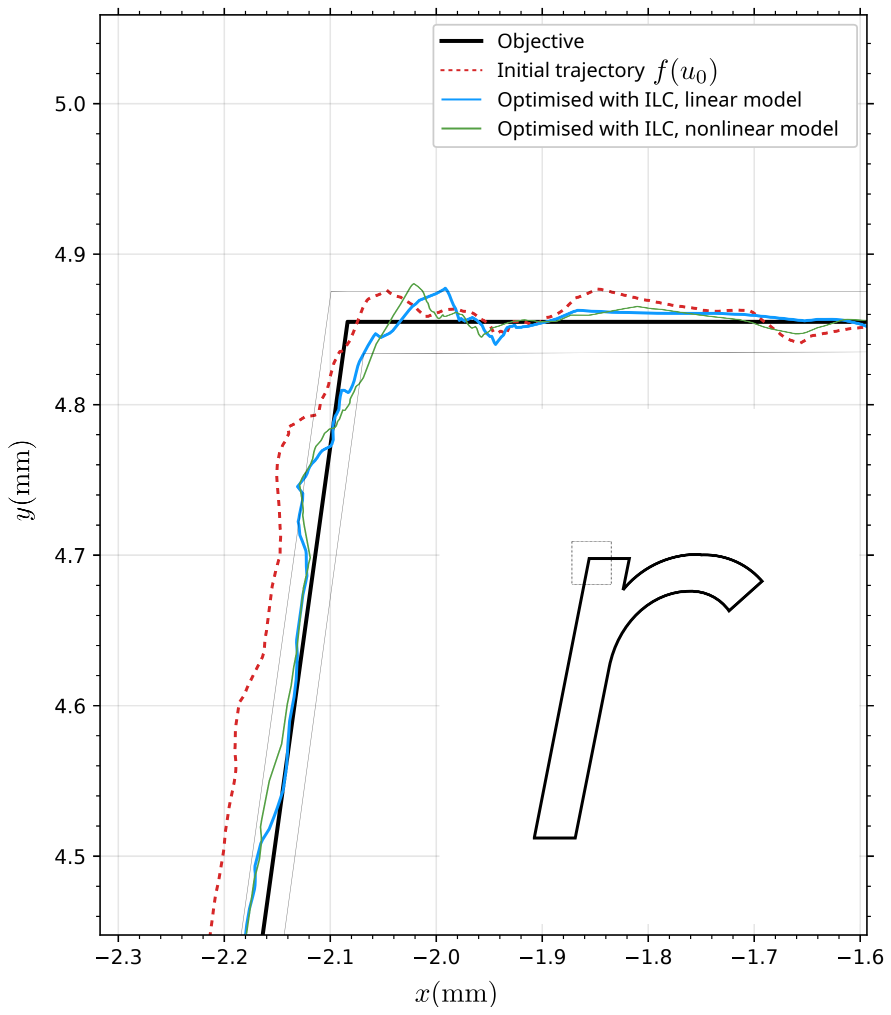


Figure 3.7: Simulation results. Detail view of trajectories before and after 20 ILC iterations. The figure shows a 20 μm band around the target geometry as a visual aid.

function of time before and after 20 steps of the proposed method. The deviations are computed between the output and the target trajectory depicted in Figure 3.7, where the output trajectories are plotted in $x - y$ coordinates.

Effect of initialization

In Figure 3.4, we show the results for two different initial conditions. The results illustrate that since the underlying problem is nonlinear it is possible to be in a local minimum and not achieve a better solution, depending on the initial conditions.

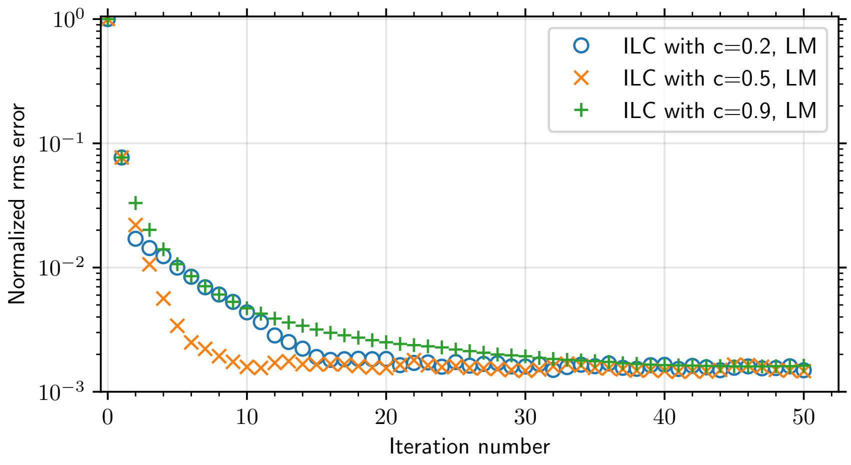


Figure 3.8: Simulation results: Normalized error (rms) as a function of the iteration number for step size $\eta_k = \eta_0 k^{-c}$ with the LM, and an initialization with rms = 5250.0 μm.

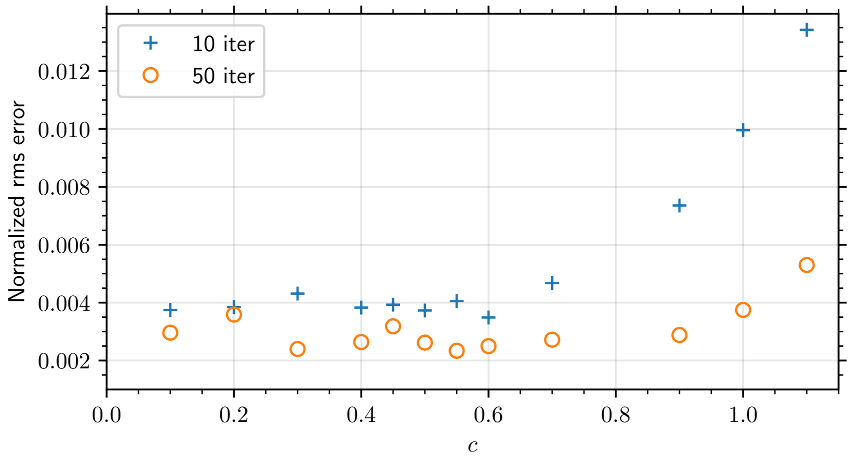


Figure 3.9: Simulation results. Output error (rms) after 10 and 50 iterations for different step size rates $\eta_k = \eta_0 k^{-c}$ with the LM, and an initialization with rms = 5250.0 μm.

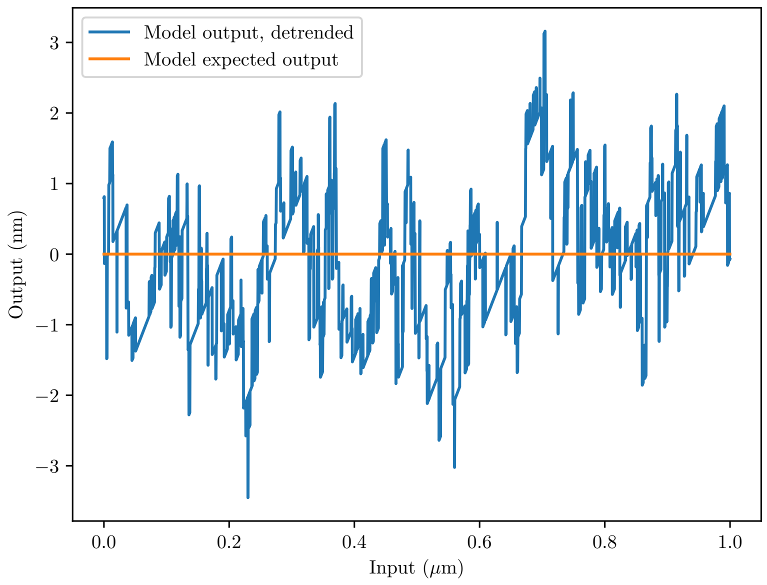


Figure 3.10: Output prediction of the NL1 for the x axis. For each point, the input is set to a linear sequence $u = \{l, 2l, \ldots, 80l\}$ m, with $l \in [1, 1 + 5 \times 10^{-6}]$ m. The horizontal axis shows the last value of the offset-free input vector $80l$ m. The vertical axis shows the output deviation from the linear least squares fit of the model output.

Effect of step size

Next, we study the effect of step size on convergence behavior. We take $\eta_k = \eta_0 k^{-c}$ and compare the convergence. We show the error trajectories for $c = \{0.2, 0.5, 0.9\}$ (Figure 3.8). In Figure 3.9 we plot the error after 10 and 50 iterations for different values of c. We observe that for c between 0.3 and 0.6, we obtain fast decreases in the error without compromising the value at a steady state.

3.6 Experimental Results

Figure 3.11 shows experimental results for different initializations of the ILC method, using the LM as the gradient model. For all four cases, the target path is the letter "r" from the letters test case. The non-optimized case tracks this path with constant path speed. In contrast, the remaining cases track the projection of the optimal time trajectory found with the first-stage optimization into the target path, as described in Sections 2.4 and 2.4. The target, as well as input trajectories, all have, by construction, the same total trajectory time. The results show that using the non-optimized target, sampled at constant speed, as an input trajectory

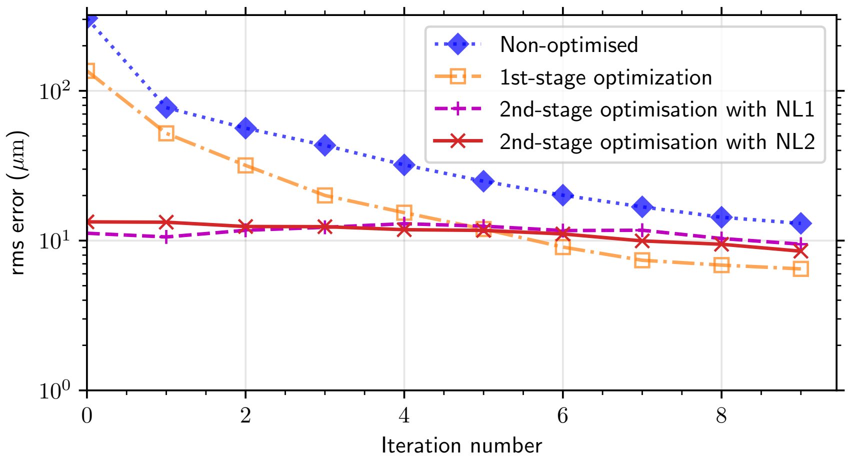


Figure 3.11: Experimental results. ILC steps using gradient information from the LM. Step size rate $\eta_k = \eta_0 k^{-0.5}$. The target shape is the letter "r" from the letters test case, performed in 0.783 s.The plot shows the evolution of the error for different initializations of the reference trajectory as a function of the iteration number. The non-optimized case tracks the target shape at constant speed. The first-stage optimization case starts from the optimization result as explained in Section 2.4, and the second-stage optimization case from the optimization result as explained in Section 2.4. For this test case, it was observed that after nine iterations the error varies at a much smaller rate.

result in a large (305.6 μm rms) error, that can be compensated for with nine iterations of the OB-ILC method to a value of 13.0 μm. Initializing the method with the result of the first-stage optimization starts also with a large (136.0 μm) error, that can be driven down to 6.5 μm. Finally, starting with an input trajectory that is the result of the second-stage optimization with either the NL1 or NL2 models, we get an error of 11.2 μm and 13.3 μm which are improved only slightly after nine iterations to 9.5 μm and 8.5 μm respectively.

Taken together this results indicate that the input trajectories designed with the second-stage optimization explore to a large degree the capabilities of the machine. Due to model mismatch, they can however be further improved by taking additional experimental data. A smaller error can be obtained from a different initialization. We hypothesize that the solution of second-stage optimization is a a local minimum of the nonlinear optimization problem, and that this property is inherited by the iterates of the OB-ILC method. The first-stage optimization initialization reaches a lower error value compared to the non-optimized version. This is to be expected, since the target trajectory in this case is optimized for path speed, slowing down for corners and other intricate features and speeding up for

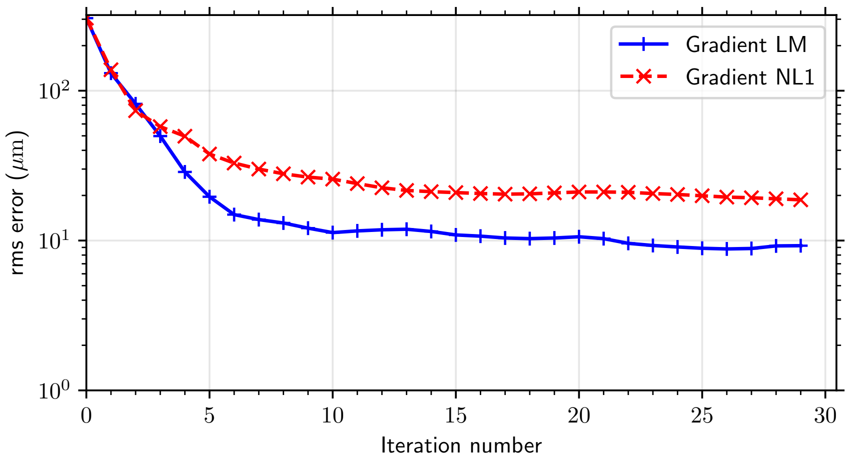


Figure 3.12: Experimental results. ILC steps using gradient information from the LM or NL1 gradient models. Step size rate $\eta_k = \eta_0 k^{-0.5}$. The target shape is the letter “r” from the letters test case, performed in 0.548 s.

low curvate segments.

Figure 3.12 shows experimental results comparing the OB-ILC iterates for two different gradient models. As seen previously in simulation, we observe that using the LM gradient model results consistently in lower errors.

3.7 Discussion

We propose an algorithm for optimization-based ILC of nonlinear systems based on the SQP framework. We formulate the ideal optimization problem and find approximate solutions using models of the true plant. We illustrate the performance of the algorithm on a precision motion control simulation study using a high-fidelity simulator. The results show that our method works well under various parameter tunings and we are able to show significant improvement in the tracking error over baseline initial iterations. Contrary to our initial assumptions, the ILC using gradient information from a simpler linear model performs best than the one taken from the more accurate nonlinear one. We observe that the performance of a model depends on its application.

CHAPTER 4

Quadcopter-based Volume Estimation in Indoor Environments

Autonomous robotic platforms are increasingly used for data collection, for example in structural inspection [58], agriculture surveillance, search and rescue, and industrial environments. In industrial warehouses it is common to store raw materials as stockpiles, and determining the current amount of material in stock is of paramount importance for logistics. However, to the best of our knowledge, the task of estimating the volume with an automated robotic platform in an indoor environment has not been addressed. On the contrary, current business practice is to take differential measurements of the volume added or removed, which is prone to drift over time, and on periodic inspections from experts, which are costly and inaccurate. A quadcopter equipped with an adequate suite of sensors could be used for this purpose, since it can fly above the pile of bulk material taking advantage of its maneuverability to take measurements from poses otherwise unreachable while avoiding obstacles in cluttered environments. Estimating the volume using autonomous quadcoptor in indoor environment imposes several requirements, e.g.: 1. indoor localization, where the location uncertainty depends on the drone state, 2. accurate measurements in environments with uneven light and dust, 3. an efficient surface reconstruction method, able to cope with large amounts of data, and 4. path planning, due to the limited time budget to fly the drone and the presence of obstacles.

Chapter Outline

This chapter is organized as follows. In Section 4.1 we summarise the state-of-the-art and contributions, in Section 4.2 we present the problem statement, in Section 4.3 we derive detailed measurement models for the localization system and LiDaR sensor, in Section 4.4 we propose a method for estimating the volume, in Section 4.5 we propose a greedy algorithm to approximate a solution of the original problem.

4.1 State-of-the-art

Localizing a mobile platform in a GPS-deprived environment with a known map can be achieved using information obtained from dead-reckoning, infrared, radio or sound-based distance measurements, visual information using a motion capture system, or from an onboard camera. In this work localization is inferred from an onboard camera, detecting a set of fixed, previously mapped features, given the lower overall system cost, flexibility and taken into account the accuracy requirements.

Surface reconstruction can be performed from images, with photogrammetry methods such as Structure from Motion (SfM) [59]. This methods are however problematic for objects with homogeneous surfaces or improper lighting. An alternative way is via active laser scanners (LiDAR) which project a laser beam and measure the time of flight of the reflected light. This sensors are precise and are not affected by the effect of scale uncertainty present in vision based measurements. Furthermore 2D LiDAR systems are lighter than their 3D counterparts, allowing the use of more maneuverable quadrotors. There are commercial examples of drone-based solutions that use this method in outdoor environments [60]. The reconstruction quality depends to a large degree on the availability and quality of measurements. Classic approaches for quality-driven and automated 3D scanning use volumetric [61] and Poisson mesh-based metric. Algorithms also been proposed for multi-view stereo reconstruction [62], defining heuristics to decide on the utility of the next measurements and optimize set viewpoints based on initial scans. Alternatively, 2D laser scanner has been used for 3D mapping in [63, 64, 65], often mounting the 2D scanner on a rotating motor to emulate 3D LiDAR properties.

In this Chapter, we consider the problem of estimating the volume of material within a given domain using a drone mounted 2D LiDAR unit operating in an indoor environment, leveraging information about the surface. This setting has challenges unique to GPS denied environments, notably the uncertainty in the localization depends on the position of the drone, which must follow trajectories that keep enough features in view to maintain localization accuracy. In [66] the authors consider visual inspection of ship hulls using underwater vehicles. In [67] the authors consider inspection of 3D object, and [68] deals with a similar problem setting, but localization uncertainty is not taken into account. This work differs since it deals with indoor environments and carefully considers the uncertainty of the measurements.

Contributions

Our contributions are threefold: 1. we derive measurement models and uncertainty estimates for the camera-based localization scheme and the LiDaR system used to measure the surface; 2. we propose a scalable methodology for estimating the volume of material based on LiDaR measurements and qualifying the uncertainty of our estimate; 3. we propose a preliminary informative path planning method that greedily minimizes the uncertainty in the volume estimate.

4.2 Problem Statement

We consider the problem of estimating the volume of a pile of bulk material inside a region of interest using a quadrotor-based mobile sensor. Let $h(x) = h(x_1, x_2)$ be the true surface function of the height of the pile, defined in the domain of interest $\mathcal{D} = [x_1^-, x_1^+] \times [x_2^-, x_2^+]$. The volume of the pile is

$$\mathcal{V} = \iint_{\mathcal{D}} h(x)dx, \tag{4.1}$$

and the dynamics of the quadcopter are given by

$$\dot{\chi}_{\text{full}}(t) = f(\chi_{\text{full}}(t), u(t)), \tag{4.2a}$$

$$u(t) = g(r(t), \hat{\chi}_{\text{full}}(t)) \tag{4.2b}$$

where $\chi_{\text{full}}(t) = (p, \theta, v, \omega, b) : [0, T] \to \mathbb{R}^{13}$ is the drone state, consisting of its position p, orientation θ, velocity v and angular velocity ω, as well as the battery state of charge b, $u(t) : [0, T] \to \mathbb{R}^4$ are the rotor voltages, g is a feedback controller that stabilizes the system to a commanded position and yaw setpoint r, and $\hat{\chi}_{\text{full}}(t)$ is an estimate of the current state of the drone. In this work we assume that the low-level controller can follow the reference closely, such that $\chi_{\text{full}_i}(t) \approx r_i(t), i \in \{1, \dots, 4\}$ if the time derivatives of $r(t)$ up to order four (velocity, acceleration, jerk and snap) are within specified bounds, derived from actuator limits $\mathcal{U}$ [69].

The quadcopter is equipped with a 2D LiDaR, measuring the radial distance d_l from the sensor to the surface of the pile, that is used to estimate the volume of the pile based on a reconstruction of the surface. The drone is also equipped with a camera and computer vision algorithms that allow it to detect and identify

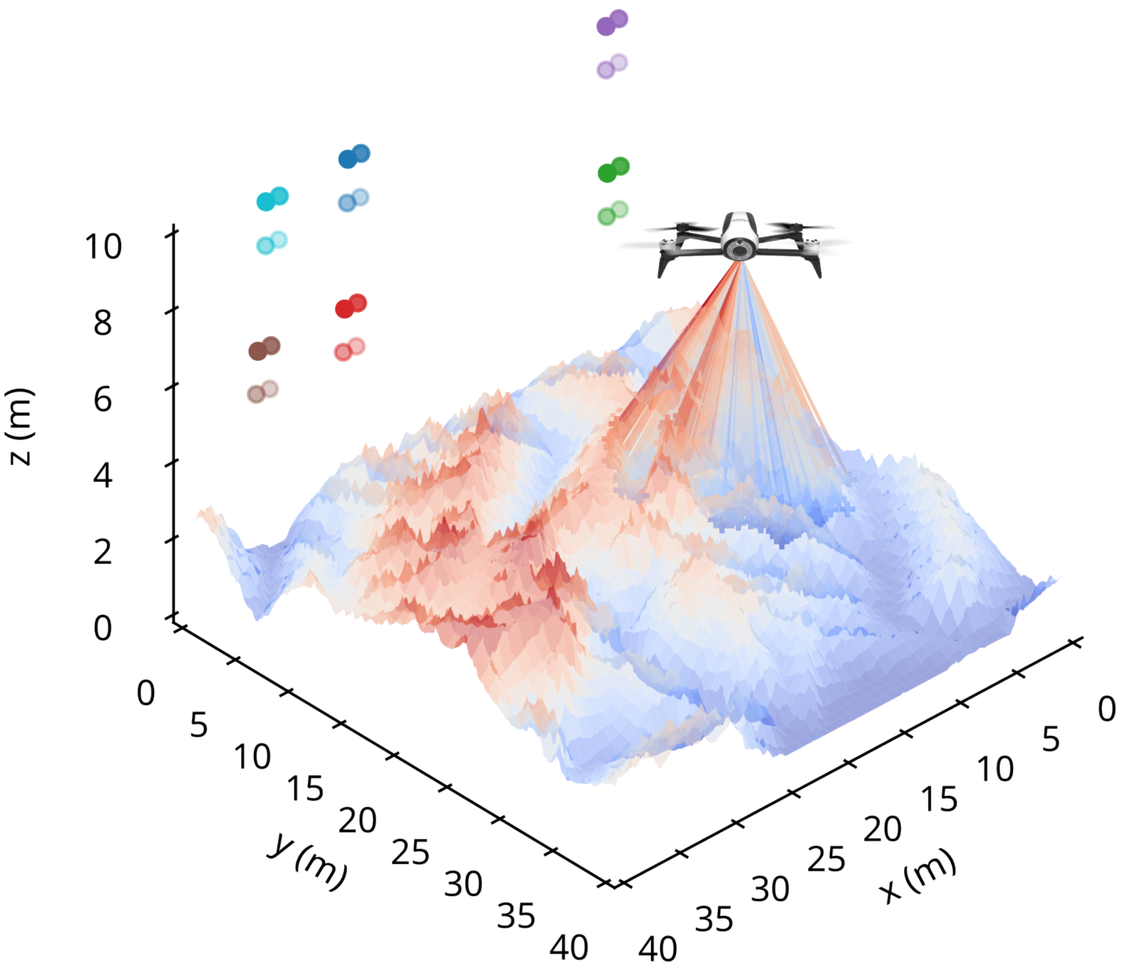


Figure 4.1: A sample of the scaled topographic data used in simulation. The color circles on the back plane are previously mapped visual features used for localization. The drone represented is not to scale.

features placed in the environment at known locations. These features are used to localize the drone which must keep a minimum number of features within its field of view at all times.

We have now all the ingredients needed to formulate our volume estimation problem. We aim to find a reference trajectory $r(t) : [0, T] \to \mathcal{C}_{\text{free}} \subseteq \mathbb{R}^3 \times [0, 2\pi[$, where T is the trajectory time and $\mathcal{C}_{\text{free}}$ is the set of 3D positions and yaw where the drone is allowed to fly and is able to detect enough features to localize itself, that minimizes the volume estimate uncertainty,

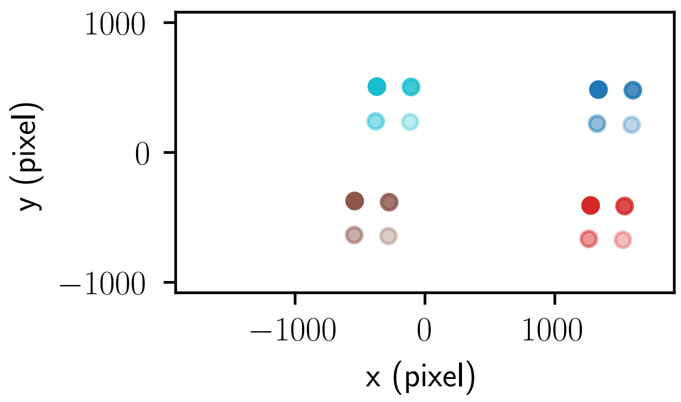


Figure 4.2: An example of the image in the camera plane, showing the detected points $\overline{P_{im}}$ which are used to infer localization.

$$
\begin{aligned}
\min_{r(t)} \quad & \mathbb{V}(\mathcal{V}(T)) && \text{(4.3a)}\\
\text{s.t.} \quad & \dot{\chi}_{\text{full}}(t) = f(\chi_{\text{full}}(t), u(t)), && \text{(4.3b)}\\
& u(t) = g(r(t), \hat{\chi}_{\text{full}}(t)), && \text{(4.3c)}\\
& \chi_{\text{full}}(t) \in \mathcal{C}_{\text{free}}, && \text{(4.3d)}\\
& u(t) \in \mathcal{U}, && \text{(4.3e)}
\end{aligned}
$$

where $\mathbb{V}$ is the variance operator. This is a hard problem as the uncertainty in the volume estimate is a consequence of uncertainty in the LiDaR measurements which in turn depends on the uncertainty in the camera-based localization system.

4.3 Localization

We are primarily interested in operations in indoor GPS denied environments. Our drone is equipped with an IMU and a front facing camera, which is used to localize the drone based on markers placed in known locations in the environment. Both markers and natural features are detected using computer vision algorithms ([70, 71]) which also have been demonstrated by Tinamu Labs ([72]). We require that enough previously mapped features are visible at any given point of the flight to assure that the state can always be determined, independently from other sensory information. Let $\overline{P_{im}} = \{P_{im\,k}\}_{k=1}^{N} \subset \mathbb{R}^2$ be the set of N features detected and identified in the image plane of the onboard camera, as exemplified in Figure 4.2. We can compute the coordinates in the image frame P_{im} from the global frame P_g as a function of the position and orientation of the drone $\chi = \{\chi_{\text{full}\,i}\}_{i=1}^{6}$

$$P_{im}(\chi) = C_{im}^{c} T_{c}^{g}(\chi) P_g, \qquad (4.4)$$

where $T_c^g(\chi)$ is the geometric transform from the global to the camera frame, and C_{im}^c the transform from camera frame to image plane given by the camera model. We can solve a nonlinear least squares problem to find an estimate of χ

$$\hat{\chi} = \arg\min_{\chi} \sum_{k=1}^{N} \|P_{im}(\chi) - \overline{P_{im}}\|_2^2. \tag{4.5}$$

Linearizing the nonlinear least squares problem around the solution, for normally distributed measurement errors, the estimate $\hat{\chi}$ is also normally distributed with variance $\mathbb{V}(\hat{\chi}) = \sigma^2(J^\intercal J)^{-1}$, where σ is the standard deviation of a measurement in the image plane, and $J = \nabla P_{im}(\chi)|_{\chi=\hat{\chi}}$ is the Jacobian. Assuming $\hat{\chi} \approx \chi$, we can compute the variance of the state estimate Σ^χ as a function of the drone coordinates

$$\Sigma^\chi(\chi) = \mathbb{V}(\hat{\chi})|_{\hat{\chi}=\chi}. \tag{4.6}$$

Let $q_{pos}(\Sigma^\chi(\chi)) = 1/\sqrt{Tr(\Sigma^\chi)}$ be a function mapping the covariance matrix to a scalar *quality of position fix* metric. This metric is used to define the feasible flying domain $\mathcal{C}_{\text{free}}$ as

$$\mathcal{C}_{\text{free}} = \{\chi \in \mathcal{C} : q_{\text{pos}}(\Sigma^\chi(\chi)) > \tau \mid \mathcal{W}\} \tag{4.7}$$

where χ is the drone state, $\mathcal{C}$ is the configuration space, τ is the minimum quality of fix threshold, and $\mathcal{W}$ the available information about the visual features positions.

The IMU and position measurements derived from the vision system are fused using an extended Kalman filter [73] to produce an estimate of the full state of the drone. The full estimate is used by the inner loop controller g while the planning algorithm only use the position and yaw localization.

4.4 Volume estimation

Now that we know the position and orientation of the drone, as well as their uncertainties we consider the LiDaR model. The sensor rotates in a plane with constant angular velocity, collecting approximately 10K distance samples per second, at discrete angles as illustrated in Figures 4.1 and 4.3. We can compute the hit point coordinates of the LiDaR measurements with the geometric transformation

$$z^m(\chi, \alpha) = T_g^l(\chi, \alpha) d_l, \tag{4.8}$$

where $T_g^l(\chi, \alpha)$ is the geometric transformation converting points in the LiDaR scan line frame to global coordinates, α is the angle of the sensor, d_l is the measured distance, and $z^m \in \mathbb{R}^3$ are the coordinates of the hit point. Propagating the

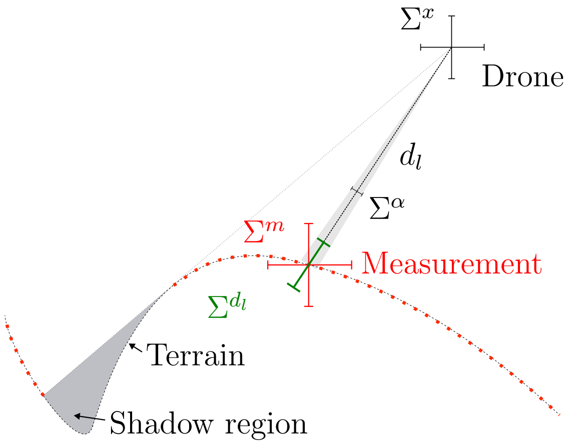


Figure 4.3: The 2D LiDaR measures the distance from the drone to the terrain d_l. The uncertainty in the drone position and orientation Σ^χ, LiDaR angle Σ^α and measurement distance Σ^{d_l} are propagated to obtain the covariance of the measurement Σ^m. Due to features of the terrain some areas might not be visible from current pose, marker as the shadow region in the bottom right.

uncertainty of each of the variables we obtain an estimate of the covariance of the measurement z^m, where again we use the Jacobian of the geometric transformation

$$\begin{aligned}\Sigma^m = {} & \nabla_\chi T_g^l(\chi,\alpha)d_l \; \Sigma^\chi \; (\nabla_\chi T_g^l(\chi,\alpha)d_l)^\intercal \\ & + \nabla_\alpha T_g^l(\chi,\alpha)d_l \; \Sigma^\alpha \; (\nabla_\alpha T_g^l(\chi,\alpha)d_l)^\intercal \\ & + T_g^l(\chi,\alpha) \; \Sigma^{d_l} \; T_g^l(\chi,\alpha)^\intercal,\end{aligned} \tag{4.9}$$

where Σ^m, Σ^χ, Σ^α and Σ^{d_l} are the covariance matrices of a measurement, the drone position and orientation, the LiDaR angle, and measured distance, respectively. Note that due the intrinsic properties of the sensor, only obstacles in a certain distance range can be detected $d_l \in [d_{min}, d_{max}]$. Figure 4.3 shows how the LiDaR scan obtains data about the surface. We condense the position uncertainty Σ^m, propagating the errors-in-variables to errors in height, given known statistical information about the slope of the terrain

$$\Sigma^m_{(z)} = \begin{bmatrix}\sigma_t & \sigma_t & 1\end{bmatrix} \Sigma^m \begin{bmatrix}\sigma_t & \sigma_t & 1\end{bmatrix}^\intercal, \tag{4.10}$$

where σ_t is the standard deviation of a Gaussian distribution fitted to an histogram of the slopes of this terrain, as shown in Figure 4.4.

Surface model

A common way to store the information about a 3D object is to use voxels, a generalization of the concept of pixel in 3D. Variations include sparse octree representations [74]. However, if we assume that the surface can be described with a

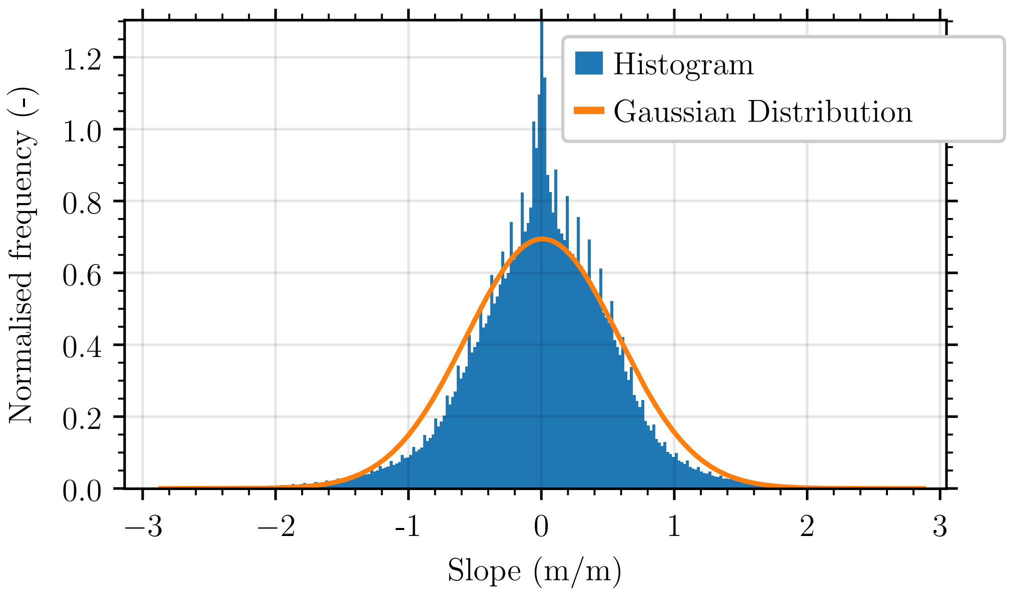


Figure 4.4: The volume slope distribution can be approximated by a normal distribution $\mathcal{N}(0, \sigma_s)$, and it is a property of the material.

function from x, y to h we can simplify the representation. One option is to use kriging-based methods [75] such as a GP [76]. This is the method used in [68] where an informative path planning framework is proposed.

We parameterize the surface using a grid of heights. Each point of the grid is described by a univariate normal distribution

$$h_{i,j} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2),\ i = \{1, \dots, N\},\ j = \{1, \dots, M\}, \tag{4.11}$$

where N and M are the dimensions of the grid. Between the grid points, we represent the surface distribution (the surface height is a random variable due to uncertainty) using a Gaussian Process model where the height grid provides inducing points.

We use a specific kernel tailored to the physics of our surface. Due to their physical properties, the materials piled up in the region of interest have a volumetric organization which can be described statistically. The Matérn Kernel is a good choice to estimate the correlation of the heights of two points separated by a distance s [76]

$$k(s) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l} s \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}}{l} s \right), \tag{4.12}$$

where l is the lengthscale, ν is a positive parameter controlling the smoothness of the function, and K_ν and Γ_ν are Bessel and Gamma functions, respectively.

We describe the surface as a sparse GP with fixed inducing points X which are obtained from the height grid, $f(x) \sim \mathcal{GP}(0, k(X, X'))$. At an arbitrary set of points $X_* \in \mathcal{D}$, we can predict the expected value and variance of the height of the surface using the equations

$$M_\Theta^{f_*} = \mathbb{E}[f_*] = K_{X_*X}[K_{XX} + \Sigma^2 I]^{-1} Z \tag{4.13a}$$

$$\Sigma_\Theta^{f_*} = \mathbb{V}[f_*] = K_{X_*X_*} - K_{X_*X}[K_{XX} + \Sigma^2 I]^{-1} K_{XX_*} \tag{4.13b}$$

where $\mathbb{E}$ is the expected value, $\mathbb{V}$ is the variance, K_{X_*X} is the covariance between the points X and X_* computed with the kernel (4.12), where s is the pairwise distance between the points ([76]), Z is the vector of the z coordinates of the inducing points, and Σ the vector of their uncertainties. For a certain kernel lengthscale, the vector $\Theta \doteq [Z, \Sigma]$ contains all the information needed to make predictions. For computational reasons, the kernel is made sparse by setting to zero the correlation between points outside of a ball with radius γl. With this model for the surface, we can readily compute the volume, its expected value and variance using the equations

$$\mathcal{V} = \iint_S f(x_*) dS \tag{4.14a}$$

$$\mu^{\mathcal{V}} = \mathbb{E}(\mathcal{V}) \approx A_\square \sum_{i=1}^{N} \sum_{j=1}^{M} M_{ij}^{f_*}, \tag{4.14b}$$

$$\left(\sigma^{\mathcal{V}}\right)^2 = \mathbb{V}(\mathcal{V}) \approx A_\square^2 \sum_{i=1}^{N} \sum_{j=1}^{M} \Sigma_{ij}^{f_*}, \tag{4.14c}$$

where $A_\square$ is the area of the base of each square cuboid.

Update

When a new measurement is obtained we first propagate it to the grid points. This is accomplished using the kernel $\Sigma_{(z)}^m + \sigma_t^2(e^{s/l} - 1)$, where σ_t is the standard deviation of the normal distribution that approximates the slope distribution, s is the distance between the measurement point and the grid coordinates, and l is the lenghtscale used in (4.12). We then use a Kalman filter to update the parameters of the height model (4.11)

$$\begin{aligned} \mu_k &= (I - K_k)\mu_{k-1} + K_k z_k \\ \sigma_k &= (I - K_k)\sigma_{k-1} \\ K_k &= \frac{\sigma_{k-1}^2}{\sigma_{k-1}^2 + ({\Sigma_{(z)}^m}_k + \sigma_t^2(e^{s/l} - 1))} \end{aligned} \tag{4.15}$$

where μ_{k-1}, μ_k, σ_{k-1}, and σ_k are the mean and standard deviation of each point of the grid, before and after the update step, z_k and $\Sigma^m_{(z)_k}$ are the measurement value and covariance from (4.9). K_k is the Kalman gain.

4.5 Planning

To estimate the volume of the surface, the robot needs to collect information of the whole surface, moving on a path suitable for this purpose. Two main approaches exist in path planning, that achieve this goal. Coverage path planning [77] is a method to design paths that visit all points of interest while avoiding obstacles. Typical cost functions minimize the length of the path, whereas with informative path planning [78] the objective is to maximize the amount of information in a feasible path. An overview of path planning algorithms can be found in [79].

For the purposes of planning we 1. assume a constant nominal surface height h_0, which allows us to compute d_l; 2. decouple the motion of the drone and the rotating LiDaR scan, since the latter is at least one order of magnitude faster; 3. neglect the possible effect of shadows, as defined in Figure 4.3; 4. we find a discrete set of waypoints instead of a continuous time trajectory. We use a greedy algorithm for planning, where the next point to be picked is the one with that minimizes the volume estimate uncertainty

$$\begin{aligned} r^*_{k+1} = \underset{r_{k+1}}{\arg\max} \quad & \|\sigma^{\mathcal{V}}_{k+1}(r_{k+1})\|_2 && (4.16a)\\ \text{s.t.} \quad & \|r_{k+1} - r_k\|_2 \leq R, && (4.16b)\\ & r \in \{\mathcal{C}_{\text{free}i}\}_{i=1}^4, && (4.16c) \end{aligned}$$

where $\sigma^{\mathcal{V}}$ is computed with (4.14c), r is the reference, R is the radius of a ball where the next step can lie. The simulations of the next section consider only the x and y components of r, and fix z and the yaw. The greedy algorithm is suboptimal but fast to evaluate, and allows us to test the surface reconstruction method. Future work will focus on implementing more advanced planning algorithms.

4.6 Simulation Results

We demonstrate our approach for volume estimation on a topographic map of the Alps mountain range, rich on features, which we scale by a factor of 10^{-3}.

We show simulation results for two trajectories, where the first one is a fixed, manually created square wave pattern, and the second is the trajectory resulting from applying the path planning algorithm from Section 4.5, in the variables r_x and r_y, while keeping r_z and r_{yaw} fixed. The disposition of visually identifiable

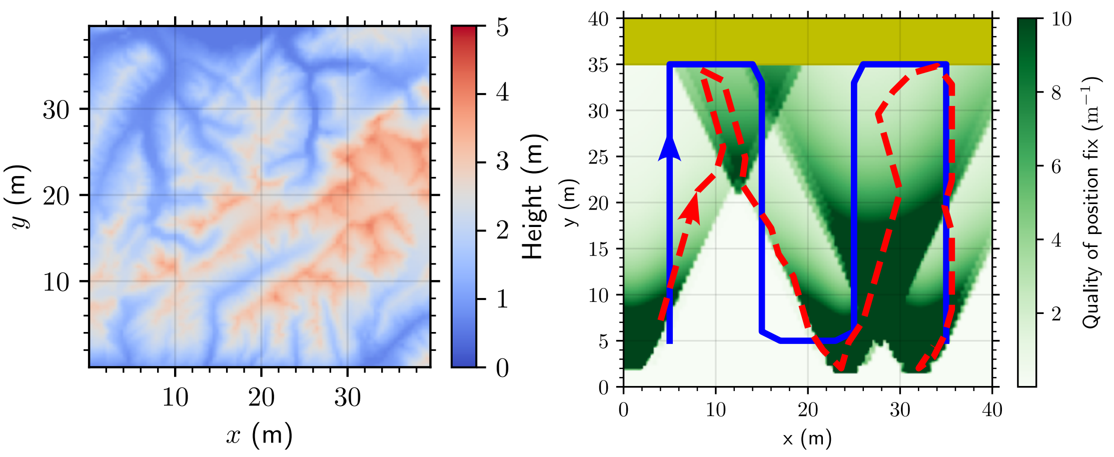
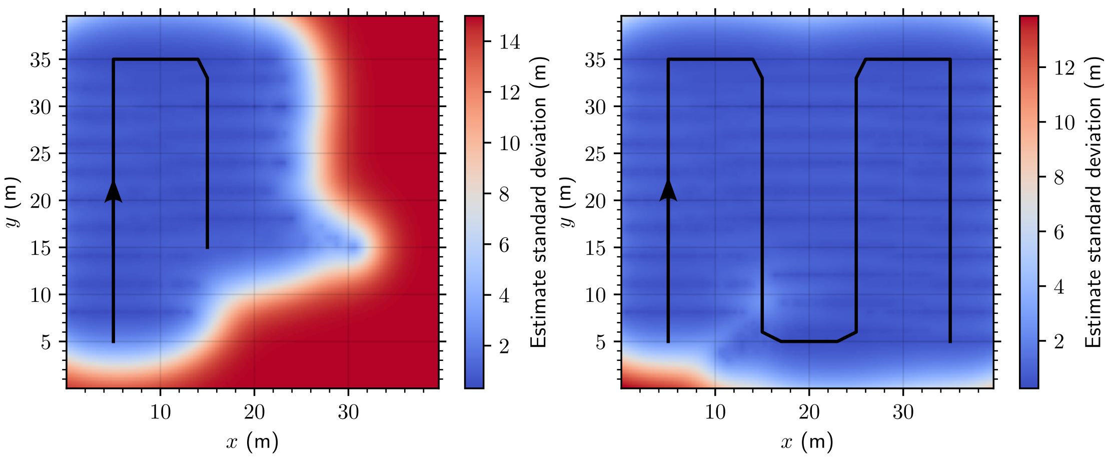


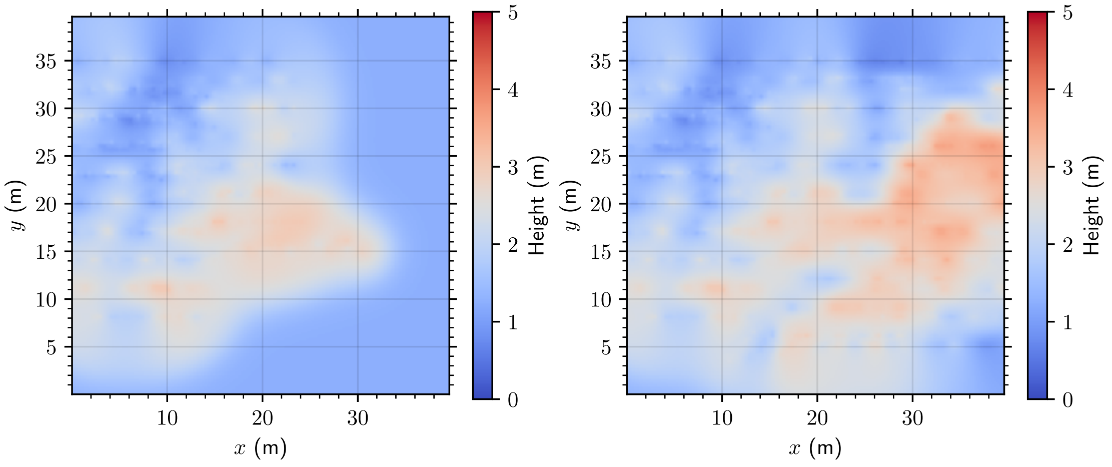
Figure 4.5: The left plot shows the ground truth surface height used in simulation. The right plot shows the quality of fix q_{pos} as a function of the xy coordinates, for $z = 7$ m and zero yaw. Overlayed are the trajectories of the square wave pattern in solid blue and the greedy algorithm in dashed red. The yellow region on the top of the plot indicates constraints in the position coordinates.

features in the example used in this simulations, shown in Figure 4.1, creates an uneven uncertainty of position map, as shown in the right plot of Figure 4.5, where darker colors represent lower uncertainty.

Figure 4.6a shows the path of the drone and the altitude uncertainty map, and Figure 4.6b the surface reconstruction map for the fixed square wave pattern. Traversing the whole region results in reduced uncertainty of surface reconstruction. It is in general not trivial to manually design paths that avoid constraints or regions with insufficient localization quality. As for the greedy algorithm and its resulting trajectory, we show in Figure 4.7a the path of the drone and the altitude uncertainty map, and in Figure 4.7b the surface reconstruction map. The simulation results show that a feasible reference trajectory is found that visits most of the region of interest through regions with a high quality of position fix and drives the uncertainty of the volume down to $\sigma^{\mathcal{V}}/\mu^{\mathcal{V}} = 2.26\%$ and a relative error of 2.53% for the greedy algorithm, comparable with $\sigma^{\mathcal{V}}/\mu^{\mathcal{V}} = 2.42\%$ with a relative error of 2.30% for the square wave pattern. We speculate that the two methods perform similarly in this example due to the greedy nature of the planning algorithm used, which optimizes only for the next step ahead, thus finding only an approximate solution for the optimization problem defined in (4.3). Also note that in this preliminary result the yaw and altitude are kept fixed, so the planner has less degrees of freedom to explore. These factors taken together make it difficult to improve on the benchmark square wave pattern. Figure 4.8 shows the evolution of the volume estimate as well as the uncertainty as a function of the number of samples



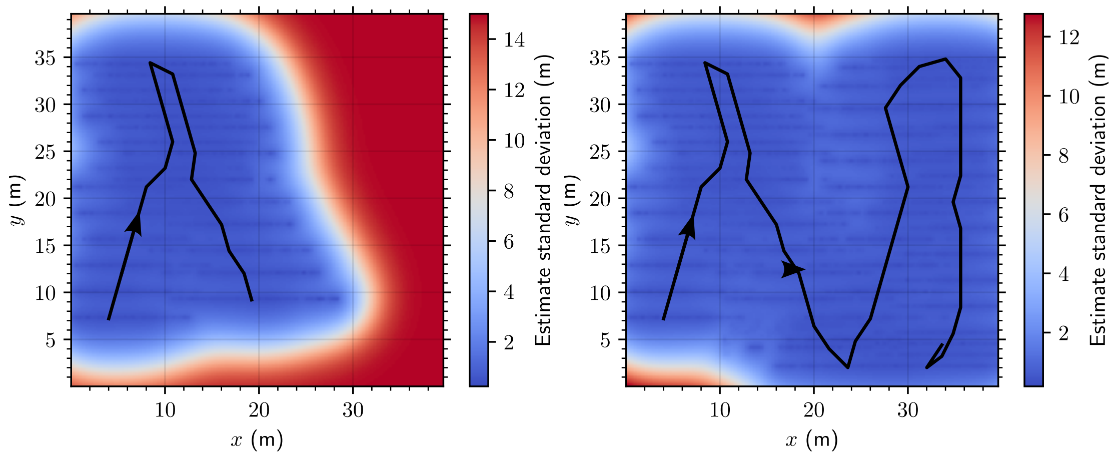
(a) Uncertainty map.



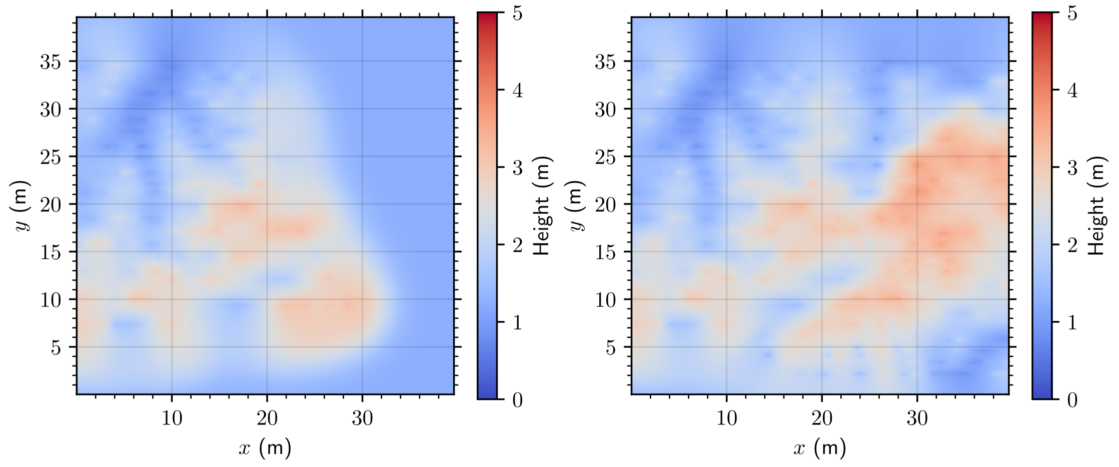
(b) Surface renconstruction.

Figure 4.6: Simulation results. Uncertainty map and Surface reconstruction with the square wave pattern after 20 steps (left plot) and 50 steps (right plot).

collected. The two paths have the same length and number of samples, and the evolution of the volume and its uncertainty is similar for both paths. Furthermore, the final reconstructions from Figures 4.6b and 4.7b approximate the ground truth shown in Figure 4.5 adequately. In summary the simple greedy planner finds a path that is by all metrics similar to the manually designed square wave pattern, allowing the automation of the task of designing paths for experimental volume estimation campaigns.



(a) Uncertainty map.



(b) Surface reconstruction.

Figure 4.7: Simulation results. Uncertainty map and Surface reconstruction with the greedy planner after 20 steps (left plot) and 50 steps (right plot).

4.7 Experimental Results

The surface reconstruction and associated uncertainty, generated from experimental data collected from a warehouse containing loose sand is shown in Figures 4.9 and 4.10 for the square wave pattern and Greedy planner respectively. The trajectory length, altitude, yaw, and number of points used for surface reconstruction is the same in both cases. The measured quadcopter trajectory is overlayed in black, with the arrow marking the direction of travel. Some artifacts from the operation of the drone are observable. In Figure 4.9, the scan pattern is preceded and followed by segments connecting it from and to the starting and finish points. In Figure 4.10 the starting point is considered by the Greedy Planner, and after

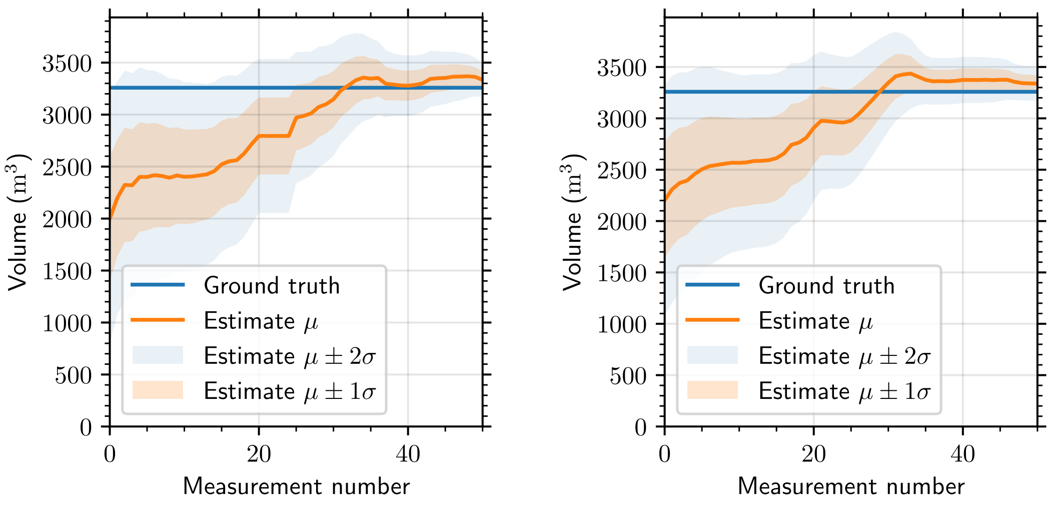


Figure 4.8: Simulation results. Evolution of the volume estimate and its uncertainty along the trajectory. The left plot shows the result for the square wave pattern and the right plot for the greedy algorithm.

the reference is completed, the quadcopter briefly hovers in place before initiating the landing procedure. An independent measurement using visual information and SfM of the same surface is shown in Figure 4.11. In Figure 4.12 we compare this independent measurement against the surface reconstruction of both trajectories. Due to the experimental nature of this data, we do not possess ground truth to compare against. A qualitative analysis of the results show that, similarly to the simulation case, the surface reconstruction using LiDaR data agrees between the two datasets. We verify that they are comparable as well with the data obtained with the SfM method, altought deviations up to $\pm 2\,\mathrm{m}$ are observed.

4.8 Discussion

We develop and implement a framework for volume and uncertainty estimation for an autonomous robotic platform collecting experimental data with a LiDaR scanner under variable position uncertainty. The surface reconstruction and volume estimation are validated in simulation with a feature rich surface, for two trajectories generated manually and with a simple greedy algorithm. The method is also tested experimentally, yielding consistent results, that are comparable to an independent method for surface reconstruction.

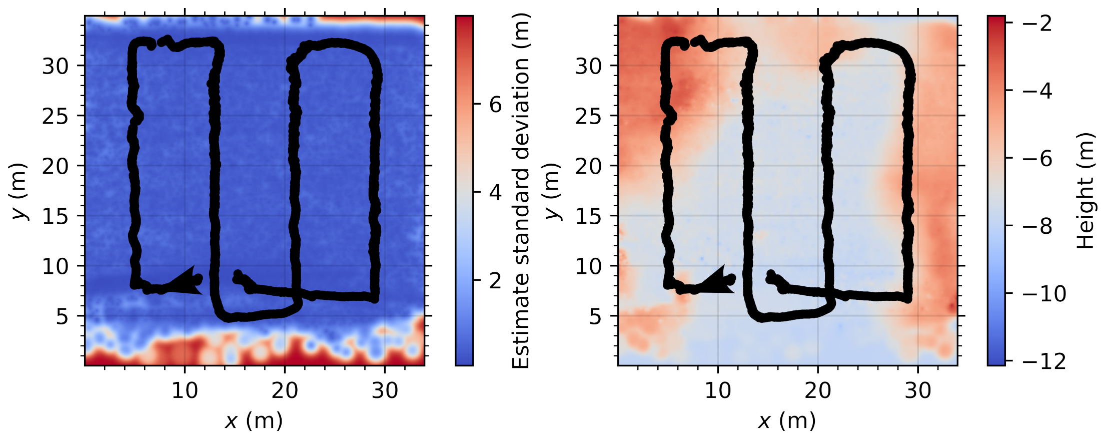


Figure 4.9: Experimental results. Uncertainty map and surface reconstruction with the square wave pattern. The black dots show the position of the drone during the scan.

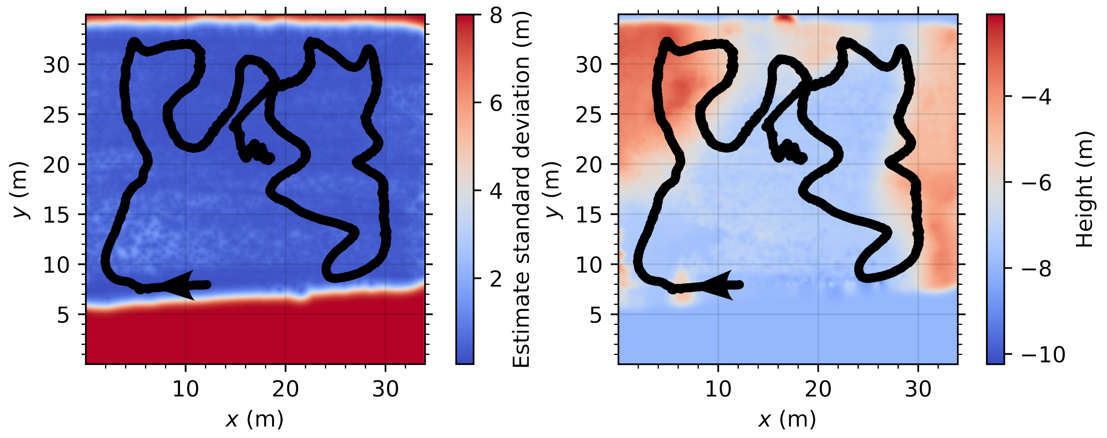


Figure 4.10: Experimental results. Uncertainty map and surface reconstruction with the greedy planner. The black dots show the position of the drone during the scan.

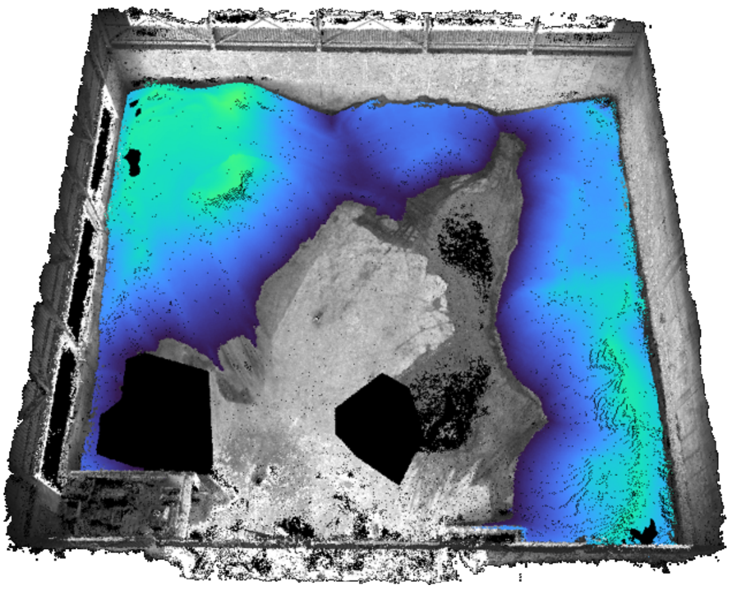


Figure 4.11: Experimental results. Independent surface reconstruction with a camera-based system using SfM provided by Tinamu Labs.

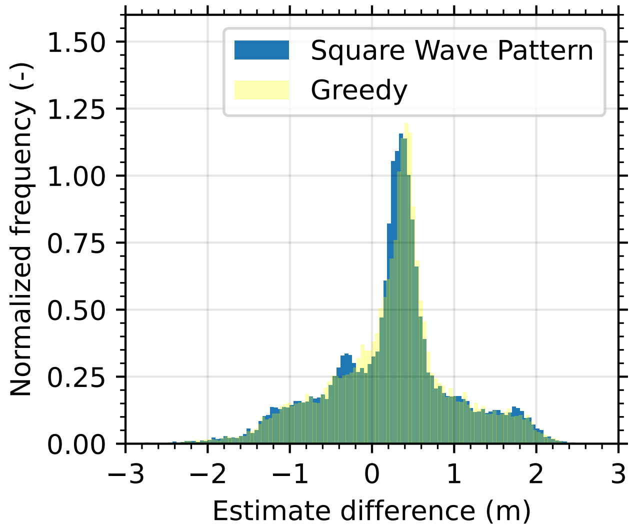


Figure 4.12: Experimental results. Histogram of the height difference between the surface reconstructions obtained with the square wave pattern and greedy trajectories, and the independent surface reconstruction with a camera-based system using SfM provided by Tinamu Labs. The two histograms overlap for the most part, with the overlap represented in green. Interpolation is used to obtain points from the grid-based surface model. This data was produce with data from the area where $10 \leq y \leq 30\,(\mathrm{m})$.

CHAPTER 5

Conclusion

In this thesis we propose a set of methods for trajectory planning, with applications in PMS control and robotic based inspection for volume estimation. The three methods proposed were demonstrated both in simulation and emprically.

5.1 Contributions

We developed a family of high fidelity, data driven, input-output models, that capture the dynamics of a specific PMS. Although specific, the model structure should generalize well for other systems with similar configurations. Due to its input-output structure the model prediction accuracy does not degrade over the prediction horizon. Using this model, we proposed and validate a method for offline optimization of input trajectories for PMS, that improves the accuracy vs. trajectory time trade-off curve, when compared to the default controller. Using a gradient model and experimental iterations, we propose and validate a OB-ILC method that reduces the tracking error.

The trajectory optimization methods of Chapters 2 and 3 are similar in their goals, but differ considerably on their approach. In Chapter 2 we rely heavily on a high-fidelity model. Although the computational burden is high, the method allows for offline optimization of input trajectories, without the need for further iterations. In contrast, in Chapter 3, a small number of iterations is needed to achieve an acceptable error, with a low computational burden. Given that the primary goal of designing fast trajectories is to improve productivity, a practitioner should consider this trade-off when deciding which approach to use. The iterative nature of the OB-ILC method, leads to experimental campaigns that take an order of magnitude longer to complete. This qualitative insight, obtained after having run a large amount of experiments, makes it clear that for one-off trajectories, or whenever the experimental apparatus is a production bottleneck, it is worth paying the additional computational cost of optimizing trajectories with the method proposed in Chapter 2. Another factor to consider is the need for an initial set of input-output responses that provides enough information to train the nonlinear model required for the method of Chapter 2. Although a database of previously run trajectories may already exist, there might be the need to run experiments for the purpose of identification. It is known that due to normal wear and tear, among other factors, the system response can change over time. We have experienced a change in dynamics first hand, when a structural part suffered damage, and after the system underwent repairs, the dynamics had changed considerably. However we were able to update the existing model, training it with a relatively small amount of experimental data from the "new" system. This indicates that one feasible option to keep up to date with the current true dynamics of the system is to continually record the experimental data and periodically update the nonlinear model. Regarding the OB-ILC method, it works best with the simpler linear model, for which much less data is needed for identification, needs less computational resources and can cope well with changes in dynamics.

We propose and validate a method for surface reconstruction of a surface by processing LiDaR data obtained from a quadcopter. The surface reconstruction method is incorporated in the planning of trajectories that minimize the volume estimate uncertainty, using a greedy planning algorithm.

Finally, with the method we propose in Section 4 we are able to efficiently reconstruct the surface from LiDaR data. The planned trajectory with a greedy algorithm offers similar levels of performance compared to a regular pattern, which is to be expected, given the uniform environment where the test was conducted.

5.2 Future Work

During the development of this work numerous divergent lines of research were identified, but not pursued due to time limitations. We present a non-exhaustive summary of possible work directions:

- As a next step we envision an experimental comparison of the controller developed in Chapter 2 against additional benchmarks. We advise to expand the search area by evaluating the adequacy of control methods initially devised for other applications for the specific context and challenges of PMS.

- A formal analysis to provide robust convergence guarantees on the proposed ILC method.

- Our findings indicate that the model demonstrating the highest accuracy in output prediction is not the one with the best derivative information for ILC. Notably, in this application, the simpler linear model outperforms the nonlinear ANN model. Consequently, it is relevant to investigate how to train machine learning models that better capture the sensitivity of the output in respect to input variations. Potential improvements could involve modifications to the loss function, alternative activation functions or network architectures, as well as smoothing of the derivative estimates.

- The trajectory planning strategy presented in Chapter 4, and employed for the validation of the surface reconstruction method, presents potential areas for improvement. One option is to convert the Greedy algorithm into a receding horizon planner. However, this approach poses challenges due to the additional computational load that it entails.

- The strategy introduced in Chapter 4 also requires further testing in more complex environments. This is necessary in order to fully evaluate the potential of the trajectory planning method.

Bibliography

[1] W. Kim and D. L. Trumper, "High-precision magnetic levitation stage for photolithography," *Precision engineering*, vol. 22, no. 2, pp. 66–77, 1998.

[2] J. J. Gorman and N. G. Dagalakis, "Force control of linear motor stages for microassembly," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 37130, pp. 615–623, 2003.

[3] T. Kim and J. J. Gorman, "A 3-dof MEMS motion stage for scanning tunneling microscopy," in *2022 IEEE 35th International Conference on Micro Electro Mechanical Systems Conference (MEMS)*, pp. 470–472, IEEE, 2022.

[4] M. Khosravi, C. Koenig, M. Maier, R. S. Smith, J. Lygeros, and A. Rupenyan, "Safety-aware cascade controller tuning using constrained Bayesian optimization," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2022.

[5] K. K. Tan, T. H. Lee, and S. Huang, *Precision Motion Control: Design and Implementation.* Springer Science & Business Media, 2007.

[6] P. Ouyang, R. Tjiptoprodjo, W. Zhang, and G. Yang, "Micro-motion devices technology: The state of arts review," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 5, pp. 463–478, 2008.

[7] S. Balula, A. Liniger, A. Rupenyan, and J. Lygeros, "Reference design for closed loop system optimization," in *2020 European Control Conference (ECC)*, pp. 650–655, IEEE, 2020.

[8] S. Balula, D. Liao-McPherson, A. Rupenyan, and J. Lygeros, "Data-driven reference trajectory optimization for precision motion systems," *Under submission*, 2023.

[9] S. Balula, E. Balta, D. Liao-McPherson, A. Rupenyan, and J. Lygeros, "Sequential quadratic programming-based iterative learning control for nonlinear systems," in *2023 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2023.

[10] S. Balula, D. Liao-McPherson, S. Stevšić, A. Rupenyan, and J. Lygeros, "Drone-based volume estimation in indoor environments," in *IFAC world congress*, 2023.

[11] A. Iannelli, M. S. Baumann, S. Balula, and R. S. Smith, "Experiments and identification of thermoacoustic instabilities with the rijke tube," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 757–763, IEEE, 2020.

[12] M. Ghobakhloo, "The future of manufacturing industry: a strategic roadmap toward industry 4.0," *Journal of Manufacturing Technology Management*, 2018.

[13] N. Kouraytem, X. Li, W. Tan, B. Kappes, and A. D. Spear, "Modeling process–structure–property relationships in metal additive manufacturing: a review on physics-driven versus data-driven approaches," *Journal of Physics: Materials*, vol. 4, no. 3, p. 032002, 2021.

[14] J. S. Srai, M. Kumar, G. Graham, W. Phillips, J. Tooze, S. Ford, P. Beecher, B. Raj, M. Gregory, M. K. Tiwari, *et al.*, "Distributed manufacturing: scope, challenges and opportunities," *International Journal of Production Research*, vol. 54, no. 23, pp. 6917–6935, 2016.

[15] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6137–6142, IEEE, 2010.

[16] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2397–2403, 2020.

[17] Y. Wang, F. Gao, and F. J. Doyle III, "Survey on Iterative Learning Control, Repetitive Control, and Run-to-Run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[18] E. C. Balta, K. Barton, D. M. Tilbury, A. Rupenyan, and J. Lygeros, "Learning-based repetitive precision motion control with mismatch compensation," *arXiv preprint arXiv:2111.10246*, 2021.

[19] K. L. Barton and A. G. Alleyne, "A norm optimal approach to time-varying ILC with application to a multi-axis robotic testbed," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 1, pp. 166–180, 2010.

[20] D. Liao-McPherson, E. C. Balta, A. Rupenyan, and J. Lygeros, "On robustness in optimization-based constrained Iterative Learning Control," *arXiv preprint arXiv:2203.05291*, vol. 6, pp. 2846–2851, 2022.

[21] K. Zhang, A. Yuen, and Y. Altintas, "Pre-compensation of contour errors in five-axis CNC machine tools," *International Journal of Machine Tools and Manufacture*, vol. 74, pp. 1–11, 2013.

[22] C.-C. Lo and C.-Y. Hsiao, "CNC machine tool interpolator with path compensation for repeated contour machining," *Computer-Aided Design*, vol. 30, no. 1, pp. 55–62, 1998.

[23] T. Haas, S. Weikert, and K. Wegener, "MPCC-based set point optimisation for machine tools," *International Journal of Automation Technology*, vol. 13, no. 3, pp. 407–418, 2019.

[24] X. Yang, R. Seethaler, C. Zhan, D. Lu, and W. Zhao, "A model predictive contouring error precompensation method," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 4036–4045, 2019.

[25] S. Di Cairano, A. Goldsmith, U. V. Kalabić, and S. A. Bortoff, "Cascaded reference governor–MPC for motion control of two-stage manufacturing machines," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 5, pp. 2030–2044, 2018.

[26] Z. Wang, C. Hu, Y. Zhu, and L. Zhu, "Prediction-model-based contouring error iterative precompensation scheme for precision multiaxis motion systems," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 5, pp. 2274–2284, 2021.

[27] Y. Jiang, J. Chen, H. Zhou, J. Yang, P. Hu, and J. Wang, "Contour error modeling and compensation of CNC machining based on deep learning and reinforcement learning," *The International Journal of Advanced Manufacturing Technology*, vol. 118, no. 1, pp. 551–570, 2022.

[28] H. Kim and C. E. Okwudire, "Simultaneous servo error pre-compensation and feedrate optimization with tolerance constraints using linear programming," *The International Journal of Advanced Manufacturing Technology*, vol. 109, no. 3, pp. 809–821, 2020.

[29] H. Kim and C. E. Okwudire, "Accurate and computationally efficient approach for simultaneous feedrate optimization and servo error pre-compensation of long toolpaths—with application to a 3d printer," *The International Journal of Advanced Manufacturing Technology*, vol. 115, no. 7, pp. 2069–2082, 2021.

[30] R. Eppler, "Airfoil data," in *Airfoil Design and Data*, pp. 163–512, Springer, 1990.

[31] G. Masetti and F. D. Giandomenico, "Analyzing forward robustness of feed-forward deep neural networks with LeakyReLU activation function through symbolic propagation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 460–474, Springer, 2020.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[34] A. Aboanber and Y. Hamada, "Generalized Runge–Kutta method for two-and three-dimensional space–time diffusion equations with a variable time step," *Annals of Nuclear Energy*, vol. 35, no. 6, pp. 1024–1040, 2008.

[35] G. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained Control and Estimation: An Optimisation Approach.* Springer Science & Business Media, 2006.

[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[37] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[38] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[39] T. D. Son, G. Pipeleers, and J. Swevers, "Robust monotonic convergent iterative learning control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1063–1068, 2015.

[40] A. Tayebi and C.-J. Chien, "A unified adaptive iterative learning control framework for uncertain nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 10, pp. 1907–1913, 2007.

[41] R. Adlakha and M. Zheng, "An optimization-based iterative learning control design method for UAV's trajectory tracking," in *2020 American Control Conference (ACC)*, pp. 1353–1359, IEEE, 2020.

[42] S. Mishra, U. Topcu, and M. Tomizuka, "Optimization-based constrained Iterative Learning Control," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 6, pp. 1613–1621, 2010.

[43] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," *IEE Proceedings-Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996.

[44] S. Gunnarsson and M. Norrlöf, "On the design of ILC algorithms using optimization," *Automatica*, vol. 37, no. 12, pp. 2011–2016, 2001.

[45] J.-X. Xu, "A survey on iterative learning control for nonlinear systems," *International Journal of Control*, vol. 84, no. 7, pp. 1275–1294, 2011.

[46] Y. Yu, C. Zhang, Y. Wang, and M. Zhou, "Neural-network-based iterative learning control for hysteresis in a magnetic shape memory alloy actuator," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 2, pp. 928–939, 2021.

[47] A. Schöllig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *2009 European Control Conference (ECC)*, pp. 1505–1510, IEEE, 2009.

[48] J. Lu, Z. Cao, R. Zhang, and F. Gao, "Nonlinear monotonically convergent Iterative Learning Control for batch processes," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5826–5836, 2017.

[49] K. E. Avrachenkov, "Iterative learning control based on quasi-Newton methods," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, vol. 1, pp. 170–174, IEEE, 1998.

[50] "Newton method based iterative learning control for discrete non-linear systems, author = Lin, T and Owens, DH and Hatonen, J, year = 2006, journal = International Journal of Control, publisher = Taylor & Francis, volume = 79, number = 10, pages = 1263–1276,"

[51] M. Volckaert, A. Van Mulders, J. Schoukens, M. Diehl, and J. Swevers, "Model based nonlinear iterative learning control: A constrained Gauss-Newton approach," in *2009 17th Mediterranean Conference on Control and Automation*, pp. 718–723, IEEE, 2009.

[52] K. Baumgärtner and M. Diehl, "Zero-order optimization-based iterative learning control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3751–3757, IEEE, 2020.

[53] Y. Chen, B. Chu, and C. T. Freeman, “Iterative learning control for path-following tasks with performance optimization,” *IEEE Transactions on Control Systems Technology*, vol. 30, no. 1, pp. 234–246, 2021.

[54] J. Bolder and T. Oomen, “Rational basis functions in iterative learning control—with experimental verification on a motion system,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 722–729, 2014.

[55] D. A. Bristow and A.-G. Alleyne, “A high precision motion control system with application to microscale robotic deposition,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 6, pp. 1008–1020, 2006.

[56] P. T. Boggs and J. W. Tolle, “Sequential quadratic programming,” *Acta numerica*, vol. 4, pp. 1–51, 1995.

[57] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: an operator splitting solver for quadratic programs,” *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[58] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, “A survey on inspecting structures using robotic systems,” *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, p. 1729881416663664, 2016.

[59] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE international symposium on mixed and augmented reality*, pp. 127–136, Ieee, 2011.

[60] E. van Rees, “Creating aerial drone maps fast,” *GeoInformatics*, vol. 18, no. 7, p. 24, 2015.

[61] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi, “An efficient method for fully automatic 3D digitization of unknown objects,” *Computers in Industry*, vol. 64, no. 9, pp. 1152–1160, 2013.

[62] B. Hepp, M. Nießner, and O. Hilliges, “Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 1, pp. 1–17, 2018.

[63] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, “Elastic LiDAR fusion: Dense map-centric continuous-time SLAM,” pp. 1206–1213, 05 2018.

[64] J. Zhang and S. Singh, “Loam: LiDaR odometry and mapping in real-time,” 07 2014.

[65] J. Zhang and S. Singh, “Low-drift and real-time LiDaR odometry and mapping,” *Autonomous Robots*, vol. 41, pp. 401–416, 02 2017.

[66] G. A. Hollinger, B. Englot, F. Hover, U. Mitra, and G. S. Sukhatme, “Uncertainty-driven view planning for underwater inspection,” in *2012 IEEE International Conference on Robotics and Automation*, pp. 4884–4891, IEEE, 2012.

[67] H. Zhu, J. J. Chung, N. R. Lawrance, R. Siegwart, and J. Alonso-Mora, “Online informative path planning for active information gathering of a 3D surface,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1488–1494, IEEE, 2021.

[68] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, “An informative path planning framework for UAV-based terrain monitoring,” *Autonomous Robots*, vol. 44, no. 6, pp. 889–911, 2020.

[69] S. Bouabdallah and R. Siegwart, “Backstepping and sliding-mode techniques applied to an indoor micro quadrotor,” in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 2247–2252, IEEE, 2005.

[70] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[71] K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani, “Monocular vision SLAM for indoor aerial vehicles,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1566–1573, 2009.

[72] Y. Jiang, “Online control of quadrotor and camera with MPC for robust vision-based flight,” Master’s thesis, ETH Zurich, 2022.

[73] G. Einicke and L. White, “Robust extended Kalman filtering,” *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2596–2599, 1999.

[74] S. Laine and T. Karras, “Efficient sparse voxel octrees,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 8, pp. 1048–1059, 2010.

[75] D. G. Krige, “A statistical approach to some basic mine valuation problems on the Witwatersrand,” *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[76] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[77] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[78] F. Stache, J. Westheider, F. Magistri, M. Popović, and C. Stachniss, “Adaptive path planning for UAV-based multi-resolution semantic segmentation,” in *2021 European Conference on Mobile Robots (ECMR)*, pp. 1–6, IEEE, 2021.

[79] S. M. LaValle, *Planning algorithms.* Cambridge University Press, 2006.

Samuel Balula

Curriculum Vitæ

Universitätstrasse 112,
8006 Zürich, Switzerland
+41 77 201 36 44
samuel.balula@gmail.com
linkedin.com/in/samuel-balula

Born 10.1993 (29)
Portuguese

A creative problem solver, with an engineering physics background, who loves to model and automate.

Education

06.2018–08.2023	**Doctoral student in Automatic Control**, *Automatic Control Lab/Inspire AG, Swiss Federal Institute of Technology Zurich (ETHZ)*, Zürich, Switzerland ○ Developed optimisation-based algorithms for trajectory planning, using high fidelity models built from experimental data. ○ Applications in Precision Motion System control and Autonomous Robotic Inspection.
09.2011–11.2016	**Master of Science in Engineering Physics**, *Instituto Superior Técnico – University of Lisbon*, Lisbon, Portugal ○ 5 years degree that combines physics and engineering. The broad curriculum includes mathematics, theoretical and experimental physics, analogue and digital electronics, robotics, machine learning, and management. ○ Used optimal control in the master thesis to swing-up and equilibrate an inverted pendulum. Awarded the 2017 Luís Vidigal Prize and the APCA 2018 M.Sc. Thesis Award.

Experience

06.2018–05.2023	**Scientific Assistant**, *Automatic Control Lab/Inspire AG, Swiss Federal Institute of Technology Zurich (ETHZ)*, Zürich, Switzerland ○ Teaching Assistant for "Nonlinear Systems and Control", "System Identification", and the "General Control Laboratory". ○ Supervised a total of 17 Semester and Master projects on a variety of control applications.
01.2017–04.2018	**Head of Research**, *Trigger.Systems*, Lisbon, Portugal ○ Trigger.Systems is a technological startup focused on closing the control loop with IoT solutions for agriculture, irrigation, and energy management. I was hired as the 5th employee. ○ Responsible for electronic design, mathematical modelling, optimisation, and product development. Participated in HR selection, contacted clients, suppliers, and, to a lesser extent, investors. As project manager, supervised the work of a small team of engineers.
09.2015–01.2017	**Teaching Assistant**, *Instituto Superior Técnico – University of Lisbon*, Lisbon, Portugal ○ Teaching Assistant for the "Microcontrollers" course (Engineering Physics program), where students learn basic concepts of electronics and embedded systems, by programming PIC microcontrollers in C and assembly. 2015/16 Excellence in Teaching Award.
09.2013–08.2015	**Researcher**, *Instituto de Plasmas e Fusão Nuclear (IPFN)*, Lisbon, Portugal ○ Collaborated in the development of the e-lab platform, where physics experimental apparatus are available online for remote control and data acquisition. ○ Developed and integrated the firmware, electronics, and science apparatus, taking projects from idea to deployment.

Other skills and interests

Software	Linux enthusiast, confident with Python, C, Julia, Mathematica, and Matlab. Experienced with C++, Bash, Assembly, HTML, javascript, CAD, Kicad, (ng)spice, Git, ROS, etc.
Hardware	Circuit and PCB design. I may tear things apart to understand how they work.
Languages	C2 English, Native Portuguese, B1 French, A1 German.