Diss. ETH No. 26951

Learning Control Objectives from Human Interactions: Methods and Applications

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

MARCEL MENNER
M.Sc., Technische Universität München

born on July 13, 1989
citizen of Germany

accepted on the recommendation of

Prof. Dr. Melanie N. Zeilinger, examiner
Prof. Dr. Robert Riener, co-examiner
Prof. Dr. Aude Billard, co-examiner

2020

Institute for Dynamic Systems and Control
ETH Zurich
Switzerland

Abstract

This thesis investigates the design of control objectives for the automatic feedback control of dynamical systems. In particular, it presents methodologies—in addition to their applications—that aim to improve the operation of dynamical systems by learning from human interactions. The methodologies rely on a melding of model-based design and data-based calibration, where model classes are defined that ensure the system's safe operation, while data are exploited to improve the performance of the automatic feedback controller. The thesis discusses three concepts for learning from human-generated data: optimality condition-based learning, supervised learning, and statistical estimation. The three concepts are discussed in three parts and are employed for learning optimal controllers for predicting human movements, for automating a rehabilitation robot, and for personalizing the driving style of autonomous vehicles.

The first part considers the learning of control objectives and constraints for predictive control formulations. It develops algorithms for learning from noisy data, where the model assumptions are expressed by optimality conditions in the form of the Karush-Kuhn-Tucker conditions. The methods can capture complex behaviors from data by means of control objectives and constraints, while generalizing well due to model assumptions on the controller structure. The methods are applied to training a predictive model of human movements in a manipulation task. Learning results indicate that individual human movements in a manipulation task can be predicted using an optimal control formulation.

The second part considers the design of control objectives using a supervised learning technique. The learning technique uses ratings as quantitative evaluations of the dynamical system's operation and introduces an application-motivated constraint to render the control objective suitable for a gradient-based feedback controller. The approach is applied to a gait rehabilitation robot with the goal of reducing the dependency on expert therapists during gait training with the robot. In this context, the control objective to be optimized by the automatic feedback controller is the physiology of the gait pattern. The supervised learning technique offers an efficient alternative to reinforcement learning without the need for a potentially lengthy trial and error search, which is vital for a gait rehabilitation robot in order to reduce the strain put on the patients. Experiments with able-bodied subjects suggest that the proposed technique permits the learning of a suitable control objective and facilitates automatically walking patients physiologically—without the intervention of therapists.

The third part considers the data-based calibration of optimal controllers using a statistical estimation technique, which is formulated by interpreting the control objective in terms of requirements for the system operation and their joint probability distribution. The key benefit of this formulation is that it systematically imposes a model for the probability distribution on the data through the control objective. The technique is applied to calibrate a motion planner for autonomous driving applications using data provided by human drivers to personalize the driving style of autonomous vehicles. Learning results using data from human drivers in a simulation environment suggest that the proposed control objective for human-conscious driving along with the statistical estimation technique enable a more natural and personalized driving style of autonomous vehicles for their human passengers.

Kurzfassung

Diese Arbeit untersucht den Entwurf von Kostenfunktionen für die Regelung von dynamischen Systemen. Die Arbeit präsentiert sowohl Methoden als auch deren Anwendungen, die darauf abzielen, den Betrieb dynamischer Systeme zu verbessern, indem sie von menschlichen Interaktionen lernen. Die Methoden stützen sich auf eine Mischung aus modellbasierter Formulierung und datenbasierter Kalibrierung, sodass Modellklassen den sicheren Betrieb des dynamischen Systems bewahren, während Daten genutzt werden, um die Regelgüte zu verbessern. Diese Arbeit diskutiert drei Konzepte für das maschinelle Lernen von menschlichen Interaktionen: Lernen basierend auf Optimalitätsbedingungen, überwachtes Lernen und statistisches Lernen. Die drei Konzepte werden in drei Teilen erörtert und werden eingesetzt für die Prädiktion menschlicher Bewegungen, das Automatisieren eines Rehabilitierungsroboters und das Personalisieren von autonomen Fahrzeugen.

Der erste Teil handelt vom Erlernen von Kostenfunktionen und operativen Beschränkungen für die modellprädikative Regelung. Es werden Algorithmen für das Lernen von verrauschten Daten entwickelt, indem Modellannahmen mittels Optimalitätsbedingungen in der Form der Karush-Kuhn-Tucker Bedingungen genutzt werden. Die entwickelten Methoden können komplexe Zusammenhänge aus Daten erfassen, während die Modellannahmen bezüglich der Reglerstruktur sicherstellen, dass die gelernten Resultate gut generalisieren. Die Methoden werden angewendet, um ein Prädikationsmodell für menschliche Bewegungsabläufe in einer haptischen Manipulierungsaufgabe zu erlernen. Die gelernten Resultate deuten darauf hin, dass individuelle menschliche Bewegungen in solch einer haptischen Aufgabe mit einem prädikativen Regler prognostiziert werden können.

Der zweite Teil handelt vom Erlernen von Kostenfunktonen mittels einer Technologie für überwachtes Lernen. Die Lernmethode nutzt Bewertungen als quantitative Evaluierung vom Betrieb dynamischer Systeme und führt eine Beschränkung für den Lernalgorithmus ein, womit die Kostenfunktion für die optimale Regelung nutzbar wird. Die Methode wird angewendet für die Automatisierung eines Rehabilitierungsroboters für Gangtherapie mit dem Ziel, die Therapie unabhängiger von den Therapeuten zu ma-chen beziehungsweise die Therapeuten zu unterstützen. Die Kostenfunktion zielt darauf ab, ein möglichst physiologisches Gangbild zu erreichen. Das entwickelte Verfahren bietet eine effiziente Alternative zu Methoden, die verstärkendes Lernen nutzen, da es kein Ausprobieren bestimmter Operationszustände bedarf, was für eine gesunde Gangtherapie essentiell ist. Experimente mit unbeeinträchtigten Freiwilligen indizieren, dass die entwickelte Methode eine geeignete Kostenfunktion für die Regelung des Roboters lernen kann und damit ein physiologisches Gangbild erzielt.

Der dritte Teil handelt vom datenbasierten Kalibrieren eines optimalen Reglers mittels einer Methode für statistisches Lernen, welche darauf basiert, dass die Kostenfunktion als stochastisch interpretiert wird. Der Hauptvorteil dieser Formulierung ist, dass den Daten eine Wahrscheinlichkeitsverteilung systematisch auferlegt wird. Die Methode wird angewendet für das Kalibrieren eines Trajektorienplaners für autonomes Fahren mit dem Ziel, autonome Fahrzeuge auf den individuellen Fahrstil der Insassen anzupassen. Die Resultate mit Daten aus einem Fahrsimulator zeigen, dass die vorgeschlagene Kostenfunktion für menschennahes Fahren, zusammen mit der Methode für statistisches Lernen, einen natürlicheren und personalisierten Fahrstil für autonome Fahrzeuge ermöglichen.

Acknowledgments

Many people have supported me on my path to graduating from ETH Zurich.

First and foremost, I owe my deepest gratitude to my adviser Melanie N. Zeilinger for her unfaltering support during the past four years, in both my research agenda and my career. I am very thankful to her for the freedom that she has given me to pursue various projects, the trust she has put into me to deliver results, and the group culture that she has created. I am also very grateful to Robert Riener and Aude Billard for co-examining my doctoral thesis and to Raffaello D'Andrea for his support. A special thanks to Florian Holzapfel for paving the way for me from Munich, through Massachusetts, to Zurich.

I want to thank my collaborators, Lars Lünenberger and the team from Hocoma AG, Stefano Di Cairano and Karl Berntorp from Mitsubishi Electric Research Labs, Alex Domahidi from embotech AG, Stefan Schrade from the Rehabilitation Engineering Laboratory at ETH Zurich, as well as the entire *breathe* team. They all have helped to shape my perspective on control theory, my approach to research, and my scientific thinking. I am very thankful to my fellow team members and colleagues at ETH Zurich, especially Lukas, Andrea, Kim, Elena, Simon, Jérôme, Michael, and Thiva, for engaging discussions, for unforgettable trips, and for their friendships. Thanks to Helen for her great administrative work and to all the students, who I had the pleasure of supervising and who have supported me with their ideas and projects. I want to thank my colleagues in the research communities, who I interacted with at conferences, during my internship, and at summer schools. They all were important in making my Ph.D. experience so memorable.

I want to thank my friends and my family for the support system that allowed me to further my education. I would like to specifically acknowledge my parents, Jutta and Reimund, for their guidance and love over the years, and my brother, Marco, for the levity in life that only a sibling can provide. Finally, I would like to thank Courtney for being my confidant and for her loving encouragement over the past four years.

Financial Support:
This thesis has been supported by the Swiss National Science Foundation under grant number PP00P2_157601/1.

Contents

Preface

This thesis documents the research carried out by the author during his doctoral studies under the supervision of Professor Melanie N. Zeilinger at the Institute for Dynamic Systems and Control (IDSC) at ETH Zurich between May 2016 and May 2020. It is presented in the form of a cumulative thesis: its main content is four self-contained research articles that have been published or are accepted for publication.

The research articles are divided into three parts and put into context by four introductory chapters. Part A describes methodologies for learning control objectives using optimality conditions. Part B introduces a method for learning control objectives using supervised learning, and presents its application to automating a rehabilitation robot. Part C presents an approach for learning control objectives for personalizing self-driving cars. Chapter I provides the motivation for the research and introduces the problems addressed in the thesis. Chapter II provides a summary of the scientific contributions in this thesis, as well as related publications. An outlook of possible future work is given in Chapter III. Chapter IV provides an extended overview of the related work in the literature.

Chapter I

Introduction

The study of feedback mechanisms and dynamical systems has created a far-reaching understanding of many aspects of society and technology. In a feedback mechanism, an observation or output of a system impacts the behavior of the same system, thereby forming a closed loop. If a feedback mechanism is well-understood, then the behavior of the system may be manipulated in order to obtain a desirable outcome. Control theory studies the manipulation of a dynamical system using feedback mechanisms by exploiting a mathematical representation of the system [1]. This mathematical representation is referred to as a model and facilitates making predictions about the evolution of the system being considered. However, as dynamical systems and robots become more and more complex and operate in increasingly unstructured environments, engineering suitable feedback mechanisms has become more difficult. On the other hand, the surge in sensing capabilities and computational resources provides a new potential for automating the design of feedback mechanisms for control using data and algorithms.

In the context of systems engineering or process engineering, utilizing feedback mechanisms is a powerful tool to automatically steer the system to its intended purpose or operating point [2]. In a very general and abstract form of automatic feedback control, a dynamical system aims to optimize a certain goal (or reach a target), while respecting the laws of nature, as well as safety measures and potential regulations:

$$\begin{aligned} \text{optimize} \quad & \text{goal} && \text{(1a)} \\ \text{respecting} \quad & \text{laws of nature} && \text{(1b)} \\ & \text{safety and regulations.} && \text{(1c)} \end{aligned}$$

The goal in (1a) that the dynamical system aims to optimize is often apparent to the system designer, but not easily stated as a precise mathematical expression, e.g., it could be given in semantic language. The laws of nature can be viewed as deterministic, such that knowing the state of the universe, along with all forces acting in nature at a given instant, an oracle could predict the state of the world at any subsequent time, cf. Laplace [3, page vi]. In principle, an oracle controller would be capable of manipulating the dynamical system using a pre-determined set of control actions, referred to as feedforward control. However, even if the state of the universe and all forces in nature were known, a model with all necessary intricacies to accurately reflect nature is infeasible to build and to simulate. In contrast to the laws of nature, the safety conditions and regulations in (1c) are man-made laws that a system ought to obey.

The task of the control system designer is to formalize (1) by finding a sufficient level of mathematical abstraction to facilitate its usability in a real-world system. Predictive control frameworks are suitable methodologies for this task and rely on repeatedly solving a mathematical formulation that approximates (1), while using sensor measurements as feedback signals to mitigate the uncertainties that result from the approximation:

$$\begin{aligned} \text{optimize} \quad & \text{control objective } \phi(x,u) && \text{(2a)}\\ \text{subject to} \quad & \text{system dynamics model } 0 = f(\dot{x},x,u) && \text{(2b)}\\ & \text{constraints and bounds on } x \text{ and } u, && \text{(2c)} \end{aligned}$$

where x refers to the state of the system and u denotes the control action. In the context of mechanical systems, the dynamics in (2b) is represented by a set of differential equations that describe the approximated motion of the system and is often derived from first principles, e.g., Newton's laws. The constraints in (2c) reflect both safety and regulation requirements, as well as bounds for the validity of the model for the system dynamics in (2b).

This thesis focuses on the design of control objectives and constraints for optimal controllers. As such controllers are executed in real-time, there is limited time available to compute the control actions [4]. Consequently, a control objective suitable for the real-time implementation of an automatic feedback controller requires the following:

Req. 1. The control objective in (2a) must be a surrogate for the actual goal in (1) that the dynamical system is meant to achieve or optimize.

Req. 2. It must be modeled in terms of the decision variables, i.e., the state, x, and the control action, u.

Req. 3. It must facilitate numerical optimization, e.g., gradient-based optimization or particle filtering.

Often these three requirements make the controller design not very intuitive resulting in a tedious manual engineering effort to meet all three requirements. In this thesis, we take a more systematic approach and use data for the automated design of control objectives.

Data-based Calibration of Optimal Controllers

Increased sensing and computing capabilities offer a novel potential in not only using data in the form of sensor measurements for the feedback control of the dynamical system, but also to alleviate the discrepancy between (1) and (2) to improve controller performance. This string of research is commonly referred to as learning-based control, see e.g. the review in [5] in the context of predictive control. Mathematically, the problem of learning a model, e.g., of the objective or the system dynamics, for the control of a dynamical system characterizes an inverse problem, in which the causal factors that produce a set of observations are inferred [6]. We define learning-based optimal control as methodologies that improve (in some suitable sense) the formulation in (2), measured with respect to (1). The main focus of research has been on "improving" the system dynamics used for the controller design as data are accumulated, either online, e.g., in [7], [8], or episodically, e.g, in [9]–[11].

This thesis largely focuses on the learning of control objectives for optimal control formulations from data, but also touches upon the principles of learning constraints. The three requirements for the design of a control objective promote the idea of using a gray-box learning approach, in which model classes with a fixed structure and with free parameters are defined that satisfy Requirements 2 and 3, while data are used to calibrate the parameters in order to meet Requirement 1. In other words, we are learning under model constraints that render the control objective applicable for feedback control. While a more flexible model class would, in principle, be possible, it is more difficult to maintain the three requirements. This philosophy stands in contrast to end-to-end learning, e.g., [12], that results in a black-box learning approach, which does not necessarily guarantee the satisfaction of the three requirements and consequently, mathematical guarantees for the system's safe operation can often not be established.

The research in this thesis is centered on the learning of control objectives that are tailored to accommodate the numerical procedure involved in solving the corresponding control problem. For the considered learning tasks, the data are provided by humans, who either demonstrate how the dynamical system should be operated or provide a quantitative evaluation of the system's behavior under different operating conditions. Part A presents methods for learning both control objectives and operational constraints for optimal control formulations that use gradient-based numerical optimization. In particular, the methods learn from demonstrations and focus on accommodating noisy data, which is particularly challenging in the presence of constraints. Part B presents a method for leveraging labeled data in the form of ratings for learning a control objective to be used for a gradient-based feedback controller. Part C presents methodologies for learning control objectives from demonstrations, which are particularly well suited for particle filter-based numerical optimization techniques, but can also be used for learning control objectives for gradient-based optimization.

Learning from Human Interactions

Learning from human interactions has gained momentum in the last decades as robotic systems are becoming more and more potent in their actuation, mechanical design, software, as well as computational resources. Often robots are intended to take over tasks from humans, e.g., opening doors, doing chores, assembling a product, or driving a car [13], [14]. This suggests that a robot may be trained by imitating an expert executing the task, which led to the coining of the terms of apprenticeship, imitation, or expert learning [15]. Learning from human interactions is a paradigm that exploits a human's experience by utilizing either demonstrations or ratings for the systematic design of autonomous systems, cf. [16]. Thus, methodologies for learning from human interactions are grounded in the fundamental hypothesis that the data represent the desired behavior or, at least, that the desired behavior can be inferred from humans.

This thesis focuses on the data-based design of automatic feedback controllers for robots and autonomous systems, where the data are provided by humans in the form of demonstrations or ratings. For learning from demonstrations, the data are not labeled, i.e., we make an implicit assumption or hypothesis that the data represent intended behavior. For learning from ratings, the data are labeled, i.e., the human provides an explicit evaluation of the behavior. In the context of learning from human interactions, one major challenge—motivating the development of the methodologies in this thesis—is the presence of noise and the suboptimal nature of the human-generated data.

The research in this thesis is centered on learning optimal control formulations, either for the purpose of predicting systems with a human in the loop or controlling autonomous systems to achieve human-like performance. Part A models human locomotion in a manipulation task with an optimal and predictive control formulation. This control model is trained using data from a human's manipulation movements and utilized for making motion predictions of individual human subjects. Part B builds a control objective required for the automation of a gait rehabilitation robot.

In this research project, the goal, as in (1a), is to "walk patients with a healthy/natural gait," which is formalized into a usable control objective satisfying the three requirements using ratings provided by experienced therapists. Part C presents methods to calibrate a motion planner for autonomous driving applications using data provided by human drivers for a personalized autonomous driving experience.

Chapter II

Contributions

This chapter briefly summarizes the key contributions of each paper and draws connections between the individual publications. Many of the results were obtained in close collaboration with industry partners and through supervised student projects, as indicated below. Additional contributions were made in several co-authored publications and participation in interdisciplinary projects, as well as through teaching assistance.

Part A. Learning Optimal Controllers using Optimality Conditions

[P1] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained Inverse Optimal Control With Application to a Human Manipulation Task," *IEEE Transactions on Control Systems Technology (early access)*, 2019.

This journal contribution considers the design of optimal controllers, where model assumptions, e.g., a safe operating region or the system dynamics, are explicitly formulated and enforced as constraints. We developed algorithms for learning the control objective of optimal control formulations from potentially noisy and corrupted data, where the model assumptions are expressed by means of optimality conditions, i.e., first-order derivative tests in the form of the Karush-Kuhn-Tucker (KKT) conditions [17], [18]. This paper demonstrates how to formulate a finite-dimensional inverse problem of an infinite-horizon, constrained optimal control problem. Furthermore, it shows how to leverage the KKT conditions for learning the constraints of an optimal controller and shows how to cope with noise corrupting parts of the data.

The motivation for this work is to predict the movements of humans by means of an optimal control formulation. The methods developed for learning from noisy and corrupted data are particularly useful in this context as human movements are not purely predictive, but often corrupted by noise and reactive movements. The method was verified in experiments, which were conducted in collaboration with Peter Worsnop, former Master's student, at UC Berkeley. In the experiments, human subjects manipulated a passive kinematic object to achieve a certain configuration. The results indicate that individual human movements in a manipulation task can be predicted using an optimal control formulation.

[P2] M. Menner and M. N. Zeilinger, "Maximum Likelihood Methods for Inverse Learning of Optimal Controllers," in *21st IFAC World Congress (forthcoming)*, 2020.

Similarly to [P1], this conference contribution uses optimality conditions for the learning of optimal controllers. This contribution delineates three different KKT-based learning methodologies that vary in their model assumptions and their computational complexities. It proposes an efficient algorithm for learning control objectives in the presence of constraints, provides an interpretation of inverse optimal control methods as a bilevel optimization problem, and examines their theoretical properties.

Part B. Automating a Rehabilitation Robot using Supervised Learning

[P3] M. Menner, L. Neuner, L. Lünenburger, and M. N. Zeilinger, "Using Human Ratings for Feedback Control: A Supervised Learning Approach With Application to Rehabilitation Robotics," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 789–801, June 2020.

This journal contribution considers control tasks, where the objective is very abstract and not trivial to define mathematically but is more intuitive for humans. We developed a supervised learning method, where the key idea is to treat ratings provided by humans as data samples from an unknown control objective. The supervised learning method uses the system's state to calculate a feature vector and defines the human ratings as labels. The control objective, i.e. the map from the features to the ratings, is learned using classification-type ideas under model constraints that render the objective applicable for feedback control.

This method was developed in the context of a research project in collaboration with Dr. Lars Lünenburger from Hocoma AG, with the goal of automating a gait rehabilitation robot. One bottleneck restricting the automation of the robot is that therapists are needed to "set up the robot to walk patients with a natural/healthy gait" (the abstract objective). The method overcomes this bottleneck and enables the automation of the gait rehabilitation robot, which can improve patient training and can be more cost-effective. The method was successfully verified in closed-loop experiments (with able-bodied volunteers). The experiments along with the implementation of the algorithm on the robot were conducted in collaboration with Lukas Neuner, former Master's student.

Part C. Personalizing Self-Driving Cars using Statistical Estimation

[P4] M. Menner, K. Berntorp, M. N. Zeilinger, and S. Di Cairano, "Inverse Learning for Data-driven Calibration of Model-based Statistical Path Planning," *IEEE Transactions on Intelligent Vehicles (early access)*, 2020.

This journal contribution considers the design of optimization-based controllers that are expressed by means of a probability distribution, which allows for formulating complex requirements/targets while achieving statistical guarantees. We developed a method for learning such a probabilistic control objective from data using statistical estimation. Similarly to [P1], the data originate from a human who demonstrates how the dynamical system should be operated. The key benefit of this control objective is that it systematically adds stochasticity to the data, which are naturally noisy and nondeterministic as they are generated by humans.

This research project was conducted in collaboration with Dr. Stefano Di Cairano and Dr. Karl Berntorp from the Mitsubishi Electric Research Laboratories and the motivation is to adapt the driving style of autonomous vehicles to the individual human passenger. The motive for the individualization stems from the fact that the feeling of comfort and cautiousness in traffic can vary between human passengers. The method was verified in a simulation environment in which human subjects demonstrated their individual driving style.

Related Publications

Further contributions were made in several co-authored publications that are not included in this cumulative doctoral thesis.

[P5] M. Menner and M. N. Zeilinger, "Convex Formulations and Algebraic Solutions for Linear Quadratic Inverse Optimal Control Problems," in *European Control Conference*, 2018, pp. 2107–2112.

This conference contribution proposes inverse optimal control methodologies for linear quadratic regulator problems, which rely on the algebraic Riccati equation [19]. The contribution presents convex formulations—i.e., algebraic formulations, linear programs, as well as semidefinite programs—for learning control objectives, and highlights their computational efficiency.

[P6] M. Menner and M. N. Zeilinger, "A User Comfort Model and Index Policy for Personalizing Discrete Controller Decisions," in *European Control Conference*, 2018, pp. 1759–1765.

Similarly to [P3], this conference contribution proposes a supervised learning technique, however, differently to [P3], it shows how to learn a control objective in closed-loop using trial and error search. This contribution presents both an index-based control policy to smartly collect and process user feedback and a user comfort model in the form of a Markov decision process with a priori unknown user-specific state transition probabilities. The control policy utilizes explicit user feedback in the form of ratings to optimize a reward measure and addresses the exploration-exploitation trade-off in a multi-armed bandit framework.

[P7] M. Menner, K. Berntorp, M. N. Zeilinger, and S. Di Cairano, "Inverse Learning for Human-Adaptive Motion Planning," in *58th IEEE Conference on Decision and Control*, 2019, pp. 809–815.

This conference contribution is the initial investigation of the journal publication [P4]. It focuses on the principles and ideas of personalizing autonomous vehicles, without elaborating on computational and implementation aspects.

[P8] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based Model Predictive Control: Towards Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

This journal contribution provides an overview of research efforts in learning-based model predictive control. It categorizes three main research branches: In the first branch, data are used to form or enhance the model of the system dynamics. In the second branch, data are used to directly adjust the controller formulation with the goal of improving the closed-loop performance. In the third branch, predictive control methodologies are used to augment learning-based control systems with inherent safety properties, i.e., the predictive controller acts as a safety filter.

[P9] S. O. Schrade, M. Menner, C. Shirota, P. Winiger, A. Stutz, M. N. Zeilinger, O. Lambercy, and R. Gassert, "Knee Compliance Reduces Peak Swing Phase Collision Forces in a Lower-Limb Exoskeleton Leg: A Test Bench Evaluation," *IEEE Transactions on Biomedical Engineering (early access)*, 2020.

This journal contribution considers powered lower limb exoskeletons, which are a viable solution for people with a spinal cord injury to regain mobility for their daily activities. In particular, it examines compliant actuation, which may reduce forces during a potential collision impacting both the hardware and the user. We investigated experimentally how a variable stiffness actuator at the knee joint influences collision forces transmitted to the user via the exoskeleton. The results indicate that compliance in the knee joint of an exoskeleton can be favorable in case of a collision and should be considered when designing powered lower limb exoskeletons.

[P10] V. Lefkololous, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-Aware Motion Prediction for Autonomous Driving: A Multiple Model Kalman Filtering Scheme," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 80–87, Jan. 2021.

This journal publication considers the problem of predicting the motion of vehicles in the surroundings of an autonomous car, for improved motion planning in lane-based driving scenarios without inter-vehicle communication. The paper proposes an algorithm for generating intention-based motion predictions with a multiple model Kalman filter, combining ideas from physics-based, maneuver-based, and interaction-aware prediction approaches, cf. [20]. The motion primitives produce non-colliding predictions and are calibrated using the inverse optimal control techniques in [P5] in order to obtain human-like motions.

This research project was conducted in collaboration with Dr. Alexander Domahidi from embotech AG and the motivation is to achieve real-time capable and accurate motion predictions needed for autonomous driving algorithms that employ predictive planning techniques. The algorithm was verified with human-driven vehicle data from the Next Generation Simulation (NGSIM) dataset.

Contributions to Interdisciplinary Projects

Further contributions were made in two interdisciplinary projects.

Low-Cost Ventilator for COVID-19 Patients

This interdisciplinary research project was started to meet the peak ventilator demands for patients suffering from COVID-19. It is a collaboration between groups headed by Kristina Shea (Engineering Design and Computing Laboratory, ETH Zurich), Marianne Schmid Daners (Product Development Group Zurich, ETH Zurich), Martin Meier (Product Design, Zurich University of the Arts), and Melanie N. Zeilinger (Intelligent Control Systems Group, ETH Zurich). The goal of the project was the development of a low-cost and modular ventilation system using two paddles that squeeze an AmbuBag, thereby pushing air into the lungs of a patient.

[P11] L. Hewing, M. Menner, N. Tachatos, M. Schmid Daners, C. du Pasquier, T. S. Lumpe, K. Shea, A. Carron, and M. N. Zeilinger, "Volume Control of Low-Cost Ventilator with Automatic Set-Point Adaptation," in *European Control Conference (submitted)*.

This contribution i) experimentally shows that for satisfying the medical requirements on the ventilator, in particular for accurately tracking tidal volumes, the controller needs to be adapted to the individual patient and the different configurations, e.g., hardware or operation modes; ii) proposes a set-point adaptation algorithm that uses sensor measurements of a flow meter to automatically adapt the controller to the setup at hand; and iii) experimentally shows that such an adaptive solution improves the performance of the ventilator for various setups.

Furthermore, the ventilator solution "breathe, from ETH with ♡" attracted public attention and press coverage:

https://ethz.ch/en/news-and-events/eth-news/news/2020/05/creating-a-low-cost-ventilator.html

Data-based Diagnosis of Normal Pressure Hydrocephalus

Machine learning techniques are often used to identify similarities and differences in data sets that are too large or too complex for humans to assess. The goal of this project is the assistive diagnosis of normal pressure hydrocephalus leveraging machine learning techniques. This project is a collaboration with Kiran Kuruvithadam and Marianne Schmid Daners

from the Product Development Group, ETH Zurich, and the Neurosurgery Department of the University Hospital Zurich.
The main symptoms used for the diagnosis of normal pressure hydrocephalus are characteristics in the gait cycle, which are difficult to identify in the doctor's office. The conceptual idea is that wearable IMU sensors can be used to capture gait characteristics over days in the patient's home environment, where the data could be used to improve the diagnosis.

Teaching Activities

Teaching Assistance

Model Predictive Control (151-0660-00L)	Spring 2017, 2018
Signals and Systems (151-0575-01L)	Fall 2016, 2017

Student Supervision

Master's Theses

- Vasileios Lefkopoulos, 2019,
  Collaboration with embotech AG, Zurich, Switzerland.
- Lukas Neuner, 2018,
  Collaboration with Hocoma AG, Volketswil, Switzerland.
- Peter Worsnop, 2017,
  Collaboration with UC Berkeley, co-supervised with Aaron Bestick.

Semester Projects

- Sophie Hall, 2020.
- Kristof Descotes, 2019.
- Ueli Wechsler, 2019,
  Co-supervised with Lukas Hewing.
- Riccardo Schira, 2018.

Bachelor's Theses

- Jan Schilliger, 2017.

Studies on Mechatronics

- Clara Baumhauer, 2016.
- Jasmine Belfrage, 2016.

Chapter III

Discussion & Future Work

Novel paradigms in machine learning and the availability of computational resources give rise to innovative concepts to automate the design and calibration of feedback controllers. This chapter discusses possible future directions for learning control objectives. The methodologies presented in this thesis fuse model-based architectures grounded in control theory with the flexibility of data science, under the assumption that the data originate from a certain model class, at least approximately. Weakening this assumption offers a variety of research ideas for principled theoretical research, as well as for applications.

Quantification of Suboptimalities

Suboptimalities in the form of noise or inexact execution are a core component and one of the main challenges in learning from (human-generated) data. While assumptions made on the model class and on the suboptimalites reduce the amount of data needed for the learning, specific choices of the assumptions may significantly deteriorate the results. A possible extension of the proposed methods is to not only learn the parameters but also the noise distribution of the data, and to identify the sources of suboptimalities. While such an approach would require more data and substantially more computational resources, the imminent benefits are improved learning results and, in the context of learning from human interactions, the potential to incorporate the identified suboptimalities into the model of human behavior for making more accurate predictions.

Concrete use cases for such an extension are found in nearly all methods in this thesis. For example, the assumptions made on the distribution of the data in [P1] or [P2] could be revisited and the distribution could be learned from the data; the assumption that human motions can be modeled as resulting from a particle filter algorithm with only one particle, as in [P4] and [P7], could be generalized to an arbitrary number of particles; or inexact execution in the form of the planning horizon that humans use for their motions could be inferred from data, rather than assumed to be constant as in [P1], [P2], or [P5]. The latter example is particularly useful for motion planning problems in autonomous driving applications, such as [P10], in order to make more accurate predictions of human-driven vehicles, i.e., by learning how far ahead do humans plan their actions.

Flexibility of Model Classes & Generalization Properties

For automatic feedback controllers with a desired and fixed structure, the methods proposed as part of this thesis offer powerful tools for their calibration. However, for the operation of some complex dynamical systems or robots, the assumption of a fixed controller structure might limit the achievable performance. While other paradigms such as imitation learning [21] can yield superior flexibility in learning, they often come at the price of less compelling generalization properties. A potential future research direction for learning control objectives is to examine model classes that allow greater flexibility, while not sacrificing properties required for employing the feedback controller, e.g., generalization to tasks different from the demonstrated one.

Concrete examples of more flexible model classes are Gaussian processes or kernel-based models [22], whose complexity and flexibility scale with the number of data points. Gaussian processes are very flexible function approximators, however they are not trivially usable as the model class for control objectives as their shape does not necessarily facilitate the feedback controller, e.g., they have many local optima. A possible remedy is to attempt to constrain the shape of the Gaussian process to promote its usability for feedback control, e.g., by constraining the mean of the Gaussian process to be convex or concave. This idea of constrained Gaussian processes has gained some attention recently, e.g., in [23]–[25], but there is still great research potential for designing control objectives. Alongside the flexible approximation properties, Gaussian processes provide an intrinsic uncertainty quantification, which in turn could also be used for feedback control. A relevant paradigm is to incorporate shape constraints into a Bayesian optimization framework. Fig. III-1 shows a toy example of a goal expressed as a reward function, $y = \max(0, -16x(x - 0.5))$, that is not easy to use as a control objective, e.g., due to a vanishing gradient for $x > 0.5$. It shows three approximations of the reward function using (a) an unconstrained Gaussian process, (b) a concavely-constrained Gaussian process, and (c) a pseudo-concavely-constrained Gaussian process. The approximation in (a) is difficult to use as a control objective due to a local optimum at $x > 0.5$, however, both (b) and (c) are suitable control objectives, where (c) approximates the goal well. This toy example illustrates the approximation capabilities of a more flexible model class, along with an uncertainty quantification with the standard deviation. Such a model class could be useful to better approximate abstract goals such as the one in [P3], where the uncertainty quantification provides information about the level of exploration of the state space.

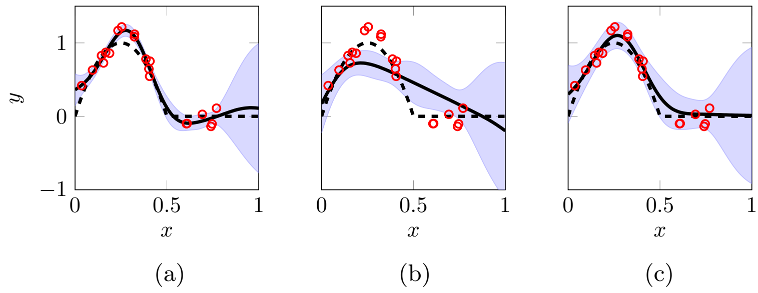


Figure III-1. Example of flexible model class for control objectives. The true reward function (dashed black) is approximated with three versions of Gaussian processes (mean: solid black, standard deviation: shaded blue). The approximations are learned using samples (red) of the true reward function.

Addendum to Paper P1

Related Work on Manipulation Tasks

Manipulation tasks have been investigated in various forms, and the approaches can be categorized as reactive, adaptive, or predictive [26], [27]. A reactive approach, e.g., [28]–[30], decouples the manipulation task and treats human inputs as a disturbance to the system. In an adaptive approach, e.g., [31], [32], the robot learns a control objective or task model over multiple interactions, treating human inputs as corrective perturbations of its own state. A predictive approach, e.g., in [33]–[37], learns a model for the human's actions enabling an optimal interaction through predictions. For example, [33] uses a predictive model to accommodate human movement in co-manipulation, whilst employing a reactive controller to deal with errors in predictions. In [34], a method is developed for optimizing robot-to-human handover poses for predicted human behavior given an ergonomic objective function. In [35], path-integral inverse reinforcement learning is applied to human-human collaboration and [36] applies an inverse optimal control approach with probabilistic movements in order to predict human reaching motion online.

Related learning approaches that are often applied to identifying human preferences or predicting human movements are reinforcement learning, e.g., [38], inverse reinforcement learning, e.g., [15], [39]–[41], or apprenticeship learning, e.g., [42]–[44]. For example, [45] shows that the training time with human supervision can be effectively reduced by introducing relative ratings, i.e., pairwise comparisons of trajectory segments. In [46], an imitation learning technique based on probabilistic movement primitives is proposed and applied to human-robot interaction. In [47], human-robot interaction policies are constructed for a scenario with multiple highly distinct future outcomes in decision making. Game-theoretic approaches are examined in [48], [49]. In [48], possible utilities motivating the agents' equilibrium behavior are identified in an inverse game theory framework and [49] presents a cooperative inverse reinforcement learning approach.

Practical Considerations of Inverse Optimal Control

The manipulation task in [P1] was included to showcase the inverse optimal control algorithm along with the constraint learning procedure. The study indicated that even with such a rather simple control objective, the human arm movements in such a manipulation task are expected to be individual and thus, the individuality of human subjects should be considered when making predictions. However, for more complex robotic tasks, the control objective considered in [P1] may not model the task sufficiently well in order to make accurate predictions. Instead, the model may only be valid for making short-term predictions and it may be beneficial to consider more sophisticated model classes, e.g., hybrid or multiple models, which in turn will increase the complexity of the inverse optimal control algorithm. However, the shortest path approach in [P1] may be useful when considering a multiple-model task because it allows for decoupling the task into subtasks, where optimality of the complete multiple-model task implies that each segment (a subtask) is optimal with respect to a shortest path formulation. The difficulty for such a learning procedure is that this decoupling requires the knowledge of the transition point between the single models, e.g., by using a detection procedure. Note that for the considered setup with a single prediction model, the learned constraints may not necessarily be a bound on the physical limitations of the human subjects, but a bound on the validity of the prediction model.

Scalability of Candidate Constraint Sets

Often convex hulls (as a collection of half-space constraints) scale badly with increased data sizes or higher state spaces—especially in the presence of noisy data. As a consequence, the algorithm to learn constraints from a pre-defined candidate constraint set as in [P1] may become infeasible for some tasks. Possible remedies are i) to remove certain half-spaces, e.g., based on area or volume criteria, in order to reduce the number of half-spaces, ii) utilize *box* constraints that use maximum and minimum values to define constraints for each dimension independently, or iii) use ellipsoidal constraints, e.g., a minimum volume ellipsoid. Table III-1 revisits Table P1-2 and shows the error metric as in (P1-17) for the three different model classes of constraint sets, i.e., the convex hull, box constraints, and ellipsoidal constraints. Compared to making predictions with an unconstrained optimal control formulation, the prediction accuracy increases with both the box and the ellipsoidal constraints, although not as much as with the convex hull. This suggests that the prediction accuracy and the computational effort can be traded off with such an optimal control formulation.

Table III-1. Prediction errors: Unconstrained vs. constrained (with different model classes)

	unconstrained	constrained		
		convex hull	box	ellipsoid
Subject 1	0.96°	0.78°	0.89°	0.90°
Subject 2	3.26°	2.45°	3.03°	3.19°
Subject 3	1.87°	1.56°	1.78°	1.80°

Implementation Details: Gradient Computation of Kinematic Model

For the manipulation task considered in [P1], the derivative of the system dynamics in (P1-1) with respect to the input is obtained as

$$\frac{\partial f(x(k),u(k))}{\partial u(k)} = T_s \begin{bmatrix} I \\ J_o^{\ddagger}(x_o(k))J_h(x_h(k)) \end{bmatrix}.$$

The derivative with respect to the state is more involved with

$$\frac{\partial f(x(k),u(k))}{\partial x(k)} = I + T_s \begin{bmatrix} 0 & 0 \\ f_h(x(k),u(k)) & f_o(x(k),u(k)) \end{bmatrix}$$

and the two partial derivatives

$$f_h(x(k), u(k)) = \frac{\partial J_o^\ddagger(x_o(k))J_h(x_h(k))u(k)}{\partial x_h(k)} \in \mathbb{R}^{4\times 4} \tag{3}$$

$$f_o(x(k), u(k)) = \frac{\partial J_o^\ddagger(x_o(k))J_h(x_h(k))u(k)}{\partial x_o(k)} \in \mathbb{R}^{4\times 4}. \tag{4}$$

Eq. (3) can be computed analytically or numerically with widely available numerical differention software such as the Symbolic Math Toolbox in MATLAB [50]. Eq. (4) cannot be computed symbolically as it involves the derivative of a pseudo-inverse with respect to its argument, which does not yield a closed analytical solution. In order to address this issue, we utilize the chain rule and techniques from tensor calculus and the derivative of the pseudo-inverse and compute (4) as

$$\underbrace{\frac{\partial J_o^\ddagger(x_o(k))J_h(x_h(k))u(k)}{\partial J_o(x_o(k))}}_{\in\mathbb{R}^{4\times 6\times 4}} : \underbrace{\frac{\partial J_o(x_o(k))}{\partial x_o(k)}}_{\in\mathbb{R}^{6\times 4\times 4}},$$

where the operator : is defined in the following.

Tensor Calculus Two fourth order tensors ϵ and τ are defined as

$$\begin{aligned} \epsilon_{ijkl} &= \delta_{ik}\delta_{jl} \\ \tau_{ijkl} &= \delta_{il}\delta_{jk}, \end{aligned} \tag{5}$$

where δ_{ij} is the Kronecker delta [51]. The operator : of two tensors A and B is defined as

$$A : B \;\Leftrightarrow\; A_{ijkl}B_{klmn}$$

and the tensor product AB indicates

$$AB \;\Leftrightarrow\; A_{ijkl}B_{lmno},$$

where we use the Einstein summation convention as summation over a set of indexed terms in an expression, i.e.

$$A_{ijkl}B_{klmn} := \sum_{k,l} A_{ijkl}B_{klmn}.$$

Eq. (5) can be used to write the following two identities:

$$AXB = A\epsilon B^\top \;:\; X \tag{6a}$$

$$X^\top = \tau \;:\; X. \tag{6b}$$

Differential of Pseudo-Inverse The derivative of the pseudo-inverse $P = \left(A^\top A\right)^{-1} A^\top$ with respect to the original tall matrix A yields a fourth order tensor and is derived using the differential $\mathrm{d}P$:

$$\mathrm{d}P = PP^\top \mathrm{d}A^\top (I - AP) + (I - PA)\mathrm{d}A^\top P^\top P - P\mathrm{d}AP, \tag{7}$$

cf. [52]. Using the introduced tensor ϵ in (5), the differential (7) can be written as

$$\mathrm{d}P = \Big(PP^\top \epsilon (I - AP)^\top + (I - PA)\epsilon P^\top P\Big) : \mathrm{d}A^\top - P\bar{\epsilon}P : \mathrm{d}A,$$

cf. (6a), where $\bar{\epsilon}$ and ϵ are fourth order tensors as in (5) of different dimensions. Further, τ in (5) is used to write

$$\mathrm{d}P = \Big(\Big(PP^\top \epsilon (I - AP)^\top + (I - PA)\epsilon P^\top P\Big) : \tau - P\bar{\epsilon}P\Big) : \mathrm{d}A,$$

cf. (6b), and thus, the derivative of the pseudo-inverse P with respect to A results into

$$\frac{\mathrm{d}P}{\mathrm{d}A} = \Big(PP^\top \epsilon (I - AP)^\top + (I - PA)\epsilon P^\top P\Big) : \tau - P\bar{\epsilon}P. \tag{8}$$

Addendum to Paper P2

Scalability of Algorithm P2-1

The computational complexity of Algorithm P2-1 highly depends on the considered problem setup, i.e., how many constraints can be discarded in Step 2, how many constraint combinations are feasible in the set $\mathcal{D}$ in Step 3, and how quickly the stopping criterion is satisfied in Step 5.

For learning tasks where the data are expected to be very noisy or where the problem setup has many different constraints in the vicinity of the data, the algorithm might scale badly as constraints may not be discarded in Step 2 and the stopping criterion might not be satisfied early. Note that the number of feasible constraint combinations in Step 3 is independent of the noise in the data. If the algorithm is expected to be computationally too expensive, the index set can be fixed, e.g., similarly to Method 1 in [P2]. Then, Algorithm P2-1 reduces to solving the optimization problem in (P2-7) with a fixed index set rather than $\mathrm{idx} \in \{0, 1\}^s$.

Chapter IV

Overview of Inverse Learning Concepts

This chapter provides an extended literature review and contrasts different methodologies used for learning control objectives from data. It considers the problem of learning a (parametrization of a) control objective for optimal control formulations.

1 Formal Problem Statement

We consider dynamical systems, represented in discrete time, of the form

$$x_{k+1} = f(x_k, u_k), \tag{3}$$

where $x_k \in R^{n_x}$ is the state at time k and $u_k \in R^{n_u}$ is the control input. This model for the dynamics is used to predict the evolution of the system in an optimal control framework with

$$\begin{aligned} \max_{x_k, u_k} \quad & r_\theta(x_0, ..., x_T, u_0, ..., u_T) && \text{(4a)} \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k) && \text{(4b)} \\ & c(x_k, u_k) \leq 0, \quad \forall k = 0, ..., T, && \text{(4c)} \end{aligned}$$

where T is the prediction horizon, $c(x_k, u_k) \leq 0$ defines constraints for the state, x_k, and the control input, u_k, and

$$r_\theta(x_0, ..., x_T, u_0, ..., u_T) = \sum_{k=0}^{T} l_{\theta,k}(x_k, u_k) \tag{5}$$

is the control objective expressed as a reward function, with the parameters θ, where $l_{\theta,k}(x_k, u_k)$ is called the stage reward. Using the model for the system dynamics in (3), we can express the reward as a function of the control inputs, u_k for times $k = 0, ..., T$, and the initial state, x_0. In order to ease exposition, we therefore define the shorthand notation

$$r_\theta(x_0, U) := r_\theta(x_0, ..., x_T, u_0, ..., u_T),$$

where $U = [\ u_0^T,\ u_1^T,\ \dots,\ u_T^T\ ]^T$ denotes the input sequence.

Notation

Given mean vector μ and covariance matrix Σ, $p(x|\mu, \Sigma)$ denotes the probability density function of the Gaussian distribution, with

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp\left(-\tfrac{1}{2}\,(x-\mu)^T\,\Sigma^{-1}\,(x-\mu)\right).$$

Further, $\propto$ reads "proportional to." We define I as the identity matrix of suitable dimension, vec as the vectorization operator, and $\|x\|_\Sigma := x^T \Sigma x$.

Problem Definition

The learning methodologies in this chapter aim to find a parametrization of the reward function using data in the form of demonstrations. In Bayesian statistics terms, the learning methods maximize the posterior distribution of the parameters, θ, i.e., $p\,(\theta|\text{data}) \propto p\,(\text{data}|\theta)\,p(\theta)$, where $p\,(\text{data}|\theta)$ is called the likelihood (of the data given the parameters) and $p(\theta)$ denotes the prior likelihood. Note that for computational reasons, learning methods typically optimize the logarithmic likelihood (log-likelihood). Formally, the inverse problem can be stated as

$$\max_\theta\ \log(p\,(\text{data}|\theta)\,p(\theta)) = \log\ p\,(\text{data}|\theta) + \log\ p(\theta) \tag{6a}$$

$$\text{s.t. } \theta \in \Theta. \tag{6b}$$

Some learning methodologies restrict the parameters to satisfy certain properties, which can be encoded as $\theta \in \Theta$. In this context, the data are control inputs and state trajectories of the form

$$\left\{\left\{x_0^{(1)}, ..., x_T^{(1)}, u_0^{(1)}, ..., u_T^{(1)}\right\},\ \dots, \left\{x_0^{(D)}, ..., x_T^{(D)}, u_0^{(D)}, ..., u_T^{(D)}\right\}\right\},$$

which we concisely express, using the control inputs and the initial state, as

$$\mathcal{D} = \left\{\left\{x_0^{(1)}, U^{(1)}\right\},\ \left\{x_0^{(2)}, U^{(2)}\right\},\ \dots, \left\{x_0^{(D)}, U^{(D)}\right\}\right\},$$

where $\{x_0^{(i)}, U^{(i)}\}$ denotes the ith demonstrated trajectory and D is the total number of demonstrated trajectories. For ease of exposition, we develop the remaining chapter for one demonstrated trajectory, $\{x_0, U\}$. The extension to multiple demonstrated trajectories is straightforward for all methods, by multiplication of the likelihoods (addition of the logarithmic likelihoods).

The learning methodologies presented in this chapter are categorized in Fig. IV-1 and differ by virtue of the model for the likelihood term $p\,(\text{data}|\theta)$, as well as the assumptions on the valid parameter space, Θ. Inverse reinforcement learning methodologies in the literature tend to focus on maximizing the observed rewards, while inverse optimal control methodologies typically utilize optimality conditions for learning control objectives. In the following, we describe the learning methods in the literature as well as in this thesis with respect to the problem definition in (6). Note that the papers included in this chapter introduce the specific learning concepts. An extended discussion of related work that builds on these concepts is omitted. Furthermore, branches in the literature that focus on problems different from the setup introduced in this section, e.g., discrete state spaces, are also omitted.

Discussion of the prior

Frequent choices for the prior likelihood are the Laplace distribution, commonly referred to as Lasso regression; the Gauss distribution, commonly referred to as Ridge regression; and $p(\theta) = \text{constant}$, for which (6) is referred to as maximum likelihood estimation [60]. For example, [53] uses a Laplace distribution, $p(\theta) \propto \prod_i \exp(-\lambda|\theta_i|)$, and [P4] uses a normal distribution, $p(\theta) \propto \prod_i p(\theta_i|\theta_i^{\text{prior}}, \sigma_p^2)$. In what follows, we use $p(\theta) = \text{constant}$ for all learning methodologies to ease exposition.

2 Inverse Reinforcement Learning

Inverse reinforcement learning methods tend to focus on maximizing the demonstrated reward and less on the structure of the (constrained) optimal control problem in (4). In the following, we discuss three conceptually different ideas that classify as inverse reinforcement learning. Note that inverse reinforcement learning methodologies are often stated for Markov decision processes with a discrete state and action space. However, the model for the learning proposed in the respective methods can similarly be applied for the setup considered in this chapter, i.e., as a model for the likelihood and the valid parameter space in (6).

2.1 Matching Feature Expectation

In [42], a learning method is proposed that is based on the idea that demonstrations yield the highest possible reward, given the structure of

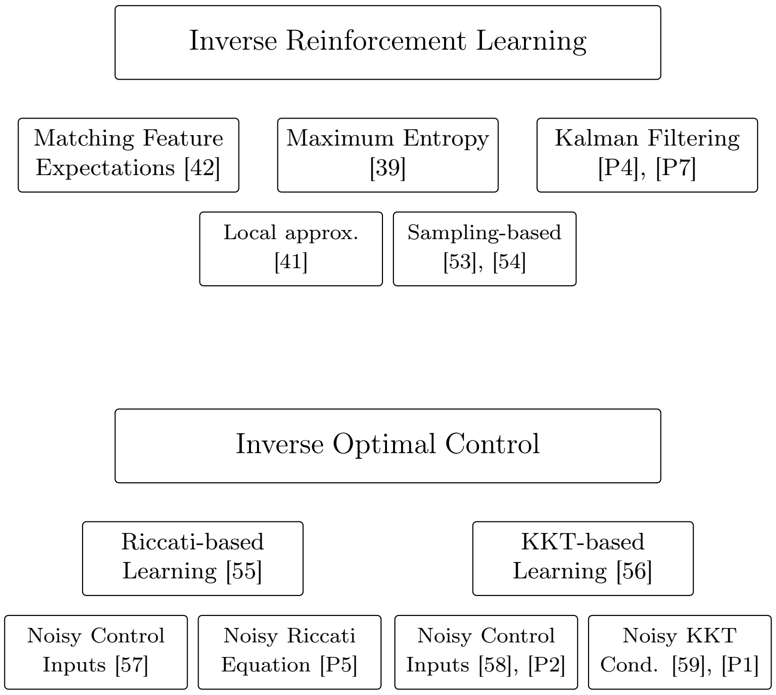

Inverse Reinforcement Learning
Matching Feature Expectations [42]
Maximum Entropy [39]
Kalman Filtering [P4], [P7]
Local approx. [41]
Sampling-based [53], [54]
Inverse Optimal Control
Riccati-based Learning [55]
KKT-based Learning [56]
Noisy Control Inputs [57]
Noisy Riccati Equation [P5]
Noisy Control Inputs [58], [P2]
Noisy KKT Cond. [59], [P1]

Figure IV-1. Overview and categorization of inverse reinforcement learning and inverse optimal control methodologies in the literature and proposed as part of this thesis.

the parametrized reward function. Here, the reward function $r_\theta(x_0, U) = \theta^T \phi(x_0, U)$ is expressed in terms of the feature vector $\phi(x_0, U)$. The goal is to find a policy, π_θ with $\|\theta\|_2 \leq 1$, i.e., a parametrization of the reward function, that minimizes the deviation to the demonstrated features: $|\theta^T \phi(x_0, \pi_\theta) - \theta^T \phi(x_0, U)| \leq \epsilon$, where $\{x_0, U\}$ is the demonstrated trajectory. Using the notation in (6), the matching feature expectations approach in [42] uses

$$\begin{aligned} p(\text{data}|\theta) &\propto \exp\Big(-|\theta^T \phi(x_0, U) - \theta^T \phi(x_0, \pi_\theta)|\Big) \\ \Theta &= \big\{\theta \;\big|\; \|\theta\|_2 \leq 1,\ \theta^T \phi(x_0, U) \geq \theta^T \phi(x_0, \pi_\theta)\big\}. \end{aligned}$$

As $\theta^T \phi(x_0, \pi_\theta)$ is nonlinear in the parametrization of the reward function, θ, [42] uses an iterative algorithm, where at iteration j, the following optimization problem is solved:

$$\{t^{(j)}, \theta^{(j)}\} = \arg\max_{t,\theta}\ t \tag{7a}$$

$$\text{s.t. } \theta^T \phi(x_0, U) \geq \theta^T \phi(x_0, \pi_{\theta^{(i)}}) + t,\ i = 0, ..., j-1 \tag{7b}$$

$$\|\theta\|_2 \leq 1, \tag{7c}$$

where the feature vector, $\phi(x_0, \pi_{\theta^{(i)}})$, is computed with the parameters $\theta^{(i)}$. In other words, the algorithm alternates between finding the parameters $\theta^{(i)}$ and the feature calculation, $\phi(x_0, \pi_{\theta^{(i)}})$. The iterative algorithm terminates when $t^{(j)} \leq \epsilon$.

Expected advantages & limitations. *The main advantages of this learning method are the low computational requirements and its easy implementation. Furthermore, the iterative implementation offers the advantage that constraints can be considered in the feature computation. The main limitation of this method is that suboptimal behavior is not explicitly considered and may corrupt the learning results. The only design choice to be made is the trajectory segment that is used to learn the parameters.*

2.2 Maximum Entropy

In [39], a model for the likelihood term in (6) is proposed that is based on the concept of entropy, with

$$p(\text{data}|\theta) = \frac{\exp\left(r_\theta(x_0, U)\right)}{Z_\theta} \tag{8}$$

and the partition function

$$Z_\theta = \int_{\mathcal{U}} \exp\Big(r_\theta(x_0, \tilde{U})\Big)\, \mathrm{d}\tilde{U}$$

if the control input space, $\mathcal{U}$, is continuous, which is the considered setup. If the control input space is discrete, $Z_\theta = \sum_{\tilde{U}\in\mathcal{U}} \exp(r_\theta(x_0, \tilde{U}))$. This model for the likelihood term has the advantage of interpreting the reward of the observed data as a sample from a probability distribution of an exponential family. However, maximizing (6) with the likelihood term (8) is often intractable due to the partition function, Z_θ, and the following methods use different approximations of Z_θ.

Local approximation of reward function

In [41], a second-order approximation of the reward function in (5) is used in order to tractably approximate the partition function, with

$$r_\theta(x_0, \tilde{U}) \approx r_\theta(x_0, U) + (\tilde{U} - U)^T g_\theta + \frac{1}{2}(\tilde{U} - U)^T H_\theta (\tilde{U} - U),$$

where

$$g_\theta = \left.\frac{\partial r_\theta(x_0, \tilde{U})}{\partial \tilde{U}}\right|_{\tilde{U}=U}, \quad H_\theta = \left.\frac{\partial^2 r_\theta(x_0, \tilde{U})}{\partial \tilde{U}^2}\right|_{\tilde{U}=U}.$$

Then, the partition function in (8) is approximated locally, around the observed trajectory, and the likelihood term in (6) is given by

$$\begin{aligned} p(\text{data}|\theta) &= \frac{\exp(r_\theta(x_0, U))}{\int_{\mathcal{U}} \exp(r_\theta(x_0, \tilde{U}))\mathrm{d}\tilde{U}} \\ &\approx \exp\left(\frac{1}{2} g_\theta^T H_\theta^{-1} g_\theta\right) |-H_\theta|^{\frac{1}{2}} (2\pi)^{-\frac{n_u T}{2}}. \end{aligned}$$

The inverse problem is given by maximizing the log-likelihood, using the second-order approximation of the reward function,

$$\max_\theta \ \frac{1}{2} g_\theta^T H_\theta^{-1} g_\theta + \frac{1}{2}\log|-H_\theta| - \frac{n_u T}{2}\log(2\pi). \qquad (9)$$

Expected advantages & limitations. *For linear systems, i.e.,* $f(x_k, u_k) = A_k x_k + B_k u_k$ *in* (3) *without constraints and a quadratic reward function,* (9) *yields the exact solution to* (8)*. Consequently, the learning results are expected to be limited by the approximation accuracy of the reward function, i.e., for nonlinear systems and in the presence of constraints, the results may deteriorate. An additional advantage is that the only design choice to be made is the trajectory segment used for learning.*

Local sampling-based approximation

Rather than using a local approximation of the reward function, in [53], the partition function, Z_θ, is approximated by K samples in the vicinity of the observed demonstration, denoted $U^{(k)}$. Thus, the likelihood term in (6) is given by

$$p(\text{data}|\theta) = \frac{\exp\left(r_\theta\left(x_0, U\right)\right)}{\sum_{k=1}^{K} \exp\left(r_\theta\left(x_0, U^{(k)}\right)\right)}.$$

The inverse problem is given by maximizing the log-likelihood,

$$\max_\theta \; r_\theta\left(x_0, U\right) - \log \sum_{k=1}^{K} \exp\left(r_\theta\left(x_0, U^{(k)}\right)\right). \tag{10}$$

Expected advantages & limitations. *The main advantage of this learning method is its relatively easy implementation. The learning results are expected to depend on the choice of the sampling distribution that is used to generate the trajectory samples as an approximation of the partition function. Furthermore, constraints are not explicitly considered in this learning formulation. Design choices to be made are the sampling distribution, the number of samples, as well as the trajectory segment used for learning.*

Guided sampling-based approximation

Similarly to [53], [54] uses samples to approximate the partition function, Z_θ. This method, in contrast, suggests that the samples are generated from a distribution $q \propto \exp(r_{\theta_\text{t}}(x_0, U))$, defined with the true reward function parameters, θ_t. However, since θ_t is unknown, [54] samples adaptively from the distribution with the current best estimate of the parameters, $\exp(r_{\theta_l}(x_0, U))$, while optimizing for the parameters. This allows for gradual improvement of the sampling distribution—and the partition function along with it—while optimizing for the parameters. This method is targeted to a class of control objectives with a nonlinear parametrization, e.g., a neural network, for which other sampling-based learning methods may deteriorate.

For this method, the likelihood term in (6) is chosen as

$$p(\text{data}|\theta) = \frac{\exp\left(r_\theta\left(x_0, U\right)\right)}{\frac{1}{K}\sum_{k=1}^{K} z_k \exp\left(r_\theta\left(x_0, U^{(k)}\right)\right)},$$

where K is the total number of sampled trajectories and z_k is the importance weight of the sampled trajectory k. The weights are computed as $z_k = (\frac{1}{L}\sum_{l=1}^{L} q_l(x_0, U^{(k)}))^{-1}$, where $q_l(\cdot)$ with $l = 1, ..., L$ are the L sampling distributions, chosen as $q_l(x_0, U^{(k)}) \propto \exp(r_{\theta_l}(x_0, U^{(k)}))$, i.e., the reward function with parameters θ_l that are used to generate the samples. Finally, maximizing the log-likelihood using this guided sampling-based method yields the following inverse problem:

$$\max_{\theta} \; r_\theta \left(x_0, U\right) - \log \frac{1}{K} \sum_{k=1}^{K} z_k \exp\left(r_\theta\left(x_0, U^{(k)}\right)\right). \tag{11}$$

Expected advantages & limitations. *This version of maximum entropy inverse reinforcement learning is expected to yield very robust learning results due to the adaptive sampling of trajectories. The sampling-based partition function is expected to converge to the true partition function as $K \to \infty$. The main limitations of the learning method are the demand for computational resources and that constraints for the system operation are not explicitly considered in the learning procedure. For the application of [54], one needs to choose the segment used for learning, the times or iterations at which to sample new trajectories, as well as the number of newly sampled trajectories, e.g., N new trajectories sampled from $q_l(x_0, U^{(k)})$ with the current parameters, θ_l.*

2.3 Kalman Filtering

This learning method has been proposed in [P7], and detailed in [P4], for a class of reward functions that can be formulated using positive definite quadratic forms. It is based on the observation that such a reward function can similarly be expressed using probability density functions of Gausian distributions, with

$$r_\theta(x_0, U) = \sum_{k=0}^{T} -\|v_k\|_{Q_\theta} - \|u_k\|_{R_\theta} \propto \log \prod_{k=0}^{T} p\left(v_k|0, Q_\theta^{-1}\right) p\left(u_k|0, R_\theta^{-1}\right)$$

with $v_k = y_{\mathrm{ref},k} - h(x_k)$, where $y_{\mathrm{ref},k}$ denotes the reference and h is the function mapping the state to the reference. Then, modeling v_k and u_k as random variables, the optimal control problem in (4) can be interpreted as a statistical estimation problem in a Kalman filter framework, where $y_{\mathrm{ref},k}$ is treated as a sensor measurement. As a result, (4) can be solved using particle filtering rather than gradient-based numerical optimization. This learning method models the data to be the result of particle filtering with one particle, for which the likelihood term and the valid parameter space are given by

$$p(\text{data}|\theta) = \prod_{k=1}^{T} p(v_k|0, Q_\theta^{-1}) p(u_k|0, R_\theta^{-1}) \frac{p(w_k|0, Q_\theta^{-1})}{p(e_k|0, \Gamma_{\theta,k})}$$
$$\Theta = \{\theta \;| Q_\theta \succeq \epsilon_Q \cdot I, R_\theta \succeq \epsilon_R \cdot I\}$$

with some $\epsilon_Q, \epsilon_R > 0$, $e_k = y_{\mathrm{ref},k} - h(f(x_{k-1}, 0))$, $w_k = e_k - J_k u_k$, $\Gamma_{\theta,k} = Q_\theta^{-1} + J_k^T R_\theta^{-1} J_k$, and $J_k = H_k G_k$, where $H_k = \frac{\partial h(x)}{\partial x}|_{x=x_k}$ and $G_k = \frac{\partial f(x,u)}{\partial u}|_{x=x_k,u=u_k}$, see [P7] and [P4] for details. Then, maximizing the log-likelihood results in the following inverse problem:

$$\max_\theta \; -\sum_{k=1}^{T} \Big(\|v_k\|_{Q_\theta^{-1}} + \|u_k\|_{R_\theta^{-1}} + \|w_k\|_{Q_\theta^{-1}} - \|e_k\|_{\Gamma_{\theta,k}} \Big) \tag{12a}$$

$$\text{s.t. } Q_\theta \succeq \epsilon_Q \cdot I, R_\theta \succeq \epsilon_R \cdot I. \tag{12b}$$

Expected advantages & limitations. *The main advantages of the method are that the data do not have to be segmented into specific trajectories prior to learning and that there are no design choices to be made. The main limitation of this method is the class of control objectives (quadratic form). Furthermore, constraints for the system operation are not explicitly considered in the learning procedure.*

3 Inverse Optimal Control

Differently from inverse reinforcement learning methodologies, inverse optimal control utilizes optimality conditions of the corresponding optimal control problem. Considering the general problem statement in (6), the optimality conditions are stated by means of the allowed parameter space, Θ. For example, if the corresponding optimal control problem is stated for unconstrained, linear systems and a quadratic control objective, i.e., a linear quadratic regulator (LQR) problem, then the inverse problem utilizes the Riccati equation; if the corresponding optimal control problem is stated for nonlinear and/or constrained systems with a potentially more general control objective, then the inverse problem utilizes the Karush-Kuhn-Tucker (KKT) conditions. For learning using both, Riccati equation and KKT conditions, we present one convex and one nonconvex variant in what follows, where the difference lies in the interpretation of the noise in the data.

3.4 Learning based on Riccati Equation

If $f(x_k, u_k) = Ax_k + Bu_k$ in (3) and the control objective is $r_\theta(x_0, U) = \sum_{k=0}^{\infty} -x_k^T Q x_k - u_k^T R u_k$ with $Q \succeq \epsilon_Q \cdot I$ and $R \succeq \epsilon_R \cdot I$ in (5), then the optimal policy is given by $u_k = Kx_k$ [19], where K is called the feedback gain and satisfies

$$\begin{aligned}(B^T PB + R)K + B^T PA = 0\\ A^T PA - P + A^T PBK + Q = 0\end{aligned}$$

with some $P \succ 0$.

Noisy control inputs (nonconvex formulation)

In this nonconvex formulation, proposed in [57], the likelihood term is given by the difference of the observed control inputs and the policy, and the parameter space Θ spans all admissible θ satisfying the Riccati equation:

$$\begin{aligned} p(\mathrm{data}|\theta) &= \prod_{k=0}^{T} p(u_k|Kx_k,\Sigma) \\ \Theta = \big\{\theta \mid\ &(B^T P_\theta B + R_\theta)K + B^T P_\theta A = 0, \\ &A^T P_\theta A - P_\theta + A^T P_\theta BK + Q_\theta = 0, \\ &P_\theta \succeq \epsilon_P \cdot I,\ Q_\theta \succeq \epsilon_Q \cdot I,\ R_\theta \succeq \epsilon_R \cdot I\big\} \end{aligned}$$

with some $\epsilon_P, \epsilon_Q, \epsilon_R > 0$. The resulting learning problem is given by

$$\max_{\theta,K} \sum_{k=0}^{T} -\|u_k - Kx_k\|_{\Sigma^{-1}} \tag{13a}$$

$$\text{s.t. } (B^T P_\theta B + R_\theta)K + B^T P_\theta A = 0 \tag{13b}$$

$$A^T P_\theta A - P_\theta + A^T P_\theta BK + Q_\theta = 0 \tag{13c}$$

$$P_\theta \succeq \epsilon_P \cdot I,\ Q_\theta \succeq \epsilon_Q \cdot I,\ R_\theta \succeq \epsilon_R \cdot I. \tag{13d}$$

Expected advantages & limitations. *The first advantage of the method is that its implementation does not require segmentation of the trajectory. The main limitation of this method, by design, is the class of systems and the control objective. For nonlinear systems, it is possible to linearize the system dynamics, but the performance may deteriorate. Furthermore, operational constraints are not explicitly considered in the learning procedure. The design choice to be made is the covariance matrix of the noise distribution, Σ.*

Noisy optimality condition (convex formulation)

An alternative convex formulation to (13) is to assume that the Riccati equation is only fulfilled approximately, where the policy, K, is obtained in a first step [P5]. Then, the likelihood term is stated in terms of noise in the Riccati equation, i.e., Λ_1, Λ_2:

$$\begin{aligned} p(\mathrm{data}|\theta) &= p\left(\mathrm{vec}(\Lambda_1)|0,\Sigma_1\right) p\left(\mathrm{vec}(\Lambda_2)|0,\Sigma_2\right) \\ \Theta(\Lambda_1,\Lambda_2) = \big\{\theta \mid\ &(B^T P_\theta B + R_\theta)K + B^T P_\theta A = \Lambda_1, \\ &A^T P_\theta A - P_\theta + A^T P_\theta BK + Q_\theta = \Lambda_2, \\ &P_\theta \succeq \epsilon \cdot I,\ Q_\theta \succeq \epsilon \cdot I,\ R_\theta \succeq \epsilon \cdot I, \\ &K = \arg\min_{\tilde{K}} \textstyle\sum_{k=0}^{T} \|u_k - \tilde{K}x_k\|_{\Sigma_u^{-1}}\big\}. \end{aligned}$$

Maximizing the log-likelihood yields the learning problem, as a convex semi-definite program:

$$
\begin{aligned}
\max_{\theta,\Lambda_1,\Lambda_2} \quad & -\|\mathrm{vec}(\Lambda_1)\|_{\Sigma_1^{-1}} - \|\mathrm{vec}(\Lambda_2)\|_{\Sigma_2^{-1}} && \text{(14a)}\\
\text{s.t.} \quad & (B^T P_\theta B + R_\theta)K + B^T P_\theta A = \Lambda_1 && \text{(14b)}\\
& A^T P_\theta A - P_\theta + A^T P_\theta B K + Q_\theta = \Lambda_2 && \text{(14c)}\\
& P_\theta \succeq \epsilon_P \cdot I,\ Q_\theta \succeq \epsilon_Q \cdot I,\ R_\theta \succeq \epsilon_R \cdot I, && \text{(14d)}
\end{aligned}
$$

where $K = \arg\min_{\tilde{K}} \sum_{k=0}^{T} \|u_k - \tilde{K} x_k\|_{\Sigma_u^{-1}}$ is the least squares solution.

Expected advantages & limitations. *This method yields the same advantages and limitations as its nonconvex counterpart in* (13)*. Additionally, the convex formulation makes this approach computationally very efficient. Design choices to be made are the several covariance matrices,* Σ_1*,* Σ_2*, and* Σ_u*.*

3.5 Learning based on Karush-Kuhn-Tucker Conditions

For constrained, nonlinear systems and a more general control objective, the optimality conditions are given by the KKT conditions [61], which are first-order derivative tests for constrained optimization problems:

$$
\begin{aligned}
\nabla_{\tilde{U}} \left(r_\theta(x_0, \tilde{U}) + \lambda^T c(x_0, \tilde{U}) \right)\Big|_{\tilde{U}=U} &= 0\\
c(x_0, U) &\le 0\\
\lambda^T c(x_0, U) &= 0\\
\lambda &\ge 0,
\end{aligned}
$$

where λ are called the dual variables.

Noisy control inputs (nonconvex formulation)

Similarly to (13), the control inputs can be modeled as noisy, with $p(U|V, \Sigma)$, where V is an optimal control input sequence that satisfies the KKT conditions [58], [P2]:

$$
\begin{aligned}
p(\text{data}|\theta) &= p(U|V, \Sigma)\\
\Theta(V) &= \Big\{ \theta \ \Big|\ \nabla_{\tilde{U}} \left(r_\theta(x_0, \tilde{U}) + \lambda^T c(x_0, \tilde{U}) \right)\Big|_{\tilde{U}=V} = 0,\\
&\qquad c(x_0, V) \le 0,\ \lambda^T c(x_0, V) = 0,\ \lambda \ge 0 \ \Big\}.
\end{aligned}
$$

Maximizing the log-likelihood, the resulting learning problem is given by

$$\max_{\theta,\lambda,V} \; -\|U-V\|_{\Sigma^{-1}} \tag{15a}$$

$$\text{s.t. } \nabla_{\tilde{U}} \left(r_\theta(x_0,\tilde{U}) + \lambda^T c(x_0,\tilde{U}) \right)\Big|_{\tilde{U}=V} = 0, \tag{15b}$$

$$c(x_0,V) \leq 0, \; \lambda^T c(x_0,V) = 0, \; \lambda \geq 0. \tag{15c}$$

Expected advantages & limitations. *The main advantage of this learning method is the systematic treatment of operational constraints. The main limitation is the computational burden that comes with the nonconvex formulation. Design choices that have to be made are the segment for the learning and the covariance matrix of the noise distribution, Σ.*

Noisy optimality condition (convex formulation)

Rather than assuming that the noise manifests itself in the control inputs, it is also possible to assume that the optimality conditions are noisy rather than being satisfied perfectly [59], [P1]. This results in a convex formulation for reward functions that are linear in their parameters. The likelihood term and the parameter space are given by

$$\begin{aligned} p(\text{data}|\theta) &= p(\Lambda|0,\Sigma) \\ \Theta(\Lambda) &= \left\{ \theta \mid \nabla_{\tilde{U}} \left(r_\theta(x_0,\tilde{U}) + \lambda^T c(x_0,\tilde{U}) \right)\Big|_{\tilde{U}=U} = \Lambda, \right. \\ &\qquad\qquad \left. \lambda_{\text{idx}} = 0, \; \lambda \geq 0 \right\}, \end{aligned}$$

where idx indicates the index set of inactive constraints. Identifying the set of inactive constraints from noisy measurements is not trivial. However, an easily implementable option—used in [P2]—is to define idx as the set of constraints for which $c_i(x_0,U) < -\epsilon_i$, where c_i is the ith constraint. The resulting learning problem is given by

$$\max_{\theta,\lambda,\Lambda} \; -\|\Lambda\|_{\Sigma^{-1}} \tag{16a}$$

$$\text{s.t. } \nabla_{\tilde{U}} \left(r_\theta(x_0,\tilde{U}) + \lambda^T c(x_0,\tilde{U}) \right)\Big|_{\tilde{U}=U} = \Lambda \tag{16b}$$

$$\lambda_{\text{idx}} = 0, \; \lambda \geq 0. \tag{16c}$$

Expected advantages & limitations. *Similarly to* (15), *the main advantage of this formulation is the systematic treatment of constraints. The main limitation is that the index set* idx *of inactive constraints has to be fixed to obtain a convex formulation. Design choices that have to be made are the segment for the learning, the covariance matrix of the noise distribution, Σ, and the index set,* idx.

Bibliography for Chapters I–IV

[1] K. J. Åström and R. M. Murray, *Feedback systems: An introduction for scientists and engineers*. Princeton university press, 2010.

[2] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice–a survey", *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[3] P. S. Laplace, *Théorie analytique des probabilités*. Courcier, 1820.

[4] M. N. Zeilinger, "Real-time model predictive control", PhD thesis, ETH Zurich, 2011.

[5] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control", *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[6] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*. siam, 2005, vol. 89.

[7] K. S. Narendra and A. M. Annaswamy, *Stable adaptive systems*. Courier Corporation, 2012.

[8] H. Fukushima, T.-H. Kim, and T. Sugie, "Adaptive model predictive control for a class of constrained linear systems based on the comparison model", *Automatica*, vol. 43, no. 2, pp. 301–308, 2007.

[9] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression", *IEEE Trans. Control Syst. Technol.*, 2019.

[10] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. A data-driven control framework", *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 1883–1896, 2018.

[11] U. Rosolia, X. Zhang, and F. Borrelli, "Data-driven predictive control for autonomous systems", *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 1, no. 1, pp. 259–286, 2018.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars", *ArXiv:1604.07316*, 2016.

[13] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]", *IEEE Robot. Automat. Mag.*, vol. 14, no. 1, pp. 20–29, 2007.

[14] K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Okada, K. Matsumoto, M. Ishikawa, I. Shimoyama, and M. Inaba, "Home-assistant robot for an aging society", *Proc. of the IEEE*, vol. 100, no. 8, pp. 2429–2441, 2012.

[15] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning", in *17th Int. Conf. Mach. Learning*, 2000, pp. 663–670.

[16] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration", *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[17] W. Karush, "Minima of functions of several variables with inequalities as side constraints", Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[18] H. W. Kuhn and A. W. Tucker, "Nonlinear programming", in *2nd Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1951.

[19] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd. Belmont, MA: Athena Scientific, 2012.

[20] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles", *ROBOMECH J.*, vol. 1, no. 1, p. 1, 2014.

[21] S. Schaal, "Is imitation learning the route to humanoid robots?", *Trends in Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[22] C. E. Rasmussen and C. K. I. Williams, *GAUSSIAN Processes for Machine Learning.* The MIT Press, 2006.

[23] C. Jidling, N. Wahlström, A. Wills, and T. B. Schön, "Linearly constrained Gaussian processes", in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 1215–1224.

[24] H. Maatouk and X. Bay, "Gaussian process emulators for computer experiments with inequality constraints", *Math. Geosciences*, vol. 49, no. 5, pp. 557–582, 2017.

[25] C. Agrell, "Gaussian processes with linear operator inequality constraints", *J. Mach. Learning Res.*, vol. 20, pp. 1–36, 2019.

[26] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey", *Found. and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[27] B. D. Argall and A. G. Billard, "A survey of tactile human–robot interactions", *Robotics and Autonomous Syst.*, vol. 58, no. 10, pp. 1159–1176, 2010.

[28] T. Takubo, H. Arai, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint", *Int. J. Robotics Res.*, vol. 21, no. 5–6, pp. 541–553, 2002.

[29] D. Ćehajić and S. Hirche, "Estimating unknown object dynamics in human-robot manipulation tasks", in *IEEE Int. Conf. Robot. and Automat.*, 2017, pp. 1730–1737.

[30] D. Kruse, R. J. Radke, and J. T. Wen, "Human-robot collaborative handling of highly deformable materials", in *Amer. Control Conf.*, 2017, pp. 1511–1516.

[31] A. Capitanelli, M. Maratea, F. Mastrogiovanni, and M. Vallati, "On the manipulation of articulated objects in human-robot cooperation scenarios", *ArXiv preprint arXiv:1801.01757*, 2018.

[32] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time", in *ACM/IEEE Int. Conf. Human-Robot Interaction*, 2018, pp. 141–149.

[33] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion", in *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2013, pp. 299–306.

[34] A. Bestick, R. Bajcsy, and A. D. Dragan, "Implicitly assisting humans to choose good grasps in robot to human handovers", in *Int. Symp. on Experimental Robotics*, 2016, pp. 341–354.

[35] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning", in *IEEE Int. Conf. Robot. and Automat.*, 2015, pp. 885–892.

[36] O. S. Oguz, Z. Zhou, and D. Wollherr, "A hybrid framework for understanding and predicting human reaching motions", *Frontiers in Robotics and AI*, vol. 5, no. 1-19, p. 27, 2018.

[37] H. Ding, G. Reißig, K. Wijaya, D. Bortot, K. Bengler, and O. Stursberg, "Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction", in *IEEE Int. Conf. Robot. and Automat.*, 2011, pp. 5875–5880.

[38] R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

[39] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning", in *23rd AAAI Conf. Artif. Intell.*, vol. 8, 2008, pp. 1433–1438.

[40] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with Gaussian processes", in *Adv. Neural Inform. Process. Syst.*, 2011, pp. 19–27.

[41] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples", in *29th Int. Conf. Mach. Learning*, 2012, pp. 475–482.

[42] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning", in *21st Int. Conf. Mach. Learning*, 2004, p. 1.

[43] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning", in *22nd Int. Conf. Mach. Learning*, New York, NY, 2005, pp. 1–8.

[44] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations", in *25th Int. Conf. Mach. Learning*, 2008, pp. 144–151.

[45] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences", in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 4302–4310.

[46] G. Maeda, M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives", in *14th IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 527–534.

[47] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction", in *IEEE Int. Conf. Robot. and Automat.*, 2018, pp. 1–9.

[48] V. Kuleshov and O. Schrijvers, "Inverse game theory", *Web and Internet Econ.*, 2015.

[49] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning", in *Adv. Neural Inform. Process. Syst.*, 2016, pp. 3909–3917.

[50] D. Chen and C. Moler, *Symbolic Math Toolbox: User'Guide.* MathWorks, 1994.

[51] K. B. Petersen and M. S. Pedersen, *The matrix cookbook (version: Nov. 15, 2012)*, 2012.

[52] G. H. Golub and V. Pereyra, “The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate”, *SIAM J. Numerical Anal.*, vol. 10, no. 2, pp. 413–432, 1973.

[53] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, “Learning objective functions for manipulation”, in *IEEE Int. Conf. Robot. and Automat.*, 2013, pp. 1331–1336.

[54] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse optimal control via policy optimization”, in *33rd Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[55] R. E. Kalman, “When is a linear control system optimal?”, *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[56] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl, “A convex approach to inverse optimal control and its application to modeling human locomotion”, in *IEEE Int. Conf. Robot. and Automat.*, 2012, pp. 531–536.

[57] M. C. Priess, J. Choi, and C. Radcliffe, “Determining human control intent using inverse LQR solutions”, in *ASME Dyn. Syst. Control Conf.*, 2013.

[58] P. M. Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn, “Data-driven inverse optimization with imperfect information”, *Math. Program.*, vol. 167, no. 1, pp. 191–234, 2018.

[59] P. Englert, N. A. Vien, and M. Toussaint, “Inverse KKT: Learning cost functions of manipulation tasks from demonstrations”, *Int. J. Robot. Res.*, vol. 36, no. 13–14, pp. 1474–1488, 2017.

[60] C. M. Bishop, *Pattern Recognition and Machine Learning.* Secaucus, NJ: Springer, 2006.

[61] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

Part A.

Learning Optimal Controllers using Optimality Conditions

Paper P1

Constrained Inverse Optimal Control With Application to a Human Manipulation Task

Marcel Menner, Peter Worsnop, and Melanie N. Zeilinger

Published in *IEEE Transactions on Control Systems Technology*

Abstract: This paper presents an inverse optimal control methodology and its application to training a predictive model of human motor control from a manipulation task. It introduces a convex formulation for learning both objective function and constraints of an infinite-horizon constrained optimal control problem with nonlinear system dynamics. The inverse approach utilizes Bellman's principle of optimality to formulate the infinite-horizon optimal control problem as a shortest path problem and Lagrange multipliers to identify constraints. We highlight the key benefit of using the shortest path formulation, i.e., the possibility of training the predictive model with short and selected trajectory segments. The method is applied to training a predictive model of movements of a human subject from a manipulation task. The study indicates that individual human movements can be predicted with low error using an infinite-horizon optimal control problem with constraints on shoulder movement.

1 Introduction

As robotic systems are applied to increasingly unstructured and unpredictable environments, the ability to identify and adapt to their environment is becoming of critical importance. The collaboration with humans represents a particular challenge, as the interaction varies between individuals. The manipulation of an articulated object by a human in collaboration with a robot is one example, where the robot performance can be improved by learning a model to describe and predict the human motor control behavior [1].

The literature on human control behavior widely agrees on the fact that human motor performance is achieved through the reactive and predictive component (see the review in [2]). The reactive component is triggered by sensory inputs and updates an ongoing motor command; it can, therefore, be interpreted as the feedback control action. The predictive component capitalizes on the ability to anticipate motor events based on memory in order to accomplish a given task under foreseeable conditions, which can be interpreted as feedforward action [3]. The existence of these two components has been highlighted in studies of various motor control tasks, including grasping and manipulation [4]–[6].

In this work, we present a shortest path inverse optimal control method, which is applied to train a predictive model of human motor control. The inverse optimal control method is thereby used to learn the parameters of an optimal control problem from demonstrated state and input trajectories. In particular, it learns both the objective function and constraints of an underlying infinite-horizon optimal control problem from observed trajectory segments of finite length using optimality conditions of a corresponding shortest path problem and a candidate constraint set. The optimality conditions are derived based on Bellman's principle of optimality [7] and the Karush-Kuhn-Tucker (KKT) optimality conditions [8]. The proposed method is convex for objective functions that are linear in their parameters and for general nonlinear systems, where relevant constraints are identified from the candidate constraint set using Lagrange multipliers. The method is utilized to train a predictive model of movements of three human subjects from a human manipulation task.

We set up a human manipulation experiment, where three human subjects manipulated one end of a passive kinematic object whose position was changed consecutively by a robot. In this context, the goal of the inverse learning method is to train a predictive model of human movements. The underlying hypothesis is that the demonstrations of the human manipulation task are optimal with respect to an infinite-horizon constrained optimal control problem. The experimental study highlights the potential of the proposed learning approach by providing good predictive performance for individual human movements. In particular, the proposed shortest path formulation is shown to be beneficial for suboptimal execution, i.e., disregard the reactive human motor control component in the application considered in this paper.

Related inverse optimal control approaches are presented in [9]–[15]. The approaches in [9]–[11] can be interpreted as an inverse method of an infinite-horizon optimal control problem, but they are restricted to unconstrained, linear systems and quadratic objective functions. In [12], a bilevel approach to solve an inverse unconstrained optimal control problem is presented. The techniques closest to our method are [13]–[15], where the KKT conditions are similarly used for learning the stage cost but the constraints are assumed to be known. The two main distinctions of our approach with respect to [13]–[15] are the consideration of an optimal control problem with an infinite horizon and the simultaneous identification of constraints from a candidate constraint set that is constructed from data with a convex optimization problem. By using a shortest path formulation, the required trajectory segment for learning the parameters of the underlying optimal control problem can be shorter, e.g., compared to [14], and the learned parameters are invariant with respect to the chosen trajectory segment. As for the application, the incorporation of constraints results in better predictions of human movement, whereas the consideration of a shortest path formulation allows for isolating trajectory segments where the predictive component is dominant, i.e., where the hypothesis of optimal demonstrations with respect to an optimal controller is valid.

2 Shortest Path Inverse Optimal Control

This section presents an inverse optimal control (IOC) approach based on a shortest path formulation to learn an objective function and constraints from observations. The observations are represented as trajectories of state measurements $x(k) \in \mathbb{R}^n$ and inputs $u(k) \in \mathbb{R}^m$ at time-step k, where

$$x(k+1) = f(x(k), u(k)) \tag{P1-1}$$

with the potentially nonlinear function $f(\cdot)$ modeling the evolution of the state. For the derivation of the inverse method in this section, we assume that $f(\cdot)$ is given. Section 4 discusses how to identify $f(\cdot)$ for the considered application.

Observed trajectories are assumed to be optimal with respect to an infinite-horizon constrained optimal control problem, i.e., $x(k+i) = x_i^\star$

and $u(k+i) = u_i^\star \ \forall \ i \geq 0$ with

$$
\begin{aligned}
\{x_i^\star, u_i^\star\}_{i=0}^{\infty} = \ & \arg\min_{x_i, u_i} \sum_{i=0}^{\infty} l(x_i, u_i; L) && && \text{(P1-2a)} \\
& \text{s.t. } x_{i+1} = f(x_i, u_i) && \forall \ i \geq 0 && \text{(P1-2b)} \\
& \qquad C(x_i, u_i) \leq 0 && \forall \ i \geq 0 && \text{(P1-2c)} \\
& \qquad x_0 = x(k) && && \text{(P1-2d)}
\end{aligned}
$$

with stage cost $l(x_i, u_i; L)$ defined as a parametric function with parameters L, constraint set $C(x_i, u_i) \leq 0$, and initial state $x(k)$. The notation $\{\cdot\}_{i=0}^{\infty}$ is used to indicate indices from $i = 0$ to ∞. The goal in this work is to train a predictive model by learning both $l(x_i, u_i; L)$ and $C(x_i, u_i)$ from state and input measurements, which is referred to as the inverse problem to (P1-2) in the following.

Problem Definition

The first difficulty in the inverse problem of (P1-2) is that measurements $x(k), u(k)$ are not available for $k \to \infty$ but only in some finite segment. We address this using a shortest path formulation (see Section 2.1). For cases, where the constraint set $C(\cdot, \cdot)$ is unknown, we propose the construction of a candidate constraint set. The main step of the proposed approach is the derivation of optimality conditions of the shortest path formulation using the candidate constraint set (see Section 2.2). The optimality conditions are then used to simultaneously identify constraints from the candidate set and learn the stage cost parameters.

2.1 Formulation of infinite-horizon as shortest path problem

We formulate the infinite-horizon problem as a shortest path problem of finite length e and show that the minimizers of both the infinite-horizon problem and the shortest path problem are identical along the path, i.e., from time k to $k+e$. Let $X^m := [\ x(k)^\intercal \ x(k+1)^\intercal \ \ldots \ x(k+e)^\intercal \ ]^\intercal \in \mathbb{R}^{n(e+1)}$ and $U^m := [\ u(k)^\intercal \ u(k+1)^\intercal \ \ldots \ u(k+e-1)^\intercal \ ]^\intercal \in \mathbb{R}^{me}$ be the collection of state and input measurements, respectively, over the time interval k through $k+e$. If X^m, U^m describe the shortest path, then they (at least locally) minimize

$$
\begin{aligned}
\{X^m, U^m\} = \arg\min_{x_i,u_i} \; & \sum_{i=0}^{e-1} l(x_i, u_i; L) \\
\text{s.t.} \; & x_{i+1} = f(x_i, u_i) \\
& C(x_i, u_i) \leq 0 \quad i = 0, ..., e-1 \\
& x_0 = x(k) \\
& x_e = x(k+e).
\end{aligned}
\tag{P1-3}
$$

Using Bellman's principle of optimality [7], we can show that X^m, U^m then also correspond to minimizers of (P1-2) for $i = k, \dots\; k+e$, which is formally stated in the following theorem.

Theorem 1. *Consider a trajectory segment of measurements X^m, U^m from a dynamical system* (P1-1). *If the observed inputs U^m are the result of the optimal control problem in* (P1-2) *for times $k, ..., k+e-1$, then X^m, U^m also (at least locally) minimize the optimization problem in* (P1-3).

Proof. The optimization problem in (P1-2) can be written as

$$
\begin{aligned}
J^\star(x(k)) = \min_{x_i,u_i} \; & \sum_{i=0}^{e-1} l(x_i, u_i; L) + \sum_{i=e}^{\infty} l(x_i, u_i; L) \\
\text{s.t.} \; & \text{(P1-2b)}, \text{(P1-2c)}, \text{(P1-2d)}.
\end{aligned}
\tag{P1-4}
$$

If $x_e^\star$ is known, then, using Bellman's principle of optimality [7] with $x_e = x_e^\star$, (P1-4) can be formulated as

$$
\begin{aligned}
J^\star(x(k)) = \min_{x_i,u_i} \; & \sum_{i=0}^{e-1} l(x_i, u_i; L) + J^\star(x_e^\star) \\
\text{s.t.} \; & x_{i+1} = f(x_i, u_i) \quad i = 0, ..., e-1 \\
& C(x_i, u_i) \leq 0 \qquad i = 0, ..., e-1 \\
& x_0 = x(k) \\
& x_e = x_e^\star.
\end{aligned}
\tag{P1-5}
$$

Hence, the minimizers of (P1-2) and (P1-5) are equal for all $i = 0, ..., e$. The result follows with $x_e^\star = x(k+e)$. $\square$

Note that problem (P1-3) differs from a standard finite-horizon formulation as used in [14] by the end-point constraint $x_e = x(k+e)$, which makes a key difference for learning the problem parameters, as will be illustrated in Section 3.

Remark 1. *The shortest path formulation originates from the hypothesis that demonstrations are optimal with respect to the infinite-horizon problem in* (P1-2)*. For a different model/ hypothesis, the formulation of the inverse problem can differ. A particular advantage of the shortest path formulation is that any path along the measured trajectory can be used for learning. This allows for selecting particular paths where the assumption of optimal execution/data is fulfilled "more closely," e.g., high signal-to-noise ratio or negligible reactive human motor control component in the application considered.*

2.2 Optimality conditions

In the following, we derive optimality conditions of the shortest path problem in (P1-3) and show how they can be used for learning both parameters of the stage cost and constraints. First, we express the optimization problem in (P1-3) in terms of the inputs u_i by recursively defining $x_i = F_i(U, x_0)$:

$$F_i(U, x_0) := \begin{cases} x_0 & \text{if} \quad i = 0 \\ f(F_{i-1}(U, x_0), u_{i-1}) & \text{else} \end{cases} \tag{P1-6}$$

with $U := \begin{bmatrix} u_0^\mathsf{T} & u_1^\mathsf{T} & \dots & u_{e-1}^\mathsf{T} \end{bmatrix}^\mathsf{T}$. Hence, the resulting optimization problem is given as

$$\begin{aligned} \min_U \ & \sum_{i=0}^{e-1} l(F_i(U, x(k)), u_i; L) \\ \text{s.t.} \ \ & C(F_i(U, x(k)), u_i) \leq 0 \quad i = 0, ..., e-1 \\ & F_e(U, x(k)) = x(k+e), \end{aligned} \tag{P1-7}$$

where we use $x_0 = x(k)$. The Lagrangian $\mathcal{L}(U, \lambda, \nu, L)$ of the optimization problem in (P1-7) is given by

$$\begin{aligned} \mathcal{L}(U, \lambda, \nu, L) &= \nu^\mathsf{T} (F_e(U, x(k)) - x(k+e)) \\ &+ \sum_{i=0}^{e-1} l(F_i(U, x(k)), u_i; L) + \lambda_i^\mathsf{T} C(F_i(U, x(k)), u_i) \end{aligned} \tag{P1-8}$$

with the Lagrange multipliers $\lambda_i \geq 0$ and $\nu \in \mathbb{R}^n$ (see [16]), and L denoting the parameters of the stage cost $l(x_i, u_i; L)$. Using $\mathcal{L}(\cdot)$ in (P1-8), the KKT optimality conditions for the trajectory segment are given by

$$\nabla_U \mathcal{L}(U,\lambda,\nu,L) = 0 \tag{P1-9a}$$

$$\lambda_i^\mathsf{T} C(F_i(U,x(k)),u_i) = 0 \qquad i = 0,...,e-1 \tag{P1-9b}$$

$$\lambda_i \geq 0 \qquad i = 0,...,e-1 \tag{P1-9c}$$

$$C(F_i(U,x(k)),u_i) \leq 0 \qquad i = 0,...,e-1 \tag{P1-9d}$$

$$F_e(U,x(k)) - x(k+e) = 0. \tag{P1-9e}$$

Construction of candidate constraint set

Eq. (P1-9d) will hold for any observed trajectory with optimal execution (primal feasibility); however, the function C might be unknown. If C is unknown, we propose to use (P1-9d) to construct candidate constraints $\bar{C}(x_i,u_i)$ as the convex hull of all observed data points of the form $P[x_i^\mathsf{T}\ u_i^\mathsf{T}]^\mathsf{T} \leq p$. A subset of the candidate constraints is then identified as constraints via the KKT conditions. A method for computing the convex hull, i.e., P and p, is, e.g., presented in [17].

Optimality conditions for learning

The idea of the proposed approach is to solve (P1-9) for the parameters L of the stage cost $l(x_i,u_i;L)$ as well as for λ_i and ν, given measurements X^m, U^m and the candidate constraints $\bar{C}(x_i,u_i)$, i.e.

$$\nabla_U\ \bar{\mathcal{L}}(U,\lambda,\nu,L)\big|_{U=U^m} = 0 \tag{P1-10a}$$

$$\lambda_i^\mathsf{T} \bar{C}(x(k+i),u(k+i)) = 0 \qquad i = 0,...,e-1 \tag{P1-10b}$$

$$\lambda_i \geq 0 \qquad i = 0,...,e-1 \tag{P1-10c}$$

with the approximate Lagrangian $\bar{\mathcal{L}}(\cdot)$ defined as in (P1-8) where $\bar{C}(F_i(U, x(k)),u_i)$ replaces $C(F_i(U,x(k)),u_i)$. Eq. (P1-9d) is only needed for the construction of candidate constraints and (P1-9e) holds by construction. Hence, both $\bar{C}(x(i),u(i)) \leq 0$ and (P1-9e) are not needed for learning the stage cost parameters [see (P1-9) with (P1-10)]. The feasibility problem in (P1-10) is convex if $l(x_i,u_i;L)$ is linear in L. One can show that (P1-10) is always feasible using the convex hull as the candidate constraint set, provided optimal and noise-free data.

The Lagrange multipliers λ_i and their values are essential in the proposed IOC approach in order to identify constraints from the candidate set. Each scalar $\lambda_{i,j}$ can be interpreted as a force keeping the optimization problem (P1-7) from violating the corresponding primal constraint $\bar{C}_j(x_i,u_i) \leq 0$ at time i. In other words, the value of a dual variable $\lambda_{i,j}$ indicates the sensitivity of the optimization problem to the corresponding constraint

[16]. We define a measure for the identification of constraint j as $\Lambda_j \geq \bar{\Lambda}$ with

$$\Lambda_j = \textstyle\sum_{i=0}^{e-1} \lambda_{i,j}, \tag{P1-11}$$

where $\bar{\Lambda} \geq 0$ is a problem-specific threshold value. If, e.g., $\Lambda_j = 0$, the j^{th} constraint does not affect the minimizer of the optimization problem and does not represent a constraint. If, however, the value of Λ_j is very high, the minimizer is strongly affected by the constraint j and the constraint is therefore crucial in explaining the observed trajectory. Hence, Λ_j relates directly to the importance of constraint j. The larger Λ_j, the more important is constraint j. We utilize this relation to identify constraints from the candidate set. The identified constraints are used in the predictive model, along with the learned parameters of the stage cost.

2.3 Sub-optimal and noisy data

Eq. (P1-10) will be feasible if, and only if, the trajectory is the solution of an optimal control problem of the form (P1-2). In practice, however, even if this modeling assumption is correct, the feasibility problem in (P1-10) will not be satisfied exactly due to measurement or process noise. In order to learn from sub-optimal or noisy data, we propose to solve the relaxed problem

$$\begin{aligned} \min_{L,\nu,\lambda_i} \quad & \left\| \nabla_U \bar{\mathcal{L}}(U,\lambda,\nu,L)\big|_{U=U^m} \right\|_2^2 \\ \text{s.t.} \quad & \lambda_i^{\mathsf{T}} \bar{C}(x(k+i),u(k+i)) = 0 \\ & \lambda_i \geq 0 \qquad\qquad i = 0,...,e-1. \end{aligned} \tag{P1-12}$$

It is easy to verify that $\left\| \nabla_U \bar{\mathcal{L}}(\cdot)\big|_{U=U^m} \right\|_2^2 = 0$ indicates optimality with respect to (P1-10) and that (P1-12) is always feasible.

Remark 2. *The use of a shortest path formulation in this work is reflected through the term $\nu^{\mathsf{T}}(F_e(U,x(k)) - x(k+e))$ in* (P1-8)*. Thus, an inverse approach with finite horizon as in [14] is obtained with $\nu = 0$.*

Remark 3 (On active and identified constraints)**.** *A constraint j is active if $\bar{C}_j(x_i,u_i) = 0$ at time i. Using the proposed method for constructing candidate constraints, there are always active candidate constraints. However, it is important to note that not all active candidates yield $\Lambda_j > 0$; it is also possible that candidate j is active, i.e., $\bar{C}_j(x_i,u_i) = 0$, and $\Lambda_j = 0$. Inversely, $\Lambda_j = 0$ does not mean that the candidate j is never active but that the observed trajectory would have been the same with and without candidate j. Hence, candidate constraint j is not identified as constraint if $\Lambda_j = 0$. Section 3 illustrates this concept in a simulation example.*

3 Illustrative Example

In this section, we illustrate the IOC procedure and highlight its key benefits in simulation for a pendulum with the discrete-time state-space representation:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) + T_s x_2(k) \\ x_2(k) - T_s \frac{g}{l} \sin x_1(k) \end{bmatrix} + T_s \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u(k)$$

with $x_1(k) = \theta(t)$ at $t = kT_s$ and $T_s = 0.01\text{s}$, $g = 9.81\text{m/s}^2$, $l = 1\text{m}$, and $m = 1\text{kg}$. $\theta(t)$ is the angle and $u(t)$ is the applied torque in Nm, where $|u(t)| \leq \bar{u}$ with $\bar{u} = 5\text{Nm}$ is assumed to be the available torque. In the following, we consider an optimal controller of the form (P1-2) with constraints $u_i \leq 5$ and $-u_i \leq 5$ and stage cost $l(x_i, u_i; Q^{\text{gt}}, r^{\text{gt}}) = x_i^\mathsf{T} Q^{\text{gt}} x_i + r^{\text{gt}} |u_i| + u_i^2$. The goal in this example is to learn the constraints and the parameters Q^{gt} and r^{gt}.

3.1 Learning with shortest path and finite horizon methods

First, we highlight the main differences between the proposed shortest path formulation and two finite-horizon methods, i.e., a method using the KKT conditions similarly as in [14] and a probabilistic IOC method which uses a likelihood maximization similarly as in [18]. The finite-horizon KKT method differs from the presented approach by virtue of the term $\nu^\mathsf{T}(F_e(U, x(k)) - x(k+e))$ in (P1-8) and thus, follows readily with $\nu = 0$ (removing the term). The proposed IOC approach, similarly as the approach in [14], yields a convex semi-definite program, which can, e.g., be solved with MOSEK [19], whereas the likelihood maximization method yields a non-convex optimization problem, which in this example is solved with a projected gradient descent method.

Figure P1-1 shows results with trajectory segments from $t = 0\text{s}$ through t_e generated with $Q^{\text{gt}} = I$ and $r^{\text{gt}} = 0$, where we enforce $Q \succeq 0$. The middle plot shows that the proposed method only needs a segment from $t = 0\text{s}$ through $t_e \approx 0.5\text{s}$ to find the ground truth. Both methods with finite horizon are not able to learn the ground truth even if the segments are long and $\theta(t)$ is close to stationarity (see $Q_{12} \approx 1$ at $t_e = 1000\text{s}$).

3.2 Learning with and without candidate constraints

Next, consider the trajectories with $Q^{\text{gt}} = 10I$ and $r^{\text{gt}} = 1$ for comparing methods with and without candidate constraints using segments from t_i to $t_i + 2\text{s}$ [see Figure P1-2 (top)].

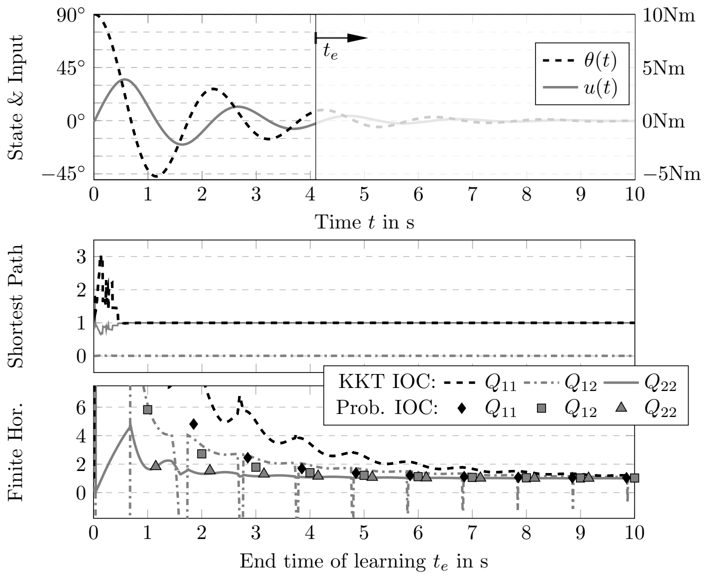


Figure P1-1. Top: State and input trajectories. Middle: Q learned with shortest path IOC. Bottom: Q learned with two finite-horizon methods: KKT and maximum likelihood.

IOC, constrained (2nd plot from the top)

The first step is to construct candidate constraints for the input $u(k)$:

$$u(k) \leq g_u \tag{P1-13a}$$

$$-u(k) \leq g_l \tag{P1-13b}$$

where g_u and g_l depend on the chosen segment and are displayed in red (diamond markers) and green (triangle markers), respectively. The algorithm returns Q and r as well as Λ_1 and Λ_2, which are defined in (P1-11) and correspond to the candidate constraints (P1-13a) and (P1-13b), respectively. The parameters Q and r are very close to the ground truth for all t_i. If $t_i < 0.96$s, $g_u = 5$ and $\Lambda_1 > 0$ suggesting that $u(k) \leq 5$ is indeed a constraint. If $t_i > 0.96$s, $g_u < 5$ and $\Lambda_1 = 0$ suggesting that $u(k) \leq g_u < 5$ is not a constraint, which is correct, as the constraint is not active. For all t_i, $g_l < 5$ and $\Lambda_2 = 0$ (not displayed) suggesting that $-u(k) \leq g_l < 5$ is not a constraint. Overall, Q is learned reliably and for $t_i < 0.96$, $u(k) \leq u_{max}$ is learned as constraint. The trajectory does not provide conclusive evidence about the existence of a lower bound, i.e., $-u(k) \leq u_{max}$, which is expected as $g_l < 5 \ \forall t_i$.

IOC, unconstrained (3rd plot from the top)

If $t_i > 0.96$s, Q and r are very close to the ground truth, which is expected since the control problem is virtually unconstrained in these segments. However, if no candidate constraints are constructed a priori, Q and r differ for $t_i < 0.96$s as the observed trajectory cannot be explained by means of an unconstrained optimal control problem.

Finite-horizon IOC, constrained (bottom plot)

The method learns the constraint $u(k) \leq 5$ using similar arguments as the proposed shortest path IOC method; however, it fails to capture the ground truth stage cost parameters with $r \approx 0$ and Q not close to Q^{gt} for all trajectory segments.

3.3 Summary of analysis

In this section, we have illustrated the benefits of the proposed approach. In particular, we showed the candidate constraint construction and how to simultaneously learn parameters of the stage cost and identify constraints from the candidate set. Further, we have shown that the proposed shortest path formulation only requires a short segment of measurements to learn the stage cost parameters and identify constraints, whereas finite-horizon approaches require a comparably long segment. Moreover, we have shown the importance of the candidate constraint set as a substantial component for correctly identifying the stage cost.

4 Manipulation of a Passive Kinematic Object

In this section, we show how to train a predictive model for human movements in a manipulation task using the proposed method. We conducted experiments with three human subjects where the underlying hypothesis is that humans plan their movements by solving a constrained optimal control problem.

4.1 Experiment description and system modeling

In the experiment, the human subjects manipulated one end of an object whose position was changed consecutively by a robot. The manipulation

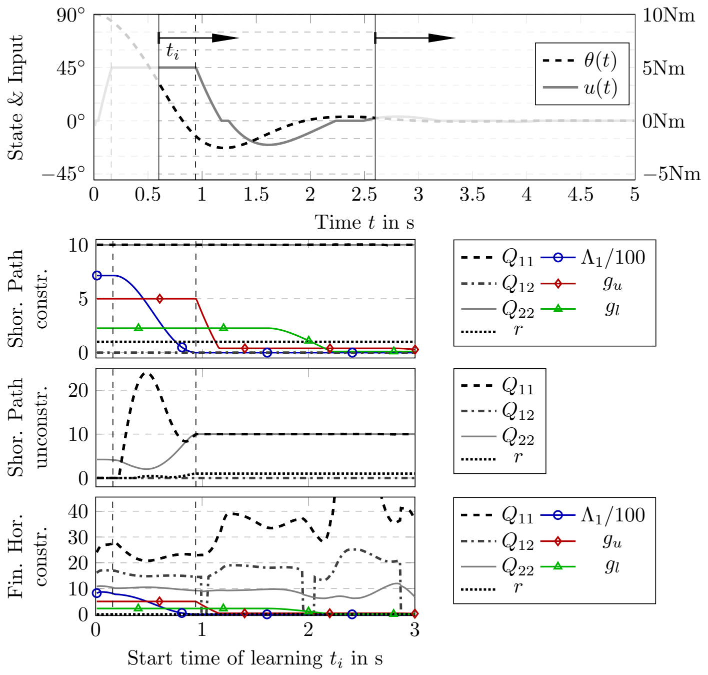


Figure P1-2. Top: State and input trajectories. 2nd from the top: Parameters learned with the proposed method with candidate constraints. 3rd from the top: Q learned without candidate constraints. Bottom: Parameters learned with finite-horizon method with candidate constraints.

task was set up to provide a foreseeable environment triggering the human's predictive motor control component such that the reactive control component can be disregarded (at least at the beginning of the movement). The object was articulated and unactuated and was composed of three lightweight wooden links and one cardboard handle, which acted as both a revolute joint and the manipulation point (see Figure P1-3). Hence, it had four revolute joints, one connecting its end link to the robot (joint 1), two connecting the three wooden links (joint 2 & 3), and the cardboard handle (joint 4), which was gripped by the subject such that the forearm and the handle acted as a single rigid body.

After familiarizing themself with the robot, the human was instructed to achieve specific angles for two of the object's joints, the joint connecting the object to the robot (joint 1 in Figure P1-3) and the first joint after that (joint 2), both of which have vertical rotational axes (perpendicular to the ground). The target angles were communicated to the subjects visually by reference-markers attached to the links. The subjects were asked to only move when the robot was stationary. First, the robot moved to disturb the system state. When the robot's motion ended, the subject corrected the reference error. Motion capture sensors were placed on all links of each kinematic chain and recorded through the Phasespace Python API.

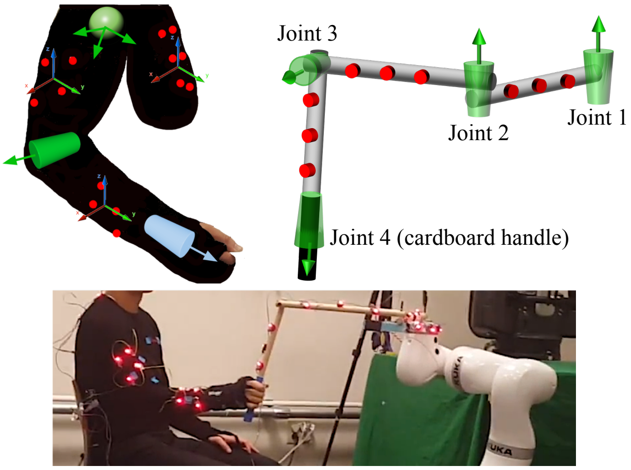


Figure P1-3. Top: Modeling of the human arm and the object. Bottom: Experiment setup with the Kuka LBR iiwa robot. Joints included in the model are shown in green, while the blue joint represents a freedom of motion that was constrained by experiment design. The motion capture markers are illustrated in red.

The derivation of the individual movement model, i.e., the system dynamics, of each subject is based on modeling the passive kinematic object and the human arm as a kinematic chain [20] whose parameters were identified from measurements. In this model, the base frame is attached to the torso and the manipulation frame is attached to the grip location of the hand. Ball joints such as the shoulder joint are modeled as three revolute joints in series with orthogonal axes intersecting at the center of the joint. This leads to the ball joint configuration being described with intrinsic Euler angles rotating around a point in space [21], [22]. The elbow joint is modeled as a single revolute joint. The wrist is modeled as three revolute joints in series; however a wrist brace was used in the experiment to restrict the motions in the frontal and sagittal plane, that is, waving and flapping motions. Pronation and supination (twisting about the forearm) could not be restricted by the brace; however the experiment was designed such that the kinematic chain of the object itself constrained this movement. Both the placement of the motion capture markers and the kinematic modeling are shown in Figure P1-3.

The system state $x(t) = [\ x_h(t)^\mathsf{T}\ x_o(t)^\mathsf{T}\ ]^\mathsf{T}$ is composed of the joint angles of the human, $x_h(t) \in \mathbb{R}^4$, and of the object, $x_o(t) \in \mathbb{R}^4$. The input to the system, $u(t) = \dot{x}_h(t)$, is given by the joint velocities of the human arm. The velocities of the object joint angles are given by:

$$\dot{x}_o(t) = J_o^\ddagger(x_o(t))V_g(t), \tag{P1-14}$$

where $J_o(x_o(t)) \in \mathbb{R}^{6\times4}$ is the Jacobian mapping joint velocities of the object to $V_g(t)$, the absolute twist velocity of the manipulation frame, and $J_o^\ddagger(x_o(t)) \in \mathbb{R}^{4\times6}$ denotes its Moore-Penrose pseudo-inverse [23]. Given that the human maintained a stationary base in the experiment, we can express $V_g(t)$ in terms of the human arm joint velocities and the Jacobian of the human arm, $J_h(x_h(t)) \in \mathbb{R}^{6\times4}$:

$$V_g(t) = J_h(x_h(t))\dot{x}_h(t). \tag{P1-15}$$

Using (P1-14) and (P1-15), $\dot{x}_o(t) = J_o^\ddagger(x_o(t))J_h(x_h(t))\dot{x}_h(t)$, and thus, the overall dynamics of the system is given by

$$\begin{bmatrix}\dot{x}_h(t)\\ \dot{x}_o(t)\end{bmatrix} = \begin{bmatrix} I \\ J_o^\ddagger(x_o(t))J_h(x_h(t))\end{bmatrix} u(t). \tag{P1-16}$$

In order to obtain the Jacobians, the twists representing the joints in each kinematic chain are identified by recording traces of the subject's range of motion and applying the techniques in [24]. The Jacobians $J_h(x_h(t))$ and $J_o(x_o(t))$ in (P1-16) are derived using the formula for the body Jacobian as in [25].

A discrete-time representation of (P1-16) is derived using an Euler-forward scheme with the sampling time T_s:

$$\begin{bmatrix}x_h(k+1)\\ x_o(k+1)\end{bmatrix} = \begin{bmatrix}x_h(k)\\ x_o(k)\end{bmatrix} + T_s \begin{bmatrix} I \\ J_o^\ddagger(x_o(k))J_h(x_h(k))\end{bmatrix} u(k).$$

An unscented Kalman filter as described in [26] is implemented to estimate the system state, where a static process model is chosen to smoothen the estimated angles, since measurement noise is amplified by the kinematic transformation. The inputs are computed as $u(k) = (x_h(k+1) - x_h(k))/T_s$.

4.2 Learning predictive model for human movements

Each of the three subjects maneuvered the object 15 times to correct the reference error induced by the robot. For each experiment, we recorded the entire trajectory from the start of the human movement until the subject was instructed to remain stationary. For reasons discussed in Section 4.3, we use the initial 1.2s, i.e., $e = 65$ in (P1-10) with sampling time $T_s = 0.0185$s for learning, which corresponds to roughly 60% of each trajectory. In order to generalize from the available sparse data, we utilize leave-one-out cross-validation [27], where we learn the parameters of the predictive model 15 times, each time removing one of the recorded trajectories. This is done to assess the robustness of the model.

Design choices

In this work, we train a predictive model with quadratic stage cost. Our goal is to exemplify the proposed method to build a simple predictive model of human movement. Quadratic stage costs are commonly used as objective function in optimal control offering a good compromise between complexity and expressivity, where the cost minimizes a trade-off between tracking a given target and control effort. Note that higher-order or more complex stage cost terms are possible with the proposed framework and there are various possibilities to express human movements [28]. Given that the task requires tracking a reference for only two of the states, we take a stage cost of the form

$$l(x_i, u_i) = (Sx_i - y_s)^\mathsf{T} Q (Sx_i - y_s) + u_i^\mathsf{T} R u_i,$$

where $y_s \in \mathbb{R}^2$ is the reference, $S = [\ 0_{2\times4}\ I_2\ 0_{2\times4}\ ]$ selects the states (two joint angles of the object) tracking y_s, and Q, R are the penalty parameters. We enforce $Q, R \succeq 0$ in order to obtain physically meaningful penalties for both deviation to the target angles and control effort. Also, we restrict the input penalties to $\sum_{i=1}^{m} R_{ii} = 1$, which fixes the scaling of the stage cost and avoids the trivial solution of all parameters being zero. We train one predictive model without constraints and one with a polytopic candidate constraint set for each subject.

Candidate constraints

The object's states $x_o(k)$ are modeled as unconstrained. The human's states $x_h(k)$ consist of the three shoulder joint angles and the elbow angle; the inputs $u(k)$ are the three angular velocities of the shoulder joint and the angular velocity of the elbow. Constraints on joint angles directly relate to constraints on $x_h(k)$, velocity constraints relate to constraints on $u(k)$, and acceleration constraints are computed as a rate constraint: $a(k) = (u(k+1) - u(k))/T_s$.

Learning results

Figure P1-4 shows the mean and standard deviation of Q and R obtained with the proposed IOC method. The most distinct feature is the scale of the parameters Q_{ij}, varying from order 10^{-2} for Subject 1, 10^{-3} for Subject 2, to 10^{-6} for Subject 3. The second most distinct feature is the difference in the diagonal elements of R that reflect movement of the shoulder, i.e., R_{11}, R_{22}, and R_{33}, whereas the penalty on elbow velocity is comparable, i.e., $R_{44} \approx 0.2$ for all subjects. Off-diagonal elements in R are similar across subjects.

Table P1-1 shows the sum of Lagrange multipliers as in (P1-11), which are used to identify constraints from the candidate constraint set. The Lagrange multipliers are stated as the mean over all experiments to identify constraints on the angle, velocity, and acceleration of shoulder and elbow joints. We consider constraint j as identified if the corresponding Lagrange multiplier $\Lambda_j \geq \bar{\Lambda} = 10^{-3}$. It can be seen that constraints are predominantly on shoulder movement. Constraints on elbow movement seem less important for all subjects. Note that even though the stage cost parameters in Figure P1-4 obtained with constrained and unconstrained IOC are relatively close for the individual subject, the resulting prediction models differ by virtue of the constraints identified as in Table P1-1.

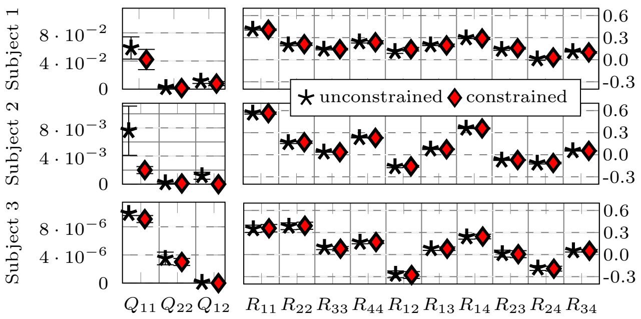


Figure P1-4. Mean and standard deviation of cost parameters Q and R for unconstrained learning (black stars) and constrained learning (red diamonds).

Table P1-1. Lagrange multipliers to identify constraints

	Angle		Velocity		Acceleration	
	Shoulder	Elbow	Shoulder	Elbow	Shoulder	Elbow
Subject 1	22.8	0	3.31*e*-2	0	1.38*e*-2	8.66*e*-4
Subject 2	11.5	0	2.78*e*-1	0	2.15*e*-2	6.98*e*-4
Subject 3	3.50	0	4.36*e*-1	2.86*e*-4	1.05*e*-1	0

4.3 Evaluation of trained human manipulation model

The difficulty in evaluating the quality of the trained model for human-centered experiments is the lack of a ground truth as reference. We therefore assess the quality of modeling human movement as an optimal control problem (P1-2) by comparing the true trajectory with the prediction provided by the model. The predictions are obtained by solving problem (P1-3) with the learned stage cost and identified constraints from the initial position at time $t = 0$s through $t = t_e = 1.2$s using IPOPT [29] (see Figure P1-5 for a sample prediction). We compute 15 sets of stage cost matrices by leaving out one trajectory for each learning. In order to evaluate the quality of the trained model, we use the left-out measured trajectory for validation against the predicted trajectory, which would result from (P1-3) with the learned stage cost and constraints. This technique ensures that the predicted trajectory is not biased by the corresponding measured trajectory. The mismatch between prediction $\hat{x}_i^j \in \mathbb{R}^8$ and measurement $x^j(i) \in \mathbb{R}^8$ of trajectory j is measured as the root mean square (RMS) error:

$$E^j = \sqrt{\tfrac{1}{8e} \textstyle\sum_{i=1}^{e} \|\hat{x}_i^j - x^j(i)\|_2^2}. \qquad \text{(P1-17)}$$

Intra-subject evaluation

First, we compute the errors E^j in (P1-17) for each trajectory j per subject. Figure P1-5 shows one measured trajectory of Subject 2 and the predictions obtained with the unconstrained and the constrained model. The prediction obtained with the unconstrained model shows a larger RMS error, best seen in the plot of human joint angles. The prediction obtained with the constrained model shows a lower error. Table P1-2 presents the mean and standard deviation over all 15 prediction errors for all subjects. It shows that, generally, the predictions have low errors ($< 3.3°$), where Subject 1 has the lowest ($< 1°$). On average, the presence of constraints improve the predictions by 20%-25%.

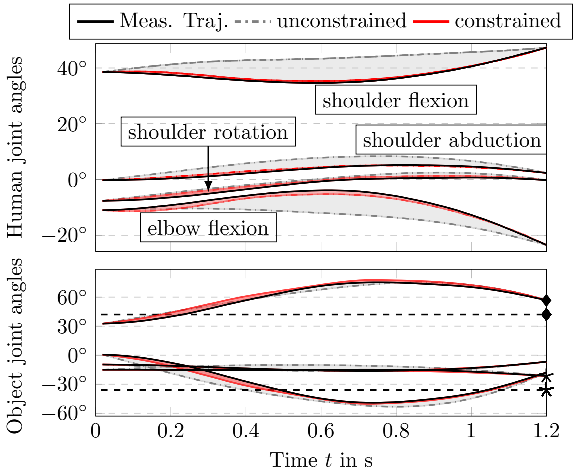


Figure P1-5. Measured trajectory in black, predicted trajectory with the unconstrained model in gray (error 4.17°) and the constrained model in red (error 1.40°). The upper plot shows the shoulder flexion, shoulder abduction, and shoulder rotation, as well as elbow flexion. The object states to be tracked are shown in the lower plot as dashed black lines and are related to the corresponding joints with a diamond and a star marker.

Table P1-2. Prediction errors: Unconstrained vs. constrained

Constraint set	unconstrained	constrained
Subject 1	$0.96° \pm 0.49°$	$0.78° \pm 0.42°$
Subject 2	$3.26° \pm 1.75°$	$2.45° \pm 0.87°$
Subject 3	$1.87° \pm 1.00°$	$1.56° \pm 0.79°$

Inter-subject cross-evaluation

Next, we analyze the individuality of the trained models, where the error E^j in (P1-17) is computed three times for each trajectory j: We compute the error using the prediction model of the subject who generated trajectory j; then, we compute E^j of the predicted trajectory $\hat{x}_i^j$ using the other subjects' prediction models, where we use the proposed IOC method with polytopic constraints.

Figure P1-6 shows an example of a measured trajectory from Subject 1, compared against predictions generated with the models of all subjects. The measured trajectory and the predicted trajectory of Subject 1 are close (error: 0.55°). The predicted trajectories of Subject 2 & 3 show higher errors. Table P1-3 states the mean and standard deviation of the errors between measurements of Subject j in columns j and prediction with the objective of Subject i in rows i over all trajectories. Hence, good separation between the subjects means large entries in the off-diagonal entries $i \neq j$. The results show high confidence in separating Subject 1 from the other two with high confusion errors (3.23°, 2.39° vs. 0.78°). The confidence to identify Subject 2 from a given trajectory is also high with confusion errors (3.99°, 3.59° vs. 2.45°). A less clear separation is observed for Subject 3, where the confusion errors are lower (2.22°, 1.91° vs. 1.56°). Overall, this cross-validation suggests that the models trained to predict the distinct motor behavior are individual.

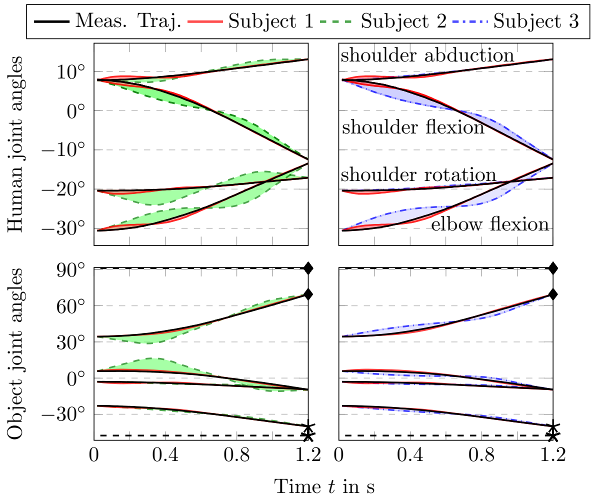


Figure P1-6. Measured trajectory of Subject 1 in black, predicted trajectory of Subject 1 in red (error: 0.55°). Left plots: Predicted trajectory of Subject 2 in green (error: 3.62°). Right plots: Predicted trajectory of Subject 3 in blue (error: 1.97°).

Table P1-3. Prediction errors: Cross-validation between subjects

	Trajectories of	Subject 1	Subject 2	Subject 3
Model	Subject 1	$0.78° \pm 0.42°$	$3.99° \pm 1.53°$	$2.22° \pm 1.14°$
	Subject 2	$3.23° \pm 1.03°$	$2.45° \pm 0.87°$	$1.91° \pm 0.93°$
	Subject 3	$2.39° \pm 0.68°$	$3.59° \pm 1.68°$	$1.56° \pm 0.79°$

Benefit of shortest path formulation

In the following, we discuss the advantages of using a shortest path formulation over a finite horizon in the context of the considered application. If the entire trajectory is used for training and stationarity is reached, i.e., e is large, both the proposed shortest path method and a finite-horizon method are similar. In the context of the considered application, however, we encountered two main challenges when considering the entire trajectory. Firstly, in the final part of the trajectory, the target angles are more or less reached and the measured signals are close to stationarity. As a result, the signal-to-noise ratio is low and can corrupt learning. Secondly, we observed small corrections around the target angles in the experiment suggesting the presence of reactive movements, which renders the final part of the trajectory not indicative of the predictive human motor control.

For shorter segments, the predictive component dominates both noise and reactive component but the solution from a finite-horizon formulation diverts from that with a shortest path (see Section 3). The proposed IOC approach allows for using only the initial part of the trajectory for learning where stationarity is not reached. Overall, the presence of both the reactive human motor control component and noise do not fulfill the assumption of optimal execution with respect to (P1-2). We used the initial 60% of the trajectory, which was observed to be a good trade-off between segment-length and avoidance of the reactive component.

Figure P1-7 revisits the trajectory in Figure P1-5 to illustrate the above discussion on the horizon length e. The upper plot shows the complete recorded trajectory, where some correction around the target angles can be observed for $t \geq 1.4$s (see joint angle marked by the diamond symbol). The lower plot displays the RMS error (P1-17) of the predictions that result from different horizon lengths e. The RMS error increases as a result of both the correction around the target angles and the low signal-to-noise ratio. It highlights that the modeling assumption as an open-loop optimal control problem is suitable for the predictive part, but not in the presence of the reactive component.

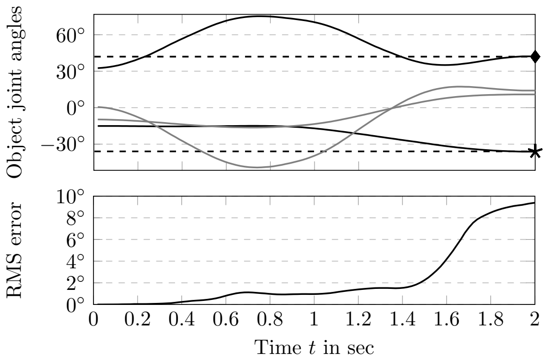


Figure P1-7. Top: Target angles to be tracked are shown as dashed black lines and are related to the corresponding joints with a diamond and a star marker. Bottom: RMS error of prediction with different horizon lengths e.

5 Conclusion

This paper presented an inverse optimal control approach to learn both cost function parameters and constraints from demonstrations, i.e., state and input measurements of dynamical systems. The shortest path formulation is shown to be the inverse problem to an infinite-horizon optimal control problem. By relying on the Karush-Kuhn-Tucker conditions, the problem is convex for cost functions that are linear in their parameters. We set up a human manipulation experiment to exemplify the proposed approach for modeling and predicting human arm movements. In the experiment, three human subjects manipulated one end of a passive kinematic object whose position was changed consecutively by a robot. The benefits of using a shortest path formulation and the consideration of constraints on human movements were highlighted. The results showed that a model with good predictive capabilities can be learned using a quadratic cost function for both states and inputs together with constraints on shoulder movements using the proposed formulation. Finally, it was shown that the predictive models of the human subjects are individual.

Bibliography

[1] J. Lee, P. H. Chang, and D. G. Gweon, "A cost function inspired by human arms movement for a bimanual robotic machining", in *IEEE Int. Conf. Robot. and Automat.*, 2012, pp. 5431–5436.

[2] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, "Principles of sensorimotor learning", *Nature Rev. Neurosci.*, vol. 12, no. 12, p. 739, 2011.

[3] M. Kawato, "Internal models for motor control and trajectory planning", *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[4] R. S. Johansson and G. Westling, "Coordinated isometric muscle commands adequately and erroneously programmed for the weight during lifting task with precision grip", *Experimental Brain Res.*, vol. 71, no. 1, pp. 59–71, 1988.

[5] R. S. Johansson and K. J. Cole, "Sensory-motor coordination during grasping and manipulative actions", *Current Opinion in Neurobiology*, vol. 2, no. 6, pp. 815–823, 1992.

[6] Q. Fu, W. Zhang, and M. Santello, "Anticipatory planning and control of grasp positions and forces for dexterous two-digit manipulation", *J. Neurosci.*, vol. 30, no. 27, pp. 9117–9126, 2010.

[7] R. E. Bellman, *Dynamic Programming.* Courier Dover Publications, 1957.

[8] H. W. Kuhn and A. W. Tucker, "Nonlinear programming", in *2nd Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1951.

[9] R. E. Kalman, "When is a linear control system optimal?", *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[10] M. C. Priess, R. Conway, J. Choi, J. M. Popovich, and C. Radcliffe, "Solutions to the inverse lqr problem with application to biological systems analysis", *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 2, pp. 770–777, 2014.

[11] M. Menner and M. N. Zeilinger, "A user comfort model and index policy for personalizing discrete controller decisions", in *Eur. Control Conf.*, 2018, pp. 1759–1765.

[12] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach", *Auton. Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[13] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl, “A convex approach to inverse optimal control and its application to modeling human locomotion”, in *IEEE Int. Conf. Robot. and Automat.*, 2012, pp. 531–536.

[14] P. Englert, N. A. Vien, and M. Toussaint, “Inverse KKT: Learning cost functions of manipulation tasks from demonstrations”, *Int. J. Robot. Res.*, vol. 36, no. 13–14, pp. 1474–1488, 2017.

[15] A. Majumdar, S. Singh, A. Mandlekar, and M. Pavone, “Risk-sensitive inverse reinforcement learning via coherent risk models”, in *Robot.: Sci. and Syst.*, 2017.

[16] S. Body and L. Vandenberghe, *Convex Optimization.* Cambridge, U.K.: Cambridge University Press, 2004.

[17] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls”, *ACM Trans. on Math. Software*, vol. 22, no. 4, pp. 469–483, 1996.

[18] S. Levine and V. Koltun, “Continuous inverse optimal control with locally optimal examples”, in *29th Int. Conf. Mach. Learning*, 2012, pp. 475–482.

[19] MOSEK ApS, “The mosek optimization software”, *Online: Http://www.mosek. com/*, 2010.

[20] G. Wu, S. Siegler, P. Allard, C. Kirtley, A. Leardini, D. Rosenbaum, M. Whittle, D. D. D’Lima, L. Cristofolini, H. Witte, O. Schmid, and I. Stokes, “ISB recommendation on definitions of joint coordinate system of various joints for the reporting of human joint motion – part i: Ankle, hip, and spine”, *J. Biomechanics*, vol. 35, no. 4, pp. 543–548, 2002.

[21] C. Bregler and J. Malik, “Tracking people with twists and exponential maps”, in *IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition*, 1998, pp. 8–15.

[22] G. P. Moll and B. Rosenhahn, “Ball joints for marker-less human motion capture”, in *Workshop Appl. Comput. Vision*, 2009, pp. 1–8.

[23] A. Ben-Israel and T. N. E. Greville, *Generalized inverses: Theory and applications.* Springer Science & Business Media, 2003, vol. 15.

[24] A. M. Bestick, S. A. Burden, G. Willits, N. Naikal, S. S. Sastry, and R. Bajcsy, “Personalized kinematics for human-robot collaborative manipulation”, in *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2015, pp. 1037–1044.

[25] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation.* CRC press, 1994.

[26] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation", in *IEEE Adaptive Syst. for Signal Process., Commun., and Control Symp.*, 2000, pp. 153–158.

[27] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. New York: Springer, 2013, vol. 112.

[28] O. Oguz, Z. Zhou, S. Glasauer, and D. Wollherr, "An inverse optimal control approach to explain human arm reaching control based on multiple internal models", *Scientific reports*, vol. 8, no. 1, p. 5583, 2018.

[29] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization", *Comput. & Chemical Eng.*, vol. 33, no. 3, pp. 575–582, 2009.

Paper P2

Maximum Likelihood Methods for Inverse Learning of Optimal Controllers

Marcel Menner and Melanie N. Zeilinger

Accepted for publication in *21st IFAC World Congress 2020*

Abstract: This paper presents a framework for inverse learning of objective functions for constrained optimal control problems, which is based on the Karush-Kuhn-Tucker (KKT) conditions. We discuss three variants corresponding to different model assumptions and computational complexities. The first method uses a convex relaxation of the KKT conditions and serves as the benchmark. The main contribution of this paper is the proposition of two learning methods that combine the KKT conditions with maximum likelihood estimation. The key benefit of this combination is the systematic treatment of constraints for learning from noisy data with a branch-and-bound algorithm using likelihood arguments. This paper discusses the theoretic properties of the learning methods and presents simulation results that highlight the advantages of using the maximum likelihood formulation for learning objective functions.

1 Introduction

Objective functions used for control design do not necessarily correspond to the actual performance specifications for a dynamical system, which may comprise complex or sparse targets. Instead, they are often chosen to facilitate gradient-based numerical optimization, which, in turn, makes their design not very intuitive and their calibration can require a tedious manual engineering effort to meet the performance specifications. Inverse learning concepts such as inverse optimal control offer an attractive design paradigm for learning objective functions from data to avoid their manual tuning. In this context, the data can originate, e.g., from a human actor, who demonstrates how to optimally operate the dynamical system being considered. Learning from demonstrations, however, necessarily implies that the data are subject to noise and other sources of sub-optimalities, which have to be taken into account.

In this paper, we present and contrast three variants of an inverse optimal control approach that leverages the Karush-Kuhn-Tucker optimality conditions, cf. [1], [2], to learn objective functions of optimal controllers, e.g., for linear quadratic or model predictive control, from noisy data. The first method is based on a convex relaxation of the KKT conditions to allow for noisy data, which is similar to the formulation in [3], [4] and included in this paper as the benchmark. The main contribution of this paper is the proposition of two inverse optimal control methods that combine the KKT conditions with a maximum likelihood estimation algorithm, which offer the key benefit of systematically dealing with state and input constraints in the presence of noisy data. The underlying assumption is that the data are samples from a distribution (rather than expecting deterministic, optimal data). Maximum likelihood estimation is enabled by an algorithm that uses branch-and-bound-type ideas based on the likelihoods of active constraints. The second contribution is a theoretical and simulative analysis of the properties of the three methods. In theory, we analyze the learning results of the inverse optimal control methods for unconstrained, linear dynamical systems and a quadratic cost function. In simulation, we present learning results for both constrained, linear and nonlinear systems.

Related work

Inverse optimal control methods typically model data as deterministic and resulting from an optimal control problem [5], whereas we explicitly consider the data as stochastic. In [6]–[8], inverse optimal control methods for linear, unconstrained systems are presented. Englert et al. [3] and Menner et al. [4] use a formulation similar to the first method presented in this paper, which is based on the relaxation of the KKT conditions. Chou et al. [9], [10] address a related problem by learning constraints. The method in [11] considers a non-deterministic model, but does not consider constraints in the learning procedure. The closest to the proposed likelihood estimation methods is [12], where the main difference lies in the proposed formulation using likelihood arguments offering the key advantage of dealing with constraints using a branch-and-bound algorithm.

Inverse reinforcement learning methods typically model data by means of a Markov decision process, cf. [13]–[15]. As a result, these methods can deal with noise by construction, but constraints are typically not considered. Compared to inverse reinforcement learning methods, we base our algorithm on the KKT conditions in order to explicitly consider constraints and noisy data.

2 Problem Statement

We consider discrete-time dynamical systems of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{P2-1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state at time k, $\boldsymbol{u}(k) \in \mathbb{R}^m$ is the input, and $\boldsymbol{f}$ is, in general, a nonlinear function.

Control Model

We consider optimal controllers of the form

$$\begin{aligned} \boldsymbol{v}_k^\star = \arg\min_{\boldsymbol{v}_k, \boldsymbol{z}_k\ \forall k} \quad & \boldsymbol{\theta}^T \boldsymbol{\phi}\left(\boldsymbol{v}_0, \dots \boldsymbol{v}_N, \boldsymbol{z}_0, \dots \boldsymbol{z}_{N+1}\right) \\ \text{s.t.} \quad & \boldsymbol{z}_{k+1} = \boldsymbol{f}(\boldsymbol{z}_k, \boldsymbol{v}_k) \quad \forall k = 0, \dots N \\ & \boldsymbol{g}(\boldsymbol{z}_k, \boldsymbol{v}_k) \leq \boldsymbol{0} \qquad \forall k = 0, \dots N \\ & \boldsymbol{z}_0 = \boldsymbol{z}(0), \end{aligned} \tag{P2-2}$$

where $\boldsymbol{v}_k^\star$ are optimal inputs at time k, $\boldsymbol{z}_k$ are the predicted states given inputs $\boldsymbol{v}_k$, and the initial condition is $\boldsymbol{z}(0)$. The minimizers of (P2-2), i.e., the nominal states $\boldsymbol{z}_k^\star$ and inputs $\boldsymbol{v}_k^\star$, express a motion plan and do not necessarily coincide with the measured states $\boldsymbol{x}(k)$ and inputs $\boldsymbol{u}(k)$. The objective function is defined by $\boldsymbol{\phi}$, which is weighted by the parameters $\boldsymbol{\theta}$. The function $\boldsymbol{g}$ defines constraints and N is the prediction horizon. We assume that $\boldsymbol{\phi}$, $\boldsymbol{f}$, and $\boldsymbol{g}$ are known and continuously differentiable.

Assumption on the data

In expectation, the data are assumed to be the solution to (P2-2), i.e., the demonstration, e.g. from a human agent, is modeled as an optimal controller. Due to noise and other sources of sub-optimalities, we assume a probability distribution for the data:

i) We model the initial condition, denoted $\boldsymbol{x}(0)$, as uncertain and assume

$$\boldsymbol{x}(0) \sim \mathcal{N}(\boldsymbol{z}(0), \boldsymbol{\Sigma}_0), \tag{P2-3a}$$

i.e., $\boldsymbol{x}(0)$ is Gaussian distributed with mean $\boldsymbol{z}(0)$ and covariance $\boldsymbol{\Sigma}_0$.

ii) We model the observed inputs, denoted $\boldsymbol{u}(k)$, as suboptimal and assume

$$\boldsymbol{u}(k) \sim \mathcal{N}(\boldsymbol{v}_k^\star, \boldsymbol{\Sigma}_k^u) \quad \forall k = 0, ..., N, \tag{P2-3b}$$

where $\boldsymbol{v}_k^\star$ are the minimizers of (P2-2).

In the context of learning from data generated by a human agent, eq. (P2-3a) and eq. (P2-3b) model that a human agent may be uncertain about the true initial state $\boldsymbol{x}(0)$ and may not execute the intended motion plan optimally.

Objective

In this paper, we learn the parameters $\boldsymbol{\theta}$ of the optimal controller in (P2-2) from data represented in the form of state $\boldsymbol{x}(k)$ and input $\boldsymbol{u}(k)$ measurements satisfying (P2-1) generated, e.g., by a human actor modeled as in (P2-3).

Notation & Preliminaries

In order to ease exposition, we vectorize sequences

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \vdots \\ \boldsymbol{u}(N) \end{bmatrix}, \ \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{N+1} \end{bmatrix}, \ \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_N \end{bmatrix}$$

and use the function $\boldsymbol{F}$ relating the vectorized sequences as in (P2-1): $\boldsymbol{Z} = \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{z}_0)$. We use $\boldsymbol{\phi}(\boldsymbol{v}_0, ...\boldsymbol{v}_N, \boldsymbol{z}_0, ...\boldsymbol{z}_{N+1}) = \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{Z})$, as well as $\boldsymbol{g}(\boldsymbol{v}_0, ...\boldsymbol{v}_N, \boldsymbol{z}_0, ...\boldsymbol{z}_{N+1}) = \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{Z})$ equivalently. Further, $\boldsymbol{U} \sim \mathcal{N}(\boldsymbol{V}, \boldsymbol{\Sigma})$ implies $\boldsymbol{u}(k) \sim \mathcal{N}(\boldsymbol{v}_k^\star, \boldsymbol{\Sigma}_k^u)$ for all $k = 0, ..., N$, i.e., $\boldsymbol{\Sigma} \in \mathbb{R}^{mN \times mN}$ is block-diagonal with blocks $\boldsymbol{\Sigma}_k^u$ and we define $\|\boldsymbol{x}\|_{\boldsymbol{X}} = \boldsymbol{x}^T \boldsymbol{X} \boldsymbol{x}$.

Let $\text{idx}, \neg\text{idx} \in \{0,1\}^s$ with $\text{idx} + \neg\text{idx} = \{1\}^s$. For a vector $\boldsymbol{\lambda} \in \mathbb{R}^s$, we define $\boldsymbol{\lambda}_{\text{idx}}$ selecting all elements λ_i for which $\text{idx}_i = 1$ ($\boldsymbol{\lambda}_{\neg\text{idx}}$ selecting λ_i for which $\text{idx}_i = 0$). $\boldsymbol{\delta}^i$ is a unit vector with $\delta_j^i = 1$ if $i = j$ and $\delta_j^i = 0$ if $i \neq j$.

Consider the optimization problem

$$\begin{aligned} \boldsymbol{V}^\star = \arg\min_{\boldsymbol{V}} \ & \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{x}) \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x}) \leq \boldsymbol{0}. \end{aligned} \tag{P2-4}$$

The KKT conditions of (P2-4) are given by

$$\mathrm{KKT}_{\boldsymbol{\theta}}(\boldsymbol{V}^\star, \boldsymbol{x}) = \begin{cases} \nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{x})|_{\boldsymbol{V}=\boldsymbol{V}^\star} = \boldsymbol{0} \\ \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x})) = 0 \\ \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x})) \leq \boldsymbol{0} \\ \boldsymbol{\lambda} \geq \boldsymbol{0} \end{cases} \tag{P2-5}$$

with the dual variables $\boldsymbol{\lambda}$ and the Lagrangian

$$\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x})) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x})).$$

The KKT conditions are necessary for constrained optimization (first-order derivative tests), i.e., any $\boldsymbol{V}^\star$ locally minimizing (P2-4) satisfies (P2-5). For more details, the reader is referred, e.g., to [16].

Proposition 1. Consider

$$\begin{aligned} f^1 = \max_{\boldsymbol{x} \in \mathbb{R}^n} \ & f(\boldsymbol{x}) \\ \text{s.t. } & g_1(\boldsymbol{x}) \leq 0 \end{aligned} \qquad \begin{aligned} f^2 = \max_{\boldsymbol{x} \in \mathbb{R}^n} \ & f(\boldsymbol{x}) \\ \text{s.t. } & g_1(\boldsymbol{x}) \leq 0, g_2(\boldsymbol{x}) \leq 0 \end{aligned}$$

and let $\{\boldsymbol{x} | g_1(\boldsymbol{x}) \leq 0\}$ be a non-empty set and $f(\boldsymbol{x})$ be bounded. Then, $f^1 \geq f^2$.

3 Inverse Learning Methods

This section presents three methods for inverse learning of the objective function, which utilize the KKT conditions in (P2-5) as follows: Suppose $\boldsymbol{V}^\star$ is the result of (P2-2) for some true parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$ and initial condition $\boldsymbol{z}(0)$. Then, $\mathrm{KKT}_{\boldsymbol{\theta}}(\boldsymbol{V}^\star, \boldsymbol{z}(0))$ hold for $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$. Hence, the KKT conditions can be used to learn $\boldsymbol{\theta}_\mathrm{t}$ given $\boldsymbol{V}^\star$. Method 1 is based on a relaxation of the KKT conditions in (P2-5) to allow for noisy data. Methods 2 and 3 are based on maximum likelihood estimation and use the distribution in (P2-3). The three methods vary in computational complexity and model assumptions in the form of approximations.

Method 1

This method uses a relaxation of the KKT conditions to directly relate the data $\boldsymbol{U}, \boldsymbol{x}(0)$ with $\boldsymbol{\theta}$:

$$\begin{aligned}\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta},\boldsymbol{\lambda}} \; & \|\nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{x}(0))|_{\boldsymbol{V}=\boldsymbol{U}}\|_I \\ \text{s.t. } & \boldsymbol{\lambda} \geq \mathbf{0} \\ & \boldsymbol{\lambda}_{\neg\text{idx}} = \mathbf{0},\end{aligned} \tag{P2-6a}$$

where $\neg$idx indexes inactive constraints, $g_i(\boldsymbol{U}, \boldsymbol{x}(0))$, with

$$\neg\text{idx}_i = \begin{cases} 1 & \text{if } g_i(\boldsymbol{U}, \boldsymbol{x}(0)) < 0 \\ 0 & \text{else.} \end{cases}$$

The main advantage is that (P2-6a) is a convex optimization problem. Compared to (P2-5), $\nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{x}(0))|_{\boldsymbol{V}=\boldsymbol{U}} \neq \mathbf{0}$ as well as $\boldsymbol{g}(\boldsymbol{U}, \boldsymbol{F}(\boldsymbol{U}, \boldsymbol{x})) \neq \mathbf{0}$ due to noisy data. Therefore, we minimize $\nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{x}(0))|_{\boldsymbol{V}=\boldsymbol{U}}$, where $\boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x})) = 0$ with $\boldsymbol{\lambda} \geq \mathbf{0}$ is relaxed to $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\lambda}_{\neg\text{idx}} = \mathbf{0}$.

Method 2

This method is based on maximum likelihood estimation and uses the expected value of the initial condition in (P2-3a) with $\boldsymbol{x}(0) = \boldsymbol{z}(0)$. Using the distribution of the control inputs in (P2-3b), the parameters $\boldsymbol{\theta}$ are estimated to maximize the probability of observing $\boldsymbol{U}$:

$$\begin{aligned}\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; & p(\boldsymbol{V}|\boldsymbol{U}, \boldsymbol{\Sigma}) \\ \text{s.t. } \boldsymbol{V} = \arg\min_{\tilde{\boldsymbol{V}}} \; & \boldsymbol{\theta}^T \boldsymbol{\phi}(\tilde{\boldsymbol{V}}, \boldsymbol{F}(\tilde{\boldsymbol{V}}, \boldsymbol{x}(0))) \\ \text{s.t. } & \boldsymbol{g}(\tilde{\boldsymbol{V}}, \boldsymbol{F}(\tilde{\boldsymbol{V}}, \boldsymbol{x}(0))) \leq \mathbf{0}.\end{aligned} \tag{P2-6b}$$

Method 3

This method considers the uncertainty about the initial condition explicitly. The method additionally optimizes over the initial condition with $\boldsymbol{x}(0) \sim \mathcal{N}(\boldsymbol{z}(0), \boldsymbol{\Sigma}_0)$ and the model for the inverse learning problem is given by

$$\begin{aligned}\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta},\boldsymbol{z}(0)} \; & p(\boldsymbol{V}|\boldsymbol{U}, \boldsymbol{\Sigma}) p(\boldsymbol{z}(0)|\boldsymbol{x}(0), \boldsymbol{\Sigma}_0) \\ \text{s.t. } \boldsymbol{V} = \arg\min_{\tilde{\boldsymbol{V}}} \; & \boldsymbol{\theta}^T \boldsymbol{\phi}(\tilde{\boldsymbol{V}}, \boldsymbol{F}(\tilde{\boldsymbol{V}}, \boldsymbol{z}(0))) \\ \text{s.t. } & \boldsymbol{g}(\tilde{\boldsymbol{V}}, \boldsymbol{F}(\tilde{\boldsymbol{V}}, \boldsymbol{z}(0))) \leq \mathbf{0}.\end{aligned} \tag{P2-6c}$$

Both maximum likelihood estimation methods (P2-6b) and (P2-6c) yield bi-level optimization problems that are solved as described in Section 4.

4 Algorithm for Maximum Likelihood Estimation

In the following, we outline the algorithm for learning $\boldsymbol{\theta}$ using Method 2. The algorithm for Method 3 follows analogously. We first replace the likelihood $p(\boldsymbol{V}|\boldsymbol{U},\boldsymbol{\Sigma})$ by its logarithmic likelihood $\log p(\boldsymbol{V}|\boldsymbol{U},\boldsymbol{\Sigma})$ (log-likelihood) and the lower level optimization problem in (P2-6b) by its KKT conditions in (P2-5). This way, the bi-level optimization problem is replaced by a combinatorial problem due to the complementary slackness condition $\boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{V},\boldsymbol{F}(\boldsymbol{V},\boldsymbol{x})) = 0$:

$$\begin{aligned} p = \max_{\boldsymbol{\theta},\boldsymbol{V},\boldsymbol{\lambda},\mathrm{idx}} \quad & \log p\left(\boldsymbol{V}|\boldsymbol{U},\boldsymbol{\Sigma}\right) \\ \text{s.t.} \quad & \mathrm{KKT}_{\boldsymbol{\theta},\mathrm{idx}}(\boldsymbol{V},\boldsymbol{x}(0)) \\ & \mathrm{idx} \in \{0,1\}^s \end{aligned} \tag{P2-7}$$

with idx selecting which of the s constraints are active, i.e.

$$\mathrm{KKT}_{\boldsymbol{\theta},\mathrm{idx}}(\boldsymbol{V},\boldsymbol{x}) = \begin{cases} \nabla_{\tilde{\boldsymbol{V}}} \mathcal{L}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{V}},\boldsymbol{x})|_{\tilde{\boldsymbol{V}}=\boldsymbol{V}} = \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{V},\boldsymbol{F}(\boldsymbol{V},\boldsymbol{x})) \leq \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{V},\boldsymbol{F}(\boldsymbol{V},\boldsymbol{x}))_{\mathrm{idx}} = 0 \\ \boldsymbol{\lambda} \geq \boldsymbol{0} \\ \boldsymbol{\lambda}_{\neg\mathrm{idx}} = \boldsymbol{0}. \end{cases}$$

Solving the combinatorial optimization problem in (P2-7) directly is computationally intensive. However, using likelihood arguments with branch-and-bound-type ideas, (P2-7) becomes practically feasible as outlined in the following.

Algorithm P2-1 summarizes the procedure to solve (P2-7), which is based on systematically enumerating candidate solutions and is conceptually similar to active-set methods, cf., [17]. The algorithm aims at reducing the number of times (P2-7) has to be solved for a fixed combination of active constraints, denoted $\mathrm{idx}^j \in \{0,1\}^s$, where we use j to index the specific combination of active constraints. First (Line 1 in Alg. P2-1), we solve

$$\begin{aligned} p^0 = \max_{\boldsymbol{\theta},\boldsymbol{V}} \quad & \log p\left(\boldsymbol{V}|\boldsymbol{U},\boldsymbol{\Sigma}\right) \\ \text{s.t.} \quad & \mathrm{KKT}_{\boldsymbol{\theta},\mathrm{idx}^0=\{0\}^s}(\boldsymbol{V},\boldsymbol{x}(0)), \end{aligned} \tag{P2-8}$$

where p^0 is the log-likelihood of observing $\boldsymbol{U}$ and no active constraints ($\mathrm{idx}^0 = \{0\}^s$). Next (Line 2), we compute an upper bound on the log-likelihood of constraint i's activeness given the data $\boldsymbol{U}$ as

$$\begin{aligned} \bar{p}^i = \max_{\boldsymbol{V}} \quad & \log p(\boldsymbol{V}|\boldsymbol{U},\boldsymbol{\Sigma}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{V},\boldsymbol{F}(\boldsymbol{V},\boldsymbol{x}(0))) \leq \boldsymbol{0} \\ & \boldsymbol{g}(\boldsymbol{V},\boldsymbol{F}(\boldsymbol{V},\boldsymbol{x}(0)))_{\boldsymbol{\delta}^i} = \boldsymbol{0}. \end{aligned} \tag{P2-9}$$

From Proposition 1, constraint i is not likely to be active if $\bar{p}^i \leq p^0$ and consequently, constraint i does not need to be enumerated, which is the first key component of the algorithm's efficiency as the number of possible constraint combinations, denoted c, can be reduced significantly.

Next (Line 3), we compute upper bounds for the log-likelihood p in (P2-7) for the fixed combinations of the active constraints idx^j (excluding the discarded constraints):

$$\begin{aligned} \tilde{p}^j = \max_{\boldsymbol{V}} \ & \log p(\boldsymbol{V}|\boldsymbol{U}, \boldsymbol{\Sigma}) \\ \text{s.t. } & \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x}(0))) \leq \boldsymbol{0} \\ & \boldsymbol{g}(\boldsymbol{V}, \boldsymbol{F}(\boldsymbol{V}, \boldsymbol{x}(0)))_{\text{idx}^j} = \boldsymbol{0} \end{aligned} \tag{P2-10}$$

for all $j = 1, ...c$. As a result, we obtain c possible candidate constraint combinations as well as their upper bounds:

$$\mathcal{D} = \left\{ \left\{ \text{idx}^1, \tilde{p}^1 \right\}, \left\{ \text{idx}^2, \tilde{p}^2 \right\}, ..., \left\{ \text{idx}^c, \tilde{p}^c \right\} \right\}. \tag{P2-11}$$

For ease of exposition, $\mathcal{D}$ is ordered so that $\tilde{p}^j \geq \tilde{p}^{j+1}$. The log-likelihoods $\tilde{p}^j$ are the second key component for the algorithm's efficiency and are used as the stopping criteria, i.e., (P2-7) is solved for idx^j starting with $j = 1$ until $\tilde{p}^j \leq \max\{p^0, ..., p^{j-1}\}$ (Line 5–9).

Algorithm P2-1 Overall algorithm

```
1: $\hat{\boldsymbol{\theta}}, p^0 \leftarrow$ Solve (P2-8) with $\text{idx} = \{0\}^s$
2: For each $i = 1, ..., s$, compute $\bar{p}^i$ using (P2-9) and discard constraint $i$
   if $\bar{p}^i \leq p^0$
3: Compute $\mathcal{D}$ in (P2-11) using (P2-10)
4: $j = 1, \hat{p} = p^0$
5: while $\tilde{p}^j > \max\{p^0, ..., p^{j-1}\} = \hat{p}$        ▷ end if less likely
6:     $\boldsymbol{\theta}^j, p^j \leftarrow$ Solve (P2-7) with fixed active constraints $\text{idx}^j$
7:     if $p^j \geq \hat{p}$
8:         $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^i$, $\hat{p} \leftarrow p^i$
9:     $j \leftarrow j + 1$
```

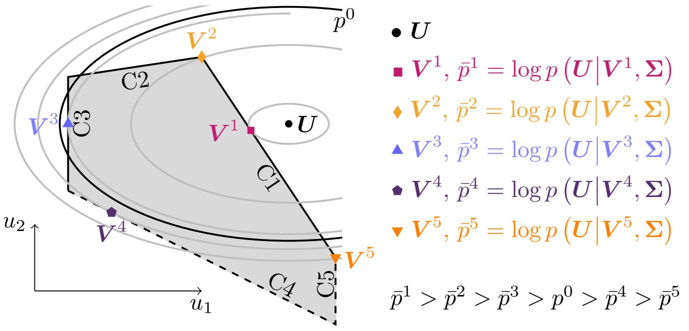
Remark 1. *We implemented a projected gradient method that uses backtracking line search [18] to solve both* (P2-6a) *for Method 1 and* (P2-7) *with the fixed active constraints* idx^j *for Method 2 and Method 3. Section 6 details the computation times of the three inverse learning methods, which show that the proposed algorithm is computationally feasible.*

Fig. P2-1 illustrates the concept of the upper bounds on the constraint likelihoods (Line 2 in Algorithm P2-1). In the given example, constraint

4 and constraint 5 do not have to be considered for learning, i.e., do not need to be enumerated. The resulting candidate constraint combinations are

$$\mathcal{D} = \left\{ \left\{ \begin{bmatrix} 1\\0\\0\\0\\0 \end{bmatrix}, \tilde{p}^1 \right\}, \left\{ \begin{bmatrix} 0\\1\\0\\0\\0 \end{bmatrix}, \tilde{p}^2 \right\}, \left\{ \begin{bmatrix} 1\\1\\0\\0\\0 \end{bmatrix}, \tilde{p}^3 \right\}, \left\{ \begin{bmatrix} 0\\0\\1\\0\\0 \end{bmatrix}, \tilde{p}^4 \right\} \right\}$$

Note that $\tilde{p}^1 = \bar{p}^1$, $\tilde{p}^2 = \bar{p}^2$, $\tilde{p}^3 = \bar{p}^2$, and $\tilde{p}^4 = \bar{p}^3$.



p^0
$\boldsymbol{V}^2$
C2
$\boldsymbol{V}^3$
C3
$\boldsymbol{V}^1$
$\bullet\,\boldsymbol{U}$
C1
u_2
$\boldsymbol{V}^4$
$\boldsymbol{V}^5$
C4
C5
u_1
● $\boldsymbol{U}$
■ $\boldsymbol{V}^1$, $\bar{p}^1 = \log p\left(\boldsymbol{U} \middle| \boldsymbol{V}^1, \boldsymbol{\Sigma}\right)$
♦ $\boldsymbol{V}^2$, $\bar{p}^2 = \log p\left(\boldsymbol{U} \middle| \boldsymbol{V}^2, \boldsymbol{\Sigma}\right)$
▲ $\boldsymbol{V}^3$, $\bar{p}^3 = \log p\left(\boldsymbol{U} \middle| \boldsymbol{V}^3, \boldsymbol{\Sigma}\right)$
⬟ $\boldsymbol{V}^4$, $\bar{p}^4 = \log p\left(\boldsymbol{U} \middle| \boldsymbol{V}^4, \boldsymbol{\Sigma}\right)$
▼ $\boldsymbol{V}^5$, $\bar{p}^5 = \log p\left(\boldsymbol{U} \middle| \boldsymbol{V}^5, \boldsymbol{\Sigma}\right)$
$\bar{p}^1 > \bar{p}^2 > \bar{p}^3 > p^0 > \bar{p}^4 > \bar{p}^5$

Figure P2-1. Illustration of upper bounds on likelihoods $\bar{p}^i$ as computed in (P2-9) for polytopic constraints $i = 1, \ldots 5$. The five upper bounds $\bar{p}^1$–$\bar{p}^5$ (gray ellipses) for the five constraints (C1–C5) as well as p^0 (black ellipse) are displayed as level sets. For Ci, $\boldsymbol{V}^i$ denotes the corresponding, projected input sequence.

5 Analysis for Linear Systems and Quadratic Cost Function

In this section, we present properties of the learning methods for a common class of dynamical systems and cost functions. Suppose the system in (P2-1) is unconstrained and linear (time-invariant or time-varying), i.e.,

$$\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k, \tag{P2-12a}$$

and the cost function is of the form

$$\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{Z}) = \sum_{k=0}^{N} \boldsymbol{z}_k^T \boldsymbol{Q}_{\boldsymbol{\theta}} \boldsymbol{z}_k + \boldsymbol{v}_k^T \boldsymbol{R}_{\boldsymbol{\theta}} \boldsymbol{v}_k \tag{P2-12b}$$

with $\boldsymbol{Q}_{\boldsymbol{\theta}}, \boldsymbol{R}_{\boldsymbol{\theta}} \succ \boldsymbol{0}$. Then, the stationarity condition can be written as

$$\nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{V}, \boldsymbol{z}(0)) = \boldsymbol{M}_{\boldsymbol{\theta}} \boldsymbol{V} + \boldsymbol{N}_{\boldsymbol{\theta}} \boldsymbol{z}(0),$$

where both $\boldsymbol{M}_{\boldsymbol{\theta}}$ and $\boldsymbol{N}_{\boldsymbol{\theta}}$ depend on the system dynamics, i.e., $\boldsymbol{A}_k, \boldsymbol{B}_k$, and are linear in their parameters $\boldsymbol{\theta}$, i.e., $\boldsymbol{M}_{\mu\boldsymbol{\theta}_1+\boldsymbol{\theta}_2} = \mu \boldsymbol{M}_{\boldsymbol{\theta}_1} + \boldsymbol{M}_{\boldsymbol{\theta}_2}$ and $\boldsymbol{N}_{\mu\boldsymbol{\theta}_1+\boldsymbol{\theta}_2} = \mu \boldsymbol{N}_{\boldsymbol{\theta}_1} + \boldsymbol{N}_{\boldsymbol{\theta}_2}$ with the scalar μ.

Let $\boldsymbol{\theta}_\mathrm{t}$ be the true parameters and, without loss of generality, $\|\boldsymbol{\theta}_\mathrm{t}\|_2 = 1$ (scale-invariance of the cost function), i.e., $\nabla_{\boldsymbol{V}} \mathcal{L}_{\boldsymbol{\theta}_\mathrm{t}}(\boldsymbol{V}, \boldsymbol{z}(0)) = \boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}} \boldsymbol{V} + \boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}} \boldsymbol{z}(0) = \boldsymbol{0}$. The goal of the learning methods is thus to estimate $\boldsymbol{\theta}_\mathrm{t}$ or any scaled version $\hat{\boldsymbol{\theta}} = \mu \boldsymbol{\theta}_\mathrm{t}$ with $\mu > 0$ from data. Desirable properties of the learning method are that $\boldsymbol{\theta}_\mathrm{t}$ results in expectation and that all $\hat{\boldsymbol{\theta}} = \mu \boldsymbol{\theta}_\mathrm{t}$ with $\mu > 0$ are equally likely.

Theorem 1 shows that the expected value of Method 3 is the true parameter vector $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_\mathrm{t}$ and that Method 3 is indifferent toward the cost function's scale, i.e., any $\hat{\boldsymbol{\theta}} = \mu \boldsymbol{\theta}_\mathrm{t}$ with $\mu > 0$ are equally likely (in expectation). Method 2 is equally indifferent toward the parameters' scale but the expected parameters are only $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_\mathrm{t}$ if $\boldsymbol{x}(0) = \boldsymbol{z}(0)$ (proof omitted as it can be similarly derived). Theorem 2 shows that the expected parameters $\hat{\boldsymbol{\theta}}$ of Method 1 are not necessarily $\boldsymbol{\theta}_\mathrm{t}$ and that Method 1 is not indifferent toward the parameters' scale.

Theorem 1. *Consider unconstrained, linear systems of the form* (P2-12a) *and cost functions* (P2-12b). *Let* $\boldsymbol{U} \sim \mathcal{N}(\boldsymbol{V}, \boldsymbol{\Sigma})$ *and* $\boldsymbol{x}(0) \sim \mathcal{N}(\boldsymbol{z}(0), \boldsymbol{\Sigma}_0)$. *In expectation, Method 3 returns* $\boldsymbol{\theta}_\mathrm{t}$ *(result 1) and any other parameter realization is necessarily* $\hat{\boldsymbol{\theta}} \propto \boldsymbol{\theta}_\mathrm{t}$ *(result 2).*

Proof. Without loss of generality, define $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t} + \mu \partial\boldsymbol{\theta}$ with $\mu \in \mathbb{R}$ and $\partial\boldsymbol{\theta} \in \mathbb{R}^p$ such that $\|\partial\boldsymbol{\theta}\|_2 = 1$. The results will be shown by proving the following statements:

Claim 1: For any $\partial\boldsymbol{\theta}$,

$$\begin{aligned}\hat{\mu} = 0 = \arg\max_{\mu, \boldsymbol{V}} \mathbb{E}\left[-\|\boldsymbol{U} - \boldsymbol{V}\|_{\boldsymbol{\Sigma}^{-1}} - \|\boldsymbol{x}(0) - \boldsymbol{z}(0)\|_{\boldsymbol{\Sigma}_0^{-1}}\right] \\ \text{s.t. } \boldsymbol{0} = \boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}+\mu\partial\boldsymbol{\theta}} \boldsymbol{V} + \boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}+\mu\partial\boldsymbol{\theta}} \boldsymbol{z}(0)\end{aligned}$$

Claim 2: For $\partial\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$, any $\mu \in \mathbb{R}$ minimizes

$$\begin{aligned}\hat{\mu} = \arg\max_{\mu, \boldsymbol{V}} \mathbb{E}\left[-\|\boldsymbol{U} - \boldsymbol{V}\|_{\boldsymbol{\Sigma}^{-1}} - \|\boldsymbol{x}(0) - \boldsymbol{z}(0)\|_{\boldsymbol{\Sigma}_0^{-1}}\right] \\ \text{s.t. } \boldsymbol{0} = \boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}+\mu\partial\boldsymbol{\theta}} \boldsymbol{V} + \boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}+\mu\partial\boldsymbol{\theta}} \boldsymbol{z}(0)\end{aligned}$$

Result 1 follows readily from Claim 1 as $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$. Result 2 follows from Claim 2 as $\boldsymbol{\theta} = (1 + \hat{\mu})\boldsymbol{\theta}_\mathrm{t} \propto \boldsymbol{\theta}_\mathrm{t}$.

Proofs of Claim 1 and Claim 2. Notice first that $\boldsymbol{M_\theta} \succ \boldsymbol{0}$ is invertible. The log-likelihood of Method 3 in (P2-6c) is proportional to

$$-\|\boldsymbol{U}-\boldsymbol{V}\|_{\boldsymbol{\Sigma}^{-1}} - \|\boldsymbol{x}(0)-\boldsymbol{z}(0)\|_{\boldsymbol{\Sigma}_0^{-1}}. \tag{P2-13}$$

Using $\boldsymbol{V} = -\boldsymbol{M}_{\boldsymbol{\theta}}^{-1}\boldsymbol{N_\theta z}(0)$, (P2-13) can be written as

$$-\|\boldsymbol{M_\theta U}+\boldsymbol{N_\theta z}(0)\|_{(\boldsymbol{M}_{\boldsymbol{\theta}}^T\boldsymbol{\Sigma M_\theta})^{-1}} - \|\boldsymbol{x}(0)-\boldsymbol{z}(0)\|_{\boldsymbol{\Sigma}_0^{-1}}. \tag{P2-14}$$

Then, using linearity ($\boldsymbol{M_\theta} = \boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}} + \mu\boldsymbol{M}_{\partial\boldsymbol{\theta}}$ and $\boldsymbol{N_\theta} = \boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}} + \mu\boldsymbol{N}_{\partial\boldsymbol{\theta}}$), $\boldsymbol{U} = \boldsymbol{V}+\partial\boldsymbol{V}$, and $\boldsymbol{x}(0) = \boldsymbol{z}(0)+\partial\boldsymbol{z}$, the expected value of (P2-14) yields

$$\begin{gathered}-\mu^2\|\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{V}+\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{z}(0)\|_{(\boldsymbol{M}_{\boldsymbol{\theta}}^T\boldsymbol{\Sigma M_\theta})^{-1}} \\ -\mathrm{trace}\left(\boldsymbol{\Sigma\Sigma}^{-1}\right) - \mathrm{trace}\left(\boldsymbol{\Sigma}_0\boldsymbol{\Sigma}_0^{-1}\right)\end{gathered} \tag{P2-15}$$

Therefore, for any $\partial\boldsymbol{\theta}$, $\mu = 0$ maximizes (P2-15), which proves Claim 1. For $\partial\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$, $\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{V}+\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{z}(0) = \boldsymbol{0}$ and $\mu \in \mathbb{R}$ minimizes (P2-15), which proves Claim 2. □

Theorem 2. *Consider unconstrained linear systems of the form* (P2-12a) *and cost functions* (P2-12b). *Let* $\boldsymbol{U} \sim \mathcal{N}(\boldsymbol{V},\boldsymbol{\Sigma})$ *and* $\boldsymbol{x}(0) \sim \mathcal{N}(\boldsymbol{z}(0),\boldsymbol{\Sigma}_0)$. *Then,* $\boldsymbol{\theta}_\mathrm{t}$ *does not result in expectation from Method 1 (result 1). Further, Method 1 is not scale-invariant (result 2).*

Proof. For the considered class of dynamical systems, the cost function in (P2-6a) is

$$\|\boldsymbol{M_\theta U}+\boldsymbol{N_\theta x}(0)\|_{\boldsymbol{I}}. \tag{P2-16}$$

We define $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t} + \mu\partial\boldsymbol{\theta}$ with $\mu \in \mathbb{R}$ and $\partial\boldsymbol{\theta} \in \mathbb{R}^p$ and $\|\partial\boldsymbol{\theta}\|_2 = 1$. Then, using linearity ($\boldsymbol{M_\theta} = \boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}} + \mu\boldsymbol{M}_{\partial\boldsymbol{\theta}}$ and $\boldsymbol{N_\theta} = \boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}} + \mu\boldsymbol{N}_{\partial\boldsymbol{\theta}}$), $\boldsymbol{U} = \boldsymbol{V}+\partial\boldsymbol{V}$, and $\boldsymbol{x}(0) = \boldsymbol{z}(0)+\partial\boldsymbol{z}$, the expected value of (P2-16) yields

$$\begin{aligned}&\mathbb{E}\left[\|\boldsymbol{M_\theta U}+\boldsymbol{N_\theta x}(0)\|_{\boldsymbol{I}}\right] = \\ &\mu^2\left(\|\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{V}+\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{z}(0)\|_{\boldsymbol{I}} + \mathrm{t}_{\partial\boldsymbol{\theta},\partial\boldsymbol{\theta}}\right) + 2\mu\ \mathrm{t}_{\boldsymbol{\theta}_\mathrm{t},\partial\boldsymbol{\theta}} + \mathrm{t}_{\boldsymbol{\theta}_\mathrm{t},\boldsymbol{\theta}_\mathrm{t}}\end{aligned} \tag{P2-17}$$

with

$$\begin{aligned}\mathrm{t}_{\boldsymbol{\theta}_\mathrm{t},\boldsymbol{\theta}_\mathrm{t}} &= \mathrm{trace}(\boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}}^T\boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}}\boldsymbol{\Sigma}) + \mathrm{trace}(\boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}}^T\boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}}\boldsymbol{\Sigma}_0) \\ \mathrm{t}_{\boldsymbol{\theta}_\mathrm{t},\partial\boldsymbol{\theta}} &= \mathrm{trace}(\boldsymbol{M}_{\boldsymbol{\theta}_\mathrm{t}}^T\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}) + \mathrm{trace}(\boldsymbol{N}_{\boldsymbol{\theta}_\mathrm{t}}^T\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}_0) \\ \mathrm{t}_{\partial\boldsymbol{\theta},\partial\boldsymbol{\theta}} &= \mathrm{trace}(\boldsymbol{M}_{\partial\boldsymbol{\theta}}^T\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}) + \mathrm{trace}(\boldsymbol{N}_{\partial\boldsymbol{\theta}}^T\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}_0).\end{aligned}$$

The optimal μ minimizing (P2-17) is not necessarily 0 but a function of $\partial\boldsymbol{\theta}$, which proves result 1:

$$\mu = -\frac{\mathrm{t}_{\boldsymbol{\theta}_\mathrm{t},\partial\boldsymbol{\theta}}}{\|\boldsymbol{M}_{\partial\boldsymbol{\theta}}\boldsymbol{V}+\boldsymbol{N}_{\partial\boldsymbol{\theta}}\boldsymbol{z}(0)\|_{\boldsymbol{I}} + \mathrm{t}_{\partial\boldsymbol{\theta},\partial\boldsymbol{\theta}}}.$$

For $\partial\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t}$, $\mu = -1$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_\mathrm{t} - \boldsymbol{\theta}_\mathrm{t} = \boldsymbol{0}$, i.e., the estimator is not scale-invariant, which proves result 2. □

6 Simulation Results

In this section, we utilize the three discussed methods for learning the cost function's parameters.

6.1 Simulation Setup

System dynamics and constraints

We consider one linear system and one nonlinear system with dynamics

$$\boldsymbol{x}(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \tag{P2-18a}$$

$$\boldsymbol{x}(k+1) = \begin{bmatrix} 1 & 1-u(k)^2 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k). \tag{P2-18b}$$

The inputs are constrained as $|u(k)| \leq 1$.

Cost function

The true cost function is chosen as

$$\boldsymbol{\theta}_{\mathrm{t}}{}^{T} \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{Z})$$

with

$$\boldsymbol{\theta}_{\mathrm{t}} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\phi}(\boldsymbol{V}, \boldsymbol{Z}) = \begin{bmatrix} \sum_{k=0}^{N} z_{1,k}^2 \\ \sum_{k=0}^{N} z_{2,k}^2 \\ \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2 \\ \sum_{k=0}^{N} v_k^2 \end{bmatrix}$$

with $\boldsymbol{z}_k = [z_{1,k} \; z_{2,k}]^T$ and $N = 10$. As the cost function is scale-invariant, we fix one parameter and learn

$$\hat{\boldsymbol{\theta}}^T = \begin{bmatrix} \hat{\theta}_1 & \hat{\theta}_2 & \hat{\theta}_3 & 1 \end{bmatrix}.$$

Data generation

In order to generate the data, we sample the initial conditions $\boldsymbol{z}(0)$ from $\boldsymbol{z}(0) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Using $\boldsymbol{z}(0)$, we obtain the optimal input sequence $\boldsymbol{V}$ using (P2-2). Then, the (sub-optimal) demonstration is generated as

$\boldsymbol{U} \sim \mathcal{N}(\boldsymbol{V}, \sigma_u^2 \boldsymbol{I}_{10})$ and $\boldsymbol{x}(0) \sim \mathcal{N}(\boldsymbol{z}(0), \sigma_0^2 \boldsymbol{I}_2)$, where σ_u and σ_0 are varied logarithmically as

$$\begin{aligned}\sigma_u, \sigma_0 \in \{&0.0001, 0.000215, 0.000464, 0.001,\\ &0.00215, 0.00464, 0.01,\\ &0.0215, 0.0464, 0.1,\\ &0.215, 0.464, 1\}\end{aligned}$$

Evaluation criterion

We evaluate the learned parameters by comparing $\boldsymbol{V}$ resulting from (P2-2) with $\boldsymbol{\theta}_\mathrm{t}$ and $\hat{\boldsymbol{V}}$ resulting from (P2-2) with $\hat{\boldsymbol{\theta}}$ as

$$\text{error} = \frac{\|\hat{\boldsymbol{V}} - \boldsymbol{V}\|_2}{\|\boldsymbol{V}\|_2}. \tag{P2-19}$$

6.2 Learning Results

For every tuple $\{\sigma_u, \sigma_0\}$, we repeat the data generation process 1000 times and learn $\hat{\boldsymbol{\theta}}$ using the three inverse learning methods.

Fig. P2-2 illustrates the median of the error in (P2-19) for the 1000 trials for $\{\sigma_u, \sigma_0\}$ and the three learning methods. For $\sigma_u = \sigma_0 = 0$, error $= 0$ for all three methods. In the presence of noise $\sigma_u, \sigma_0 > 0$, the learning results degrade differently. The error increases for larger standard deviations $\{\sigma_u, \sigma_0\}$ for all three methods. However, it can be seen that with increased noise levels (sub-optimal data), the error increases more quickly for Method 1, whereas the errors remain smaller for Method 2 and Method 3. Now, consider small σ_u. For increased σ_0, the error for Method 3 is significantly smaller than Method 2, which is expected since Method 3 optimizes over $\boldsymbol{z}(0)$. For small σ_0 and increasing σ_u, Method 3 also outperforms Method 2, which suggests that optimizing over the initial condition is advantageous even for small uncertainties in the initial conditions. Note that the standard deviations $\sigma_u = \sigma_0 > 0.1$ are unrealistically high as $|u| \leq 1$.

Fig. P2-3 shows a more detailed statistical evaluation of the error in (P2-19) for the 1000 trials and $\sigma_u = \sigma_0$. First, it can be seen that the estimation with Method 2 and Method 3 have a lower error compared to Method 1. The errors tend to be lower for the nonlinear system compared to the linear system, which can best be seen for $\sigma_0 = \sigma_u < 0.01$. The relatively low errors of Method 2 and Method 3 suggest the superiority of the maximum likelihood formulation over the convex relaxation approach of Method 1 measured with respect to the predictive performance, i.e., (P2-19).

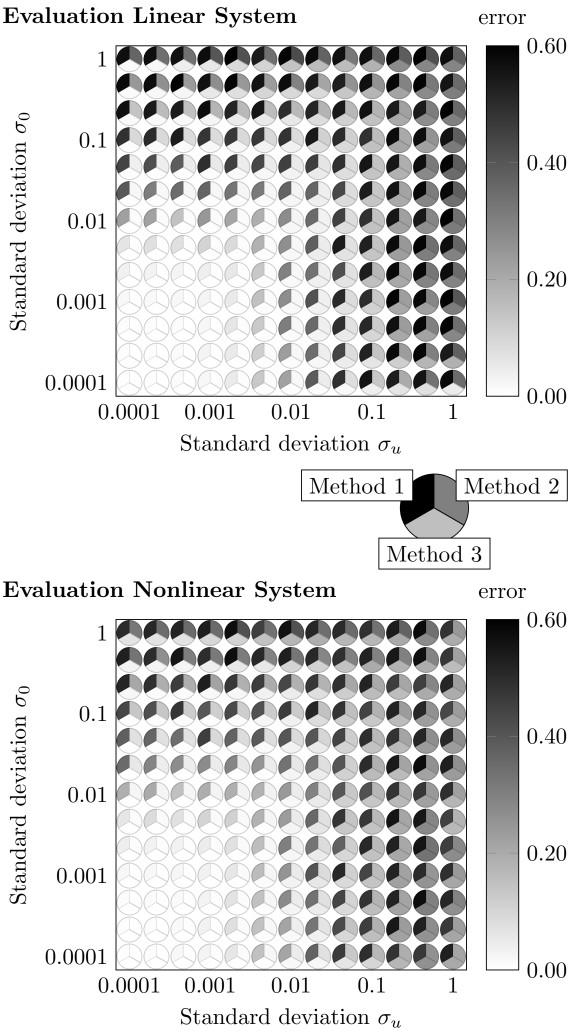


Figure P2-2. Median of error $\in [0, 0.6]$ for different standard deviations σ_u and σ_0 (color map from white to black) for the three inverse learning methods.

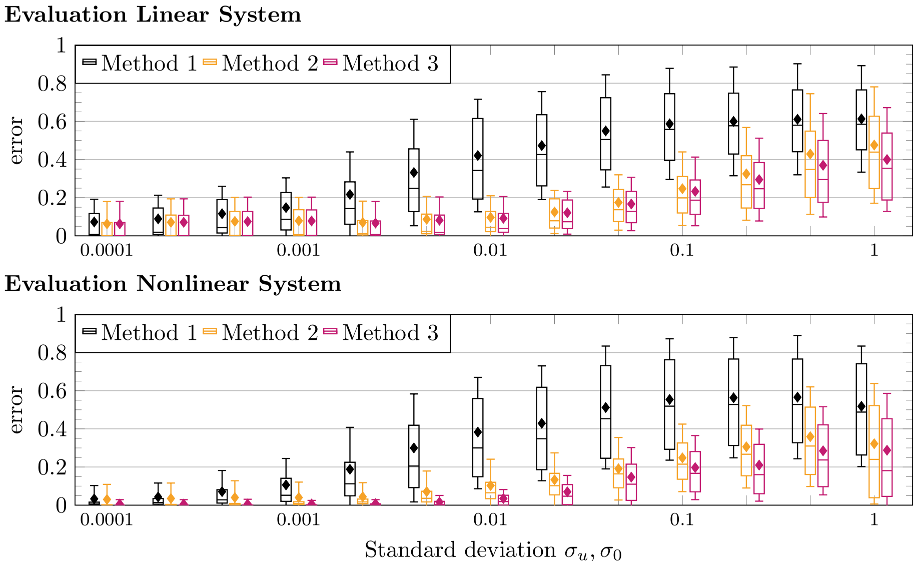


Figure P2-3. Evaluation of the error in (P2-19) for the three inverse learning methods for different noise levels $\sigma_u = \sigma_0$. The diamond symbol and vertical line represent the mean and the median, respectively; the box edges represent the 25th and the 75th percentiles; and the whiskers represent the standard deviation.

6.3 Computation Time

Table P2-1 states the median computation time for all samples of the initial conditions for the three learning methods using MATLAB with the hardware configuration: 3.1 GHz Intel Core i7, 16 GB 1867 MHz DDR3, and Intel Iris Graphics 6100 1536 MB. Method 1 is convex and, therefore, the computation is cheap and requires less than one second. Method 2 is computationally slightly more involved but can still be solved in around one second. Method 3 is more demanding as also the initial condition, $\boldsymbol{z}(0)$, is an optimization variable.

Table P2-1. Computation time

System	Method	Median over all samples
	Method 1	$T_{\mathrm{L,M1}} = 0.148\mathrm{s}$
Linear (P2-18a)	Method 2	$T_{\mathrm{L,M2}} = 1.19\mathrm{s}\ (8.04T_{\mathrm{L,M1}})$
	Method 3	$T_{\mathrm{L,M3}} = 3.07\mathrm{s}\ (20.8T_{\mathrm{L,M1}})$
	Method 1	$T_{\mathrm{NL,M1}} = 0.142\mathrm{s}$
Nonlinear (P2-18b)	Method 2	$T_{\mathrm{NL,M2}} = 0.584\mathrm{s}\ (4.12T_{\mathrm{NL,M1}})$
	Method 3	$T_{\mathrm{NL,M3}} = 1.44\mathrm{s}\ (10.2T_{\mathrm{NL,M1}})$

Fig. P2-4 shows a more detailed statistical evaluation of the computation times for $\sigma_u = \sigma_0$. The three learning methods have their respective peak computation times at different noise levels, i.e., Method 1's maximum computation times are highest for lower noise levels (peak of median at $\sigma_u = \sigma_0 = 0.0001$); Method 2's maximum times occur for slightly higher noise levels (peak at $\sigma_u = \sigma_0 = 0.00215$); whereas Method 3's peak is for high noise levels (peak at $\sigma_u = \sigma_0 = 0.0215$). For all methods and noise levels, the mean value is higher than the median, which is expected as the median is less susceptible to outliers, i.e., instances with particularly long computation times.

7 Conclusion

This paper presented three inverse optimal control methods; one method that uses a convex relaxation of the KKT optimality conditions and two methods that combine the KKT conditions with maximum likelihood estimation. It proposed a branch-and-bound-style algorithm for the maximum likelihood formulation, which is based on likelihood arguments to systematically deal with constraints in the presence of noisy data. A simulation study exemplified the three inverse learning methods with both a constrained, linear and a nonlinear system. The results showed that the likelihood estimation methods can be implemented quite efficiently and yield robust learning results, whereas the convex method is computationally efficient but less robust to noise in the training data.

Bibliography

[1] W. Karush, "Minima of functions of several variables with inequalities as side constraints", Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[2] H. W. Kuhn and A. W. Tucker, "Nonlinear programming", in *2nd Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1951.

[3] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations", *Int. J. Robot. Res.*, vol. 36, no. 13–14, pp. 1474–1488, 2017.

[4] M. Menner, P. Worsnop, and M. N. Zeilinger, "Constrained inverse optimal control with application to a human manipulation task", *IEEE Trans. Control Syst. Technol.*, 2019. DOI: 10.1109/TCST.2019.2955663.

[5] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control", *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[6] R. E. Kalman, "When is a linear control system optimal?", *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[7] M. Menner and M. N. Zeilinger, "Convex formulations and algebraic solutions for linear quadratic inverse optimal control problems", in *Eur. Control Conf.*, 2018, pp. 2107–2112.

[8] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach", *Auton. Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[9] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations", *ArXiv:1812.07084*, 2018.

[10] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty", *IEEE Robot. and Automat. Lett.*, 2020. DOI: 10.1109/LRA.2020.2974427.

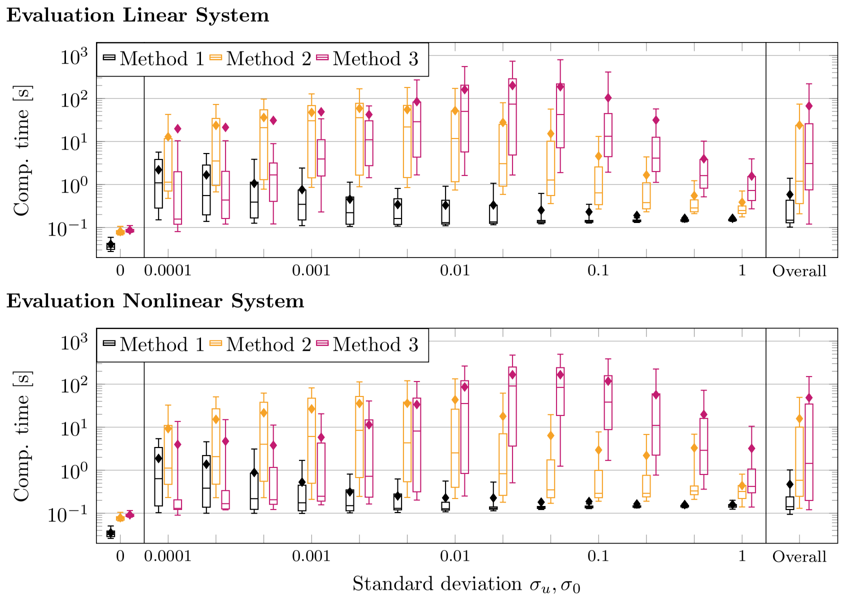


Figure P2-4. Statistical evaluation of the computation time for the three inverse learning methods with $\sigma_u = \sigma_0$. The diamond symbol and vertical line represent the mean and the median, respectively; the box edges represent the 75th and the 25th percentiles; and the whiskers represent the 90th and the 10th percentiles.

[11] M. Menner, K. Berntorp, M. N. Zeilinger, and S. Di Cairano, "Inverse learning for human-adaptive motion planning", in *58th IEEE Conf. Decision and Control*, 2019, pp. 809–815.

[12] P. M. Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn, "Data-driven inverse optimization with imperfect information", *Math. Program.*, vol. 167, no. 1, pp. 191–234, 2018.

[13] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning", in *23rd AAAI Conf. Artif. Intell.*, vol. 8, 2008, pp. 1433–1438.

[14] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples", in *29th Int. Conf. Mach. Learning*, 2012, pp. 475–482.

[15] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization", in *33rd Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[16] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[17] K. G. Murty and F.-T. Yu, *Linear complementarity, linear and nonlinear programming.* Berlin: Heldermann, 1988, vol. 3.

[18] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives", *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

Part B.

Automating a Rehabilitation Robot using Supervised Learning

Paper P3

Using Human Ratings for Feedback Control: A Supervised Learning Approach With Application to Rehabilitation Robotics

Marcel Menner, Lukas Neuner, Lars Lünenburger, and Melanie N. Zeilinger

Published in *IEEE Transactions on Robotics*

Abstract: This article presents a method for tailoring a parametric controller based on human ratings. The method leverages supervised learning concepts in order to train a reward model from data. It is applied to a gait rehabilitation robot with the goal of teaching the robot how to walk patients physiologically. In this context, the reward model judges the physiology of the gait cycle (instead of therapists) using sensor measurements provided by the robot, and the automatic feedback controller chooses the input settings of the robot to maximize the reward. The key advantage of the proposed method is that only a few input adaptations are necessary to achieve a physiological gait cycle. Experiments with nondisabled subjects show that the proposed method permits the incorporation of human expertise into a control law and to automatically walk patients physiologically.

©2020 IEEE. Reprinted, with permission, from M. Menner, L. Neuner, L. Lünenburger, and M. N. Zeilinger, "Using Human Ratings for Feedback Control: A Supervised Learning Approach With Application to Rehabilitation Robotics," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 789–801, June 2020.

1 Introduction

Humans can perform very complex tasks that are difficult to achieve with autonomous systems. The dependency on human supervision or expertise still restricts the efficient operation of many complex systems. An important domain where human expertise is usually needed is rehabilitation robotics, where we consider the robot-assisted gait trainer Lokomat® [1] in this paper, see Fig. P3-1. Robotic systems like the Lokomat have recently been introduced in gait rehabilitation following neurological injuries with the goal of mitigating the limitations of conventional therapy [2]–[6]. However, training with such robots still requires the supervision and interaction of experienced therapists [1].

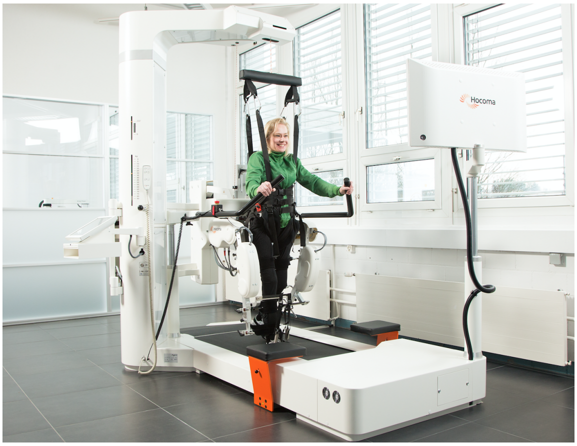


Figure P3-1. Lokomat® gait rehabilitation robot (Hocoma AG, Volketswil, CH).

Gait rehabilitation with the Lokomat currently requires physiotherapists to manually adjust the mechanical setup and input settings, e.g., the speed of the treadmill or the range of motion, in order to bring patients into a physiological and safe gait cycle. Therapists have to be trained specifically for the device and acquire substantial experience in order to achieve good input settings. Although there are guidelines for their adjustment [7], it remains a heuristic process, which strongly depends on the knowledge and experience of the therapist. Automatic adaptation of input settings can reduce the duration of therapists' schooling, improve patient training, make the technology more broadly applicable, and can be more cost-effective. In this work, we propose a method to automatically adapt the input settings.

Although the motivation behind this work is in the domain of rehabilitation robotics, the proposed method addresses general human-in-the-loop scenarios, where expert knowledge can improve system operation.

In this paper, we propose a two-step approach to achieve automatic input adaptations: First, we define a feature vector to characterize the gait cycle and postulate a reward model to judge the physiology of the gait cycle using the feature vector. The reward model is trained with therapists' ratings using a supervised learning technique, where the feature vector is obtained from sensor measurements provided by the robot. The sensor measurements are the angle, torque, and power of both the hip and knee joints of the robot. Second, we use the gradient of the reward model to determine input adaptations that achieve the desired gait cycle. This involves a steady-state model to relate the gradient of the reward model with respect to the feature vector (high dimensional) to input settings (low dimensional) that adjust the gait cycle. A key component in the proposed formulation is that the reward model and its gradient are formulated as functions of the feature vector rather than the input settings. The high dimensionality of the feature vector allows us to use one model for all human subjects with very different body types, which enables a very efficient online application of the proposed method. In order to train both the reward model and the steady-state model, we collected data with various physiological and non-physiological input settings from 16 nondisabled subjects. The subjects were instructed to be passive while being walked by the robot in order to imitate patients with limited or no ability to walk in the early stages of recovery. Experiments with ten nondisabled subjects highlighted the ability of the proposed method to improve the walking pattern within a few adaptations starting from multiple initially non-physiological gait cycles.

Related Work

Adaptive control strategies have been the subject of a body of research in robotic gait trainers with the goal of improving the therapeutic outcome of treadmill training [8]–[13]. The work in [8] presents multiple strategies for automatic gait cycle adaptation in robot-aided gait rehabilitation based on minimizing the interaction torques between device and patient. Biomechanical recordings providing feedback about a patient's activity level are introduced in [9], [10]. Automated synchronization between treadmill and orthosis based on iterative learning is introduced in [11]. In [12], a path control method is proposed to allow voluntary movements along a physiological path defined by a virtual tunnel. An algorithm to adjust the mechanical impedance of an orthosis joint based on the level of support required by a patient is proposed in [13]. Further research in the domain of rehabilitation robotics is presented, e.g., in [14], [15]. In [14], the human motor system is modeled and analyzed as approximating an optimization problem trading off effort and kinematic error. In [15], a patient's psychological state is estimated to judge their mental engagement. Different from the work in rehabilitation robotics presented in [8]–[15], we present a method for input setting adaptation of a rehabilitation robot based on a feedback controller, which is derived from human ratings.

In the following, we discuss research directions related to the techniques employed in the proposed approach.

Gait cycle classification is often used to distinguish human subjects according to two classes [16]–[19], e.g., young/elderly or healthy/impaired. In [16], a supervised learning method for automatic recognition of movement patterns is presented to discriminate gait patterns of young and elderly people. In order to improve classification performance, [17] employs a kernel-based principal component analysis for the extraction of features. Gait patterns are also used to diagnose diseases that symptomatically cause gait abnormalities, e.g., [18], [19]. Different from [16]–[19], this paper does not aim to identify or classify human individuals but to generalize from data of multiple individuals by classifying gait patterns according to their physiology. Further, the obtained classifier is not used to predict discrete/binary classes but as a continuous reward, which is maximized using feedback control.

Reinforcement learning uses a trial and error search to find a control policy [20]–[28]. The framework proposed in [21] allows human trainers to shape a policy using approval or disapproval. In [22], human-generated rewards in a reinforcement learning framework are employed for a 2-joint velocity control task. In [23], a policy-shaping method is presented where human feedback is not used as a reward signal but directly as a label for the policy. In [24], human preferences are learned through ratings based on a pairwise comparison of trajectories with the goal of reducing human feedback. In [25], a robot motion planning problem is considered, where users provide a ranking of paths that enable the evaluation of the importance of different constraints. In [26], a method is presented that actively synthesizes queries to a user to update a distribution over reward parameters. In [27], user preferences in a traffic scenario are learned based on human guidance by means of feature queries. In [28], human ratings are used to learn a probability distribution of individual preferences modeled as a Markov decision process. While a reinforcement learning approach could in principle be applied to the considered problem, the online application of these methods typically requires a few hundred human ratings to learn a policy. This is infeasible when working with a patient, where a comparatively small number of feedback rounds has to be sufficient. The main difference between our method and reinforcement learning is that we do not use trial and error search but we build a reward model that is maximized online.

Inverse optimal control and *inverse reinforcement learning* aim at learning a reward model or cost model from demonstrations of human behavior [29]–[46]. Inverse optimal control methods model demonstrations to be the result of an optimal control problem [33]–[40] and often aim at transferring human expertise to an autonomous system, e.g., for humanoid locomotion [33], [34], identifying human movements [35]–[37], robot manipulation tasks [38], or autonomous driving [39]. In inverse reinforcement learning [41]–[46], demonstrations are typically modeled to be the result of probabilistic decision-making in a Markov decision process. The fundamental difference between the proposed method and inverse optimal control/inverse reinforcement learning methods is the utilization of ratings instead of demonstrations to learn a reward model.

2 Hardware Description & Problem Definition

The Lokomat® gait rehabilitation robot (Hocoma AG, Volketswil, CH) is a bilaterally driven gait orthosis that is attached to the patient's legs by Velcro straps. In conjunction with a bodyweight support system, it provides controlled flexion and extension movements of the hip and knee joints in the sagittal plane. Leg motions are repeated based on predefined but adjustable reference trajectories. Additional passive foot lifters ensure ankle dorsiflexion during the swing. The bodyweight support system partially relieves patients from their bodyweight via an attached harness. A user interface enables gait cycle adjustments by therapists via a number of input settings [1], [10].

Input Settings

One important task of the therapist operating the Lokomat is the adjustment of the input settings to obtain a desirable gait trajectory. A total of 13 input settings can be adjusted to affect the walking behavior, which are introduced in Table P3-1. In this work, we propose a method that can automate or assist the therapists in the adjustment of the input settings by measuring the gait cycle.

2.1 State-of-the-Art Therapy Session

The current practice of gait rehabilitation with the Lokomat includes the preparation and setup of the patient and device, actual gait training, and

Table P3-1. Input Settings of the Lokomat

Input Setting & Description	Step-size	Range
Hip Range of Motion (Left & Right) Defines the amount of flexion and extension	3°	23°, 59°
Hip Offset (Left & Right) Shifts movements towards extension or flexion	1°	-5°, 10°
Knee Range of Motion (Left & Right) Defines amount of flexion	3°	32°, 77°
Knee Offset (Left & Right) Shifts movement into flexion for hyperextension correction	1°	0°, 8°
Speed Sets the treadmill speed	0.1km/h	0.5km/h, 3km/h
Orthosis speed Defines the orthosis and affects walking cadence	0.01	0.15, 0.8
Bodyweight Support Defines carried weight for unloading	continuous	0kg, 85kg
Guidance Force Sets amount of assistance	5%	0%, 100%
Pelvic Defines lateral movement	1cm	0cm, 4cm

finally removing the patient from the system [7]. Gait training is further divided into three phases:

1. Safe walk: The patient is gradually lowered until the dynamic range for the bodyweight support is reached. The purpose of this first phase is to ensure a safe and non-harmful gait cycle.

2. Physiological walk: After ensuring safe movements, the gait cycle is adjusted so that the patient is walked physiologically by the robot.

3. Goal-oriented walk: The gait cycle is adjusted to achieve therapy goals for individual sessions while ensuring that the patient's gait remains physiological.

In this paper, we focus on the physiological walk. In a state-of-the-art therapy session, therapists are advised to follow published heuristic guidelines on how to adjust the input settings based on observations (visual feedback) in order to reach a physiological walk. Three examples of the heuristic guidelines are as follows: If the step length does not match walking speed, then the hip range of motion or treadmill speed should be adjusted; if the heel strike is too late, then the hip offset or the hip range of motion should be decreased; if the foot is slipping, then the orthosis speed or the knee range of motion should be decreased. An extended overview of heuristics can be found in [7]. This heuristic approach requires experience and training with experts, which incurs high costs and limits the availability of the rehabilitation robot due to the small number of experienced experts. The proposed method aims to alleviate this limitation as described in the following.

2.2 Technological Contribution

We propose a method for automatically suggesting suitable input settings for the Lokomat based on available sensor measurements in order to walk patients physiologically. The proposed framework can be used for recommending input settings for therapists, automatic adaptation of input settings, or as an assistive method for therapists during schooling with the Lokomat. Fig. P3-2 illustrates the proposed method as a recommendation system. The method is derived assuming that the mechanical setup of the Lokomat is done properly, such that the purpose of adapting the input settings is the improvement of the gait cycle and not corrections due to an incorrect setup.

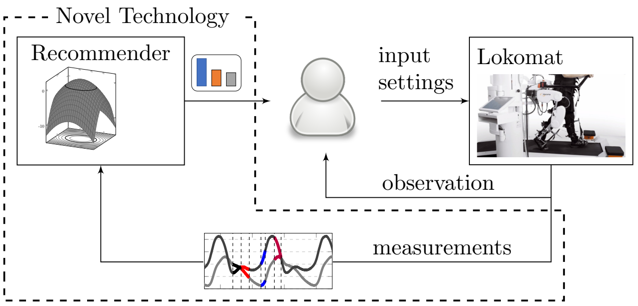


Figure P3-2. Overview of the proposed method as a recommendation system. The novel technology (dashed lines) augments the state-of-the-art control loop of a therapist and the Lokomat. Sensor measurements of angle, torque, and power of both hip and knee joints provided by the Lokomat are used to compute recommendations for the input adaptations.

3 Controller Design based on Human Ratings

This section describes the proposed human feedback-based controller. In the setup considered, input settings $\boldsymbol{s} \in \mathbb{R}^m$ of the controlled system lead to a gait cycle represented by a feature vector $\boldsymbol{x} \in \mathbb{R}^n$ in steady-state:

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{s}), \tag{P3-1}$$

where $\boldsymbol{f}$ is an unknown function. For the considered application, the input settings $\boldsymbol{s}$ are given in Table P3-1 and the feature vector $\boldsymbol{x}$ is composed of statistical features of measurements, which characterize the gait cycle and are further discussed in Section 4. Here, the notion of a steady-state means that any transients due to an input adaptation have faded. The control objective is to find input settings $\boldsymbol{s}^\star$ for which $\boldsymbol{x}^\star = \boldsymbol{f}(\boldsymbol{s}^\star)$ represents a desired system state, i.e., a physiological gait cycle in the considered application.

Control Law and Conceptual Idea

The method is based on a reward model, reflecting the control objective, and a steady-state model, associating a feature vector with an input setting. The reward model is a function that assigns a scalar value to the feature vector estimating an expert rating of the "goodness" of the feature vector. The reward thereby provides a direction of improvement for the feature vector, which is mapped to a change in input settings via the steady-state model.

We define the control law in terms of input adaptations $\Delta\boldsymbol{s}$:

$$\Delta\boldsymbol{s} = \boldsymbol{f}^{-1}(\alpha\Delta\boldsymbol{x} + \boldsymbol{x}) - \boldsymbol{s},$$

where $\Delta\boldsymbol{x}$ is the direction of improvement, $\boldsymbol{f}^{-1} : \mathbb{R}^n \to \mathbb{R}^m$ is the steady-state model (the inverse mapping of $\boldsymbol{f}$ in (P3-1)), and $\alpha > 0$ is the gain of the control law. We compute $\Delta\boldsymbol{x}$ as the gradient of the reward model $r(\boldsymbol{x}) \in \mathbb{R}$, i.e.,

$$\Delta\boldsymbol{x} = \nabla_{\boldsymbol{x}} r(\boldsymbol{x}).$$

Fig. P3-3 shows an example of a reward model and indicates how its gradient is used for feedback control using the steady-state model. Both models $r(\boldsymbol{x})$ and $\boldsymbol{f}^{-1}(\cdot)$ are inferred from data. In order to train the reward model, we utilize ratings on an integer scale as samples of the reward model, i.e., $r_i = 1, ...S$, where $r_i = 1$ is the worst and $r_i = S$ is the best rating. Additionally, we train a steady-state model $\boldsymbol{f}^{-1}(\cdot)$ to relate the direction of improvement suggested by the reward model to the corresponding input adaptation (bottom part of Fig. P3-3). In order to build both the reward model and the steady-state model, N training samples are collected. Each training sample with index i consists of a feature vector $\boldsymbol{x}_i$, the input settings $\boldsymbol{s}_i$, and the corresponding rating $r_i \in \{1, ..., S\}$:

$$\{\boldsymbol{x}_i, \boldsymbol{s}_i, r_i\}_{i=1}^{N}. \tag{P3-2}$$

Note that throughout this paper, the feature vector $\boldsymbol{x}$ is normalized using collected data $\boldsymbol{x}_i$ such that the collected data are zero-mean with unit-variance in order to account for different value ranges and units, cf., [47].

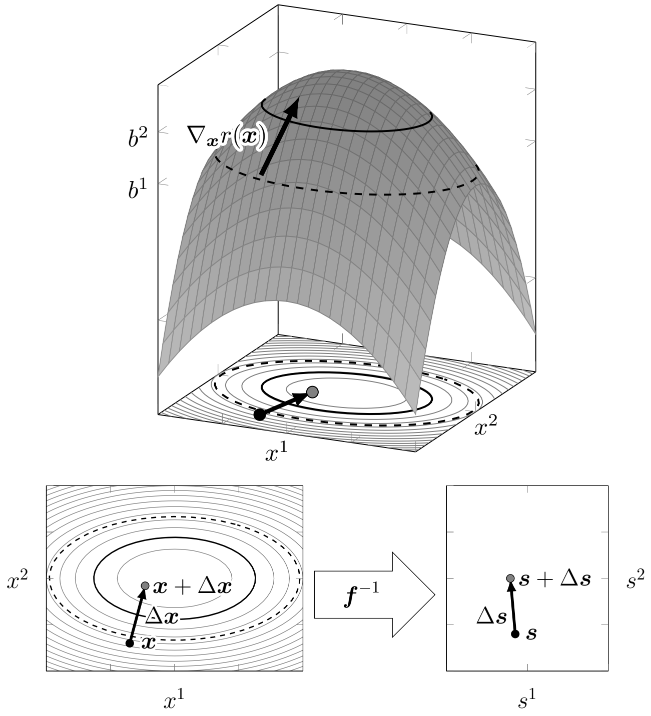


Figure P3-3. Top: Example of reward model with gradient vector $\nabla_{\boldsymbol{x}} r(\boldsymbol{x})$ where $\boldsymbol{x} = [x^1 \; x^2]^\mathsf{T}$ and projected level sets onto the $x^1 - x^2$ plane. The example shows a case of three ratings $r_i = 1, 2, 3$ separated by two classification boundaries indicated as solid black and dashed black ellipses. Bottom: Steady-state model to compute $\Delta\boldsymbol{s}$ from $\Delta\boldsymbol{x}$ where $\boldsymbol{s} = [s^1 \; s^2]^\mathsf{T}$.

Outline

The reward model is trained with the feature vector $\boldsymbol{x}_i$ and its corresponding rating r_i in (P3-2) using a supervised learning technique (Section 3.1). The resulting reward model is then used to compute the gradient $\nabla_{\boldsymbol{x}} r(\boldsymbol{x})$ as the direction of improvement. Finally, a steady-state model relates this direction of improvement with necessary changes in input settings $\boldsymbol{s}$. The steady-state model is computed using a regression technique (Section 3.2).

3.1 Reward Model using Supervised Learning

The first step of the framework is the learning of a reward model reflecting the physiology of the gait based on supervised learning techniques [47]. The reward model is a continuous function, i.e., it provides a reward for all $\boldsymbol{x}$, whereas observations $\boldsymbol{x}_i$ are potentially sparse.

In view of the considered application, we postulate a reward model of the form:

$$r(\boldsymbol{x}) = 0.5\boldsymbol{x}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b, \tag{P3-3}$$

where $\boldsymbol{W} = \boldsymbol{W}^{\mathsf{T}} \prec 0$, $\boldsymbol{w} \in \mathbb{R}^n$, and $b \in \mathbb{R}$ are the parameters to be learned from expert ratings given in the form of integers on a scale from 1 to S. The rationale for selecting a quadratic model with negative definite $\boldsymbol{W}$ is the observation that the gait degrades in all relative directions when changing input settings from a physiological gait. The important properties of this reward model are that a vanishing gradient indicates that global optimality has been reached and its computational simplicity. This motivates the gradient ascent method for optimizing performance.

In order to learn $\boldsymbol{W}$, $\boldsymbol{w}$, and b in (P3-3), we construct $S-1$ classification problems. These $S-1$ classification problems share the parameters $\boldsymbol{W}$, $\boldsymbol{w}$, and b of the reward model and the corresponding classification boundaries are given by

$$r^l(\boldsymbol{x}) = 0.5\boldsymbol{x}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b^l$$

for all $l = 1, ..., S-1$ with $b^l = b - l - 0.5$ separating the S different ratings such that $r^l(\boldsymbol{x}_i) > 0$ if $r_i > l + 0.5$. Further, for each data sample i and each l, we define

$$y_i^l = \begin{cases} 1 & \text{if } r_i > l + 0.5 \\ -1 & \text{else.} \end{cases}$$

Hence, an ideal reward model with perfect data and separation satisfies

$$y_i^l r^l(\boldsymbol{x}_i) \geq 0 \quad \begin{array}{c} \forall\ i = 1, ..., N \\ \forall\ l = 1, ..., S-1. \end{array} \tag{P3-4}$$

In order to allow for noisy data and imperfect human feedback, (P3-4) is relaxed to find $r^l(\boldsymbol{x})$ that satisfies (P3-4) "as closely as possible" by introducing a margin $\xi_i^l \geq 0$. This approach is closely related to a Support Vector Machine, cf., [47], with a polynomial kernel function of degree two. The functions $r^l(\boldsymbol{x})$ correspond to $S-1$ classification boundaries in a multi-category classification framework. The parameters $\boldsymbol{W}$, $\boldsymbol{w}$, and b of the reward model (P3-3) are computed by solving the following optimization problem using L1 regularization:

$$\begin{aligned}
\underset{\boldsymbol{W},\boldsymbol{w},b^l,b,\xi_i^l}{\text{minimize}} \quad & \sum_{l=1}^{S-1}\sum_{i=1}^{N} \xi_i^l + \lambda_1 \cdot (\|\boldsymbol{W}\|_1 + \|\boldsymbol{w}\|_1) && \text{(P3-5a)}\\
\text{subject to} \quad & y_i^l r^l(\boldsymbol{x}_i) \geq 1 - \xi_i^l, \qquad \forall\, i = 1, ..., N && \text{(P3-5b)}\\
& \xi_i^l \geq 0, \qquad \forall\, l = 1, ..., S-1 && \text{(P3-5c)}\\
& r^l(\boldsymbol{x}_i) = 0.5\boldsymbol{x}_i^\mathsf{T}\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b^l && \text{(P3-5d)}\\
& b^l = b - l - 0.5 && \text{(P3-5e)}\\
& \boldsymbol{W} = \boldsymbol{W}^\mathsf{T} \prec 0 && \text{(P3-5f)}
\end{aligned}$$

where $\lambda_1 > 0$ controls the trade-off between minimizing the training error and model complexity captured by the norm $\|\boldsymbol{W}\|_1 = \sum_{j=1}^{n}\sum_{k=1}^{n}|W_{jk}|$ (elementwise 1-norm) and $\|\boldsymbol{w}\|_1$, which is generally applied to avoid overfitting of a model and is sometimes also called lasso regression [47].

3.2 Feedback Control using Reward Model

The second step of the proposed framework is to exploit the trained reward model for feedback control. The idea is *(i)* to use the gradient of the reward model as the direction of improvement and *(ii)* to relate this gradient to a desired change in inputs with a steady-state model.

(i) Gradient of reward model

The gradient of the inferred reward model is the direction of best improvement. The control strategy is to follow this gradient in order to maximize reward. The gradient of the quadratic reward model in (P3-3) is

$$\Delta\boldsymbol{x} = \nabla_{\boldsymbol{x}} r(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}.$$

(ii) Mapping of gradient to setting space with steady-state model

In order to advance the system along the gradient direction, we relate the direction of improvement $\Delta\boldsymbol{x}$ to a change in input settings with a steady-state model $\boldsymbol{f}^{-1}$. We use a first order approximation $\boldsymbol{s} = \boldsymbol{f}^{-1}(\boldsymbol{x}) \approx$

$\boldsymbol{Mx} + \boldsymbol{m}$ with $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, $\boldsymbol{m} \in \mathbb{R}^m$ to compute the change in input settings $\Delta\boldsymbol{s}$ as

$$\Delta\boldsymbol{s} = \boldsymbol{M}(\alpha(\boldsymbol{Wx} + \boldsymbol{w}) + \boldsymbol{x}) + \boldsymbol{m} - \boldsymbol{s}, \tag{P3-6}$$

where α can be interpreted as feedback gain or the learning rate in a gradient ascent method. The steady-state model is estimated as the least squares solution of the data in (P3-2):

$$\underset{\boldsymbol{M},\boldsymbol{m}}{\text{minimize}} \sum_{i=1}^{N} \|\boldsymbol{s}_i - \boldsymbol{Mx}_i - \boldsymbol{m}\|_2^2 + \lambda_2 \cdot (\|\boldsymbol{M}\|_1 + \|\boldsymbol{m}\|_1) \tag{P3-7}$$

where, again, we use $\lambda_2 > 0$ to control the trade-off between model fit and model complexity.

Using the quadratic reward model in (P3-3) and the linear steady-state model in (P3-7), the application of the proposed control strategy (P3-6) requires only matrix-vector multiplications, which is computationally inexpensive and can be performed online, cf., Algorithm P3-1 for an overview of the method. Additionally, as will be shown empirically, the application requires only a few online input adaptations.

Algorithm P3-1 Training and Application of the Method

```
Training                                                   ▷ rating needed
 1: Collect data set in (P3-2).
 2: Compute reward model $\boldsymbol{W}, \boldsymbol{w}, b$ with (P3-5) and steady-state model
    $\boldsymbol{M}, \boldsymbol{m}$ with (P3-7).
Online Algorithm
 3: do
 4:     Obtain feature vector $\boldsymbol{x}$ from measurement.
 5:     Apply adaptation $\Delta\boldsymbol{s} = \boldsymbol{M}(\alpha(\boldsymbol{Wx} + \boldsymbol{w}) + \boldsymbol{x}) + \boldsymbol{m} - \boldsymbol{s}$.
 6:     Wait until steady state is reached.
 7: while stopping criterion not fulfilled                  ▷ cf. Section 4.4
```

Remark 1. *As we will show in the analysis in Section 5, the linear mapping $\boldsymbol{s} \approx \boldsymbol{Mx} + \boldsymbol{m}$ yields sufficient accuracy for the considered application. For more complex systems, one might consider a different steady-state model, e.g., higher-order polynomials or a neural network to approximate $\boldsymbol{f}^{-1}$.*

Remark 2. *In principle, reinforcement learning could be applied to directly learn physiological settings. The proposed two-step and model-based method, in contrast, makes use of the higher dimensionality of the feature* *vector to characterize the gait cycle. Its key advantage is that fewer samples are required online and thus, fewer steps to find physiological settings, which is essential for the considered application.*

Remark 3. *The proposed method iteratively approaches the optimal settings $\boldsymbol{s}^\star$ with the gradient ascent method. This is important for the considered application to cautiously adapt the input settings of the robot with a human in the loop.*

Remark 4. *It is also possible to determine the direction of improvement using second-order derivatives of the reward model, e.g., using a Newton-Raphson method. However, as numerical second-order derivatives would be noisier, we choose first-order derivatives, which are simple and yield a more stable estimate of the best (local) improvement.*

4 Adaptation of Gait Rehabilitation Robot to Walk Patients Physiologically

In this section, we show how to apply the method presented in Section 3 to automatically adapt, or recommend a suitable adaptation, of the Lokomat's input settings in order to walk patients physiologically. A core element is the reward model that has been built on therapists' ratings and is used to judge the physiology of the gait. For simplicity, we adjust settings for the left and right leg symmetrically. This does not pose a problem for the presented study with nondisabled subjects but might be revisited for impaired subjects in future work.

In this work, we focus on the physiological walk and exclude the guidance force and the pelvic input settings as they are mainly used for goal-oriented walk [7]. This exclusion is valid for the physiological walk where the guidance force and pelvic settings are kept constant at 100% and 0cm, respectively. Hence, seven input settings are considered in the application of the method.

Safety

The proposed method is implemented to augment a previously developed safety controller that ensures the safe operation of the Lokomat. This safety controller intervenes if the input settings exceed nominal ranges for forces and positions of the robot's joints. An additional contingency controller stops the robot when the deviation of the measured gait trajectory and the desired gait trajectory becomes too large. In this way, the overall behavior is guaranteed to have the necessary safety requirements for the patient and the robot, yet among the safe input settings, the ones that improve the gait cycle are chosen. The reader is referred to [48] for a more detailed description of the Lokomat's safety mechanisms.

4.1 Gait Cycle

The walking of a human is a repetitive sequence of lower limb motions to achieve forward progression. The gait cycle describes such a sequence for one limb and commonly defines the interval between two consecutive events that describe the heel strike (initial ground contact) [49]. The gait cycle is commonly divided into two main phases, the stance and the swing phase. The stance phase refers to the period of ground contact, while the swing phase describes limb advancement. Fig. P3-4 illustrates the subdivision of these two main phases of the gait cycle into multiple sub-phases, beginning and ending with the heel strike. This results in a common description of gait using a series of discrete events and corresponding gait phases [49]. We focus on four particular phases of the gait cycle, which are emphasized in Fig. P3-4:

Heel strike (HS):	The moment of initial contact of the heel with the ground.
Mid-stance (MS):	The phase in which the grounded leg supports the full body weight.
Toe off (TO):	The phase in which the toe lifts off the ground.
Mid-swing (SW):	The phase in which the raised leg passes the grounded leg.

4.2 Evaluation of Gait Cycle and Data Collection

We derive the reward model based on the four phases. For evaluating the four gait phases, we introduce a scoring criterion in consultation with experienced therapists:

Rating 1: Safe, but not physiological.

Rating 2: Safe, not entirely physiological gait cycle.

Rating 3: Safe and physiological gait cycle.

Data Collection

A total of 16 nondisabled subjects participated in the data collection. The 16 subjects were between 158cm - 193cm (5'2" - 6'4") in height, 52kg - 93kg (115lbs - 205lbs) in weight, and aged 25 - 62. Informed consent for the use of the data has been received from all human subjects. The data collection for each subject involved an evaluation of the four gait phases by therapists for several input settings in order to collect data in a wide range of gait cycles. The nondisabled subjects were instructed to be passive throughout the data collection, i.e., they were walked by the robot. This allowed us to collect data for both physiological and non-physiological gait cycles. Measurements of the Lokomat were recorded for all evaluations. For each subject, the experienced therapists first manually tuned the input settings to achieve rating 3 for all four phases (Set 0 in Table P3-2), where the resulting input settings are referred to as initial physiological gait (IPG). Table P3-2 shows the input settings used in the data collection as deviations from the initial physiological gait. Each subject walked for approximately 60 seconds for each set of input settings, while the therapist provided evaluations of the walking pattern. The assessment started after a transient interval of approximately 15 seconds to ensure that the walking has reached a steady state. Note that the input settings resulting in a physiological gait pattern varied between subjects.

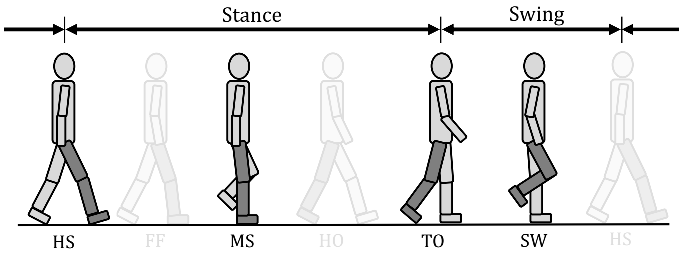


Figure P3-4. Gait phases in order: Heel strike, foot flat (FF), mid-stance, heel off (HO), toe off, mid-swing. Both FF and HO phase are not rated in this work, but presented for consistency with the literature [49].

The scoring criterion and the consideration of the four phases, as well as the data collection protocol, were introduced in consultation with clinical experts from Hocoma (U. Costa and P. A. Gonçalves Rodrigues, personal communication, Nov. 05, 2017). As a result, we obtained the chosen input settings, the corresponding ratings on an integer scale from 1 to 3, and the recording of measurements of the Lokomat. Next, we discuss the computation of the feature vector from the recorded measurements.

Table P3-2. Input Setting for Data Collection

Set	Input Settings	Value
0	Initial Set	IPG
1	Hip Range of Motion	IPG + 6°
2	Hip Range of Motion	IPG + 12°
3	Hip Range of Motion	IPG – 6°
4	Hip Range of Motion	IPG – 12°
5	Hip Offset	IPG + 4°
6	Hip Offset	IPG + 8°
7	Hip Offset	IPG – 5°
8	Hip Range of Motion, Hip Offset	IPG + 12° / IPG – 3°
9	Hip Range of Motion, Hip Offset	IPG + 12° / IPG + 3°
10	Hip Range of Motion, Hip Offset	IPG – 12° / IPG – 3°
11	Hip Range of Motion, Hip Offset	IPG – 12° / IPG + 3°
12	Knee Range of Motion	IPG + 6°
13	Knee Range of Motion	IPG + 12°
14	Knee Range of Motion	IPG – 9°
15	Knee Range of Motion	IPG – 15°
16	Knee Offset	IPG + 4°
17	Knee Offset	IPG + 8°
18	Knee Range of Motion / Knee Offset	IPG + 15° / IPG + 6°
19	Knee Range of Motion / Knee Offset	IPG + 21° / IPG + 6°
20	Speed	IPG + 0.5km/h
21	Speed	IPG + 1.0km/h
22	Speed	IPG – 0.5km/h
23	Speed	IPG – 1.0km/h
24	Orthosis Speed	IPG + 0.03
25	Orthosis Speed	IPG + 0.05
26	Orthosis Speed	IPG – 0.03
27	Orthosis Speed	IPG – 0.05
28	Bodyweight Support	IPG + 15%
29	Bodyweight Support	IPG + 30%
30	Bodyweight Support	IPG – 15%
31	Bodyweight Support	IPG – 30%

Feature Vector

We use the gait index signal of the Lokomat as an indicator to identify progression through the gait cycle. The gait index is a sawtooth signal and is displayed in the bottom plot in Fig. P3-5. It is used to determine the time-windows of the four phases, cf., the dashed lines in Fig. P3-5. The time-windows are used to compute the feature vector, composed of statistical features for power, angle, and torque for both hip and knee joints, cf., Table P3-3. The result is one feature vector for each phase: $\boldsymbol{x}_{\mathrm{HS}}, \boldsymbol{x}_{\mathrm{MS}}, \boldsymbol{x}_{\mathrm{TO}}, \boldsymbol{x}_{\mathrm{SW}} \in \mathbb{R}^{12}$. The Lokomat provides measurements of all the signals listed in Table P3-3 synchronized by the gait index signal, which makes the computation of the features simple.

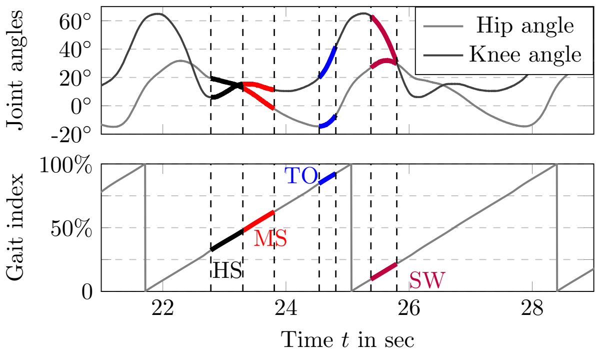


Figure P3-5. Top: Joint angles. Bottom: Segmentation of time signals into four phases using the gait index with HS in 34.5%-47.5%, MS in 47.5%-65.5%, TO in 84.5%-92.5%, and SW in 9.5%-21.5% of one period of the gait index. The falling edge of the gait index does not align with the biomechanical definition of a gait cycle but enables separation of the gait cycle into phases.

Remark 5. *For each subject, the data collection takes around one hour, including rating the gait cycle. As described in Algorithm P3-1, the application of the control law does not include further training and the control law is therefore not personalized to the subject.*

Remark 6. *Initially, we defined more features than the twelve in Table P3-3, e.g., frequency domain features, which the supervised learning problem in* (P3-5) *with L1 regularization discarded. In order to reduce the problem dimension in the online algorithm, we discarded them as well.*

4.3 Reward Model and Steady-State Model for Lokomat

Given the data set, we apply the method in Section 3 to learn four reward models. We obtain a reward model for each of the four phases represented as $\boldsymbol{W}_j$, $\boldsymbol{w}_j$, and b_j from solving (P3-5), where $j \in \{\mathrm{HS}, \mathrm{MS}, \mathrm{TO}, \mathrm{SW}\}$.

Table P3-3. Values for Feature Vector

#	Joint	Signal	Unit	Feature
x^1	hip	joint power	Nm/s	mean
x^2	hip	angle	rad	min
x^3	hip	angle	rad	max
x^4	hip	angle	rad	range
x^5	hip	torque	Nm	mean
x^6	hip	torque	Nm	variance
x^7	knee	joint power	Nm/s	mean
x^8	knee	angle	rad	min
x^9	knee	angle	rad	max
x^{10}	knee	angle	rad	range
x^{11}	knee	torque	Nm	mean
x^{12}	knee	torque	Nm	variance

The steady-state model $\boldsymbol{M} \in \mathbb{R}^{7\times 48}$, $\boldsymbol{m} \in \mathbb{R}^7$ in (P3-7) is computed by stacking the features of the four phases:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\mathrm{HS}}^{\mathsf{T}} & \boldsymbol{x}_{\mathrm{MS}}^{\mathsf{T}} & \boldsymbol{x}_{\mathrm{TO}}^{\mathsf{T}} & \boldsymbol{x}_{\mathrm{SW}}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}.$$

4.4 Control Law for Gait Rehabilitation Robot

Once the four reward models and the steady-state model are trained using the data in (P3-2), the controller automatically chooses input setting adaptations given the current measurements, i.e., it does not require ratings from therapists. The input adaptation $\Delta\boldsymbol{s}$ is computed as

$$\Delta\boldsymbol{s} = \boldsymbol{M} \begin{bmatrix} \alpha(\boldsymbol{W}_{\mathrm{HS}}\boldsymbol{x}_{\mathrm{HS}} + \boldsymbol{w}_{\mathrm{HS}}) + \boldsymbol{x}_{\mathrm{HS}} \\ \alpha(\boldsymbol{W}_{\mathrm{MS}}\boldsymbol{x}_{\mathrm{MS}} + \boldsymbol{w}_{\mathrm{MS}}) + \boldsymbol{x}_{\mathrm{MS}} \\ \alpha(\boldsymbol{W}_{\mathrm{TO}}\boldsymbol{x}_{\mathrm{TO}} + \boldsymbol{w}_{\mathrm{TO}}) + \boldsymbol{x}_{\mathrm{TO}} \\ \alpha(\boldsymbol{W}_{\mathrm{SW}}\boldsymbol{x}_{\mathrm{SW}} + \boldsymbol{w}_{\mathrm{SW}}) + \boldsymbol{x}_{\mathrm{SW}} \end{bmatrix} + \boldsymbol{m} - \boldsymbol{s}.$$

While $\Delta\boldsymbol{s}$ yields continuous values, the input settings are adjusted in discrete steps, cf., the step-sizes in Table P3-1. We aim to change one setting at a time, which is common practice for therapists [7] and eases the evaluation. The following suggests a method to select one single adaptation from $\Delta\boldsymbol{s}$.

Input Setting Selection & Stopping Criterion

In order to select one single discrete change in input setting, we normalize $\Delta \boldsymbol{s}$ to account for different value ranges and different units per individual setting and select the input corresponding to the largest in absolute value:

$$k^\star = \arg \max_{k=1,\ldots,7} \left| \frac{\Delta s^k}{\bar{s}^k - \underline{s}^k} \right|$$

with associated index $k^\star$, where the normalization $\bar{s}^k - \underline{s}^k$ is the range of the input setting k in Table P3-1. Hence, the algorithm chooses one adaptation with step-size in Table P3-1. The input adaptation is stopped when the largest normalized absolute value of change is smaller than a pre-defined parameter β, i.e., $\left| \Delta s^{k^\star} / (\bar{s}^{k^\star} - \underline{s}^{k^\star}) \right| \leq \beta$. This indicates closeness to the optimum, i.e., that a physiological gait is reached.

5 Model Evaluation in Simulation

We first analyze the algorithm in simulation to investigate the model quality. In this simulation study, we compare two reward models: One that uses ratings on an integer scale from 1 to 3 ($S = 3$ in (P3-5)) and one that uses only binary ratings, i.e., good and bad ($S = 2$ in (P3-5)). For the case $S = 3$, we use the collected ratings without modification. For the case $S = 2$, we combine the data points with rating 1 and rating 2 as *samples of a bad gait* and use the data points with rating 3 as *samples of a good gait.*

5.1 Evaluation Metrics and Results

In order to evaluate the trained models, we split the experimentally collected data into training (80%) and validation data (20%). This split is done randomly and repeated 500 times to assess the robustness of the models. This technique is known as 5-fold cross validation [50] and ensures that the validation data are not biased by training on the same data.

Evaluation of Reward Model

We evaluate the accuracy of the reward model trained with three ratings, i.e., $S = 3$, by computing the pairwise difference in estimated rewards $r(\boldsymbol{x}_i) - r(\boldsymbol{x}_j)$ for two data samples i and j, classified with respect to their ratings r_i and r_j. The metric is motivated by the fact that two different ratings should be distinguishable. We define $\Delta\bar{r}_{nm}$ as

$$\Delta\bar{r}_{nm} = \frac{1}{|I_n||I_m|} \sum_{i \in I_n} \sum_{j \in I_m} \left(r(\boldsymbol{x}_i) - r(\boldsymbol{x}_j) \right), \tag{P3-8}$$

where $I_n = \{i | r_i = n\}$ is an index set of data points with ratings $r_i = n$. If the trained reward model and data were perfect, $\Delta\bar{r}_{nm} = n - m$ with zero standard deviation.

We evaluate the accuracy of the reward model trained with binary ratings, i.e., $S = 2$, by computing the classification accuracy of good versus bad ratings:

$$\bar{p}_{\text{good/bad}} = \frac{1}{|I_{\text{good}}||I_{\text{bad}}|} \sum_{i \in I_{\text{good}}} \sum_{j \in I_{\text{bad}}} \mathcal{I}_{r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)} \tag{P3-9}$$

with $\mathcal{I}_{r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)} = 1$ if $r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)$ and $\mathcal{I}_{r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)} = 0$, otherwise, and $I_{\text{good}} = I_3$ and $I_{\text{bad}} = I_1 \cup I_2 = \{i | r_i = 1 \text{ or } r_i = 2\}$.

Fig. P3-6 reports statistical values of both $\Delta\bar{r}_{nm}$ in (P3-8) for evaluating the reward model with $S = 3$ and $\bar{p}_{\text{good/bad}}$ in (P3-9) for evaluating the reward model with $S = 2$ over the 500 splits of training and validation data. For the reward model computed with $S = 3$, the overall deltas in estimated rewards match the deltas in ratings very closely with 2.00 for $\Delta\bar{r}_{31}$, 0.97 for $\Delta\bar{r}_{32}$, and 1.04 for $\Delta\bar{r}_{21}$. For the reward model computed with $S = 2$, the overall classification accuracy ($r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)$ if $r_i > r_j$) is 92.5%.

Evaluation of Steady-State Model

The steady-state model is evaluated using the prediction error $\bar{e}^k$ defined as

$$\bar{e}^k = \frac{1}{N} \sum_{i=1}^{N} |s_i^k - \boldsymbol{M}_{k\star} \boldsymbol{x}_i - \boldsymbol{m}_k|, \tag{P3-10}$$

where k is the index of the input setting and $\boldsymbol{M}_{k\star}$ is the kth row of matrix $\boldsymbol{M}$ and $\boldsymbol{m}_k$ is the kth entry of vector $\boldsymbol{m}$. As we use normalized values for the input settings with $s_i^k \in [0, 1]$, the error $\bar{e}^k$ can be interpreted as a percentage offset from the correct input setting.

Table P3-4 reports the mean and standard deviation of the errors $\bar{e}^k$ in (P3-10) over the 500 random splits of training and validation data for all input settings k. It shows an overall average error of 4.17% and that the errors for all input settings are consistently lower than 6%.

Evaluation of Overall Algorithm

We evaluate the performance of the overall algorithm by comparing the collected data with the output of the algorithm. Let the changes in input settings during data collection for all data samples $i = 1, ...N$ be $\Delta \boldsymbol{s}_i^{\text{ex}} = \boldsymbol{s}^{\text{ex}} - \boldsymbol{s}_i$, where $\boldsymbol{s}_i$ are the input settings of data point i and $\boldsymbol{s}^{\text{ex}}$ are the physiological settings, which are set by the therapist at the beginning of the data collection. Note that $\boldsymbol{s}^{\text{ex}}$ depends on the subject, however, we omit this dependency in the notation for ease of exposition. It is also important to note that $\boldsymbol{s}^{\text{ex}}$ are not the only possible physiological input settings. We compare the input adaptation proposed by our algorithm $\Delta \boldsymbol{s}_i$ against the deviation from the physiological settings $\Delta \boldsymbol{s}_i^{\text{ex}}$, where we can have three different outcomes:

Case 1 (Same Setting & Same Direction)

The algorithm selects the input adaptation in the same direction as during data collection, which is known to be a correct choice as it is closer to the physiological settings $\boldsymbol{s}^{\text{ex}}$.

Case 2 (Same Setting & Opposite Direction)

The algorithm selects the same setting but in the opposite direction as during data collection, which is likely to be an incorrect choice.

Case 3 (Different Setting)

The algorithm selects a different input adaptation, the implications of which are unknown and could be either correct or incorrect, which cannot be evaluated without closed-loop testing.

We compute the percentage of data points falling in each case for each setting k and for $\Delta \boldsymbol{s}_i^{\text{ex}} = 0$ (no adaptation), i.e., p_{C1}^k, p_{C2}^k, and p_{C3}^k for Case 1, Case 2, and Case 3, respectively, where $p_{\text{C1}}^k + p_{\text{C2}}^k + p_{\text{C3}}^k = 1$. If the algorithm replicated the data collection perfectly, then $p_{\text{C1}}^k = 1$ for all settings k. Given the discrete and unique setting selection, the overall algorithm has 15 options to choose from: An increase in one of the seven settings by one unit, a decrease in one of the seven settings by one unit, or *no adaptation*. Hence, random decision-making yields a probability of $p = 1/15 \approx 6.7\%$ for each option.

Table P3-5 reports mean and standard deviation of the percentage values of the three cases. The algorithm chooses the input adaptations for hip range of motion, hip offset, knee range of motion, and knee offset very often when their adaptation leads to $\boldsymbol{s}^{\text{ex}}$ (86.7%–100.0%). Also, it often chooses *no adaptation* when the gait is physiological, with input settings

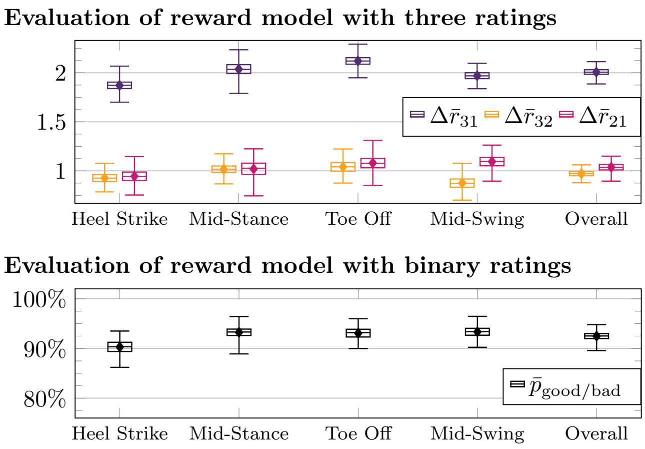


Figure P3-6. Evaluation of reward models for individual phases and overall. The mean over 500 splits of training and validation data, along with the median, 25th and 75th percentiles, and maximum and minimum values are indicated by a diamond, a line, box edges, and whiskers, respectively. Top: Pairwise difference in estimated rewards $\Delta\bar{r}_{31}$, $\Delta\bar{r}_{32}$, and $\Delta\bar{r}_{21}$. Bottom: Classification accuracy $\bar{p}_{\text{good/bad}}$.

Table P3-4. Mean and Standard Deviation of Steady-State Model

s^k	Setting	Error $\bar{e}^k$
s^1	Hip Range of Motion	0.0578 ± 0.0019
s^2	Hip Offset	0.0370 ± 0.0008
s^3	Knee Range of Motion	0.0547 ± 0.0021
s^4	Knee Offset	0.0324 ± 0.0009
s^5	Speed	0.0307 ± 0.0009
s^6	Orthosis Speed	0.0315 ± 0.0010
s^7	Bodyweight Support	0.0542 ± 0.0017
	Overall	0.0417 ± 0.0186

$\boldsymbol{s}^{\text{ex}}$. Table P3-5 also shows that decision-making with the proposed algorithm is more ambiguous for the input adaptations of speed, orthosis speed, and bodyweight support. Overall, the algorithm proposes a setting that is closer to $\boldsymbol{s}^{\text{ex}}$ (Case 1) in 80.7% and 80.6% for the reward models trained with $S = 3$ and $S = 2$, respectively. The algorithm suggests a probably incorrect input adaptation in less than 1% (Case 2). In around 19%, the algorithm suggests a different input adaptation (Case 3).

Table P3-5. Evaluation of Overall Algorithm in Simulation

	Three ratings (1, 2, or 3) $S = 3$			
s^k	Setting	p^k_{C1} in %	p^k_{C2} in %	p^k_{C3} in %
	No Adaptation	77.6 ± 3.6	-	22.4 ± 3.6
s^1	Hip Range of Motion	86.7 ± 1.8	0	13.3 ± 1.8
s^2	Hip Offset	96.4 ± 1.0	0	3.6 ± 1.0
s^3	Knee Range of Motion	91.0 ± 1.7	0	9.1 ± 1.7
s^4	Knee Offset	100.0 ± 0.0	0	0
s^5	Speed	71.1 ± 2.8	0	29.0 ± 2.8
s^6	Orthosis Speed	33.3 ± 3.7	3.5 ± 1.6	63.2 ± 3.8
s^7	Bodyweight Support	55.6 ± 3.8	0	44.5 ± 3.8
	Overall accuracy	80.7 ± 1.0	0.3 ± 0.1	19.0 ± 1.0

	Binary ratings (good or bad) $S = 2$			
s^k	Setting	p^k_{C1} in %	p^k_{C2} in %	p^k_{C3} in %
	No Adaptation	76.8 ± 3.7	-	23.2 ± 3.7
s^1	Hip Range of Motion	88.6 ± 1.7	0	11.4 ± 1.7
s^2	Hip Offset	95.6 ± 1.1	0	4.4 ± 1.1
s^3	Knee Range of Motion	90.0 ± 2.0	0	10.0 ± 2.0
s^4	Knee Offset	100.0 ± 0.0	0	0
s^5	Speed	71.3 ± 2.9	0	28.7 ± 2.9
s^6	Orthosis Speed	32.1 ± 3.8	2.9 ± 1.4	65.0 ± 3.9
s^7	Bodyweight Support	53.9 ± 3.9	0	46.2 ± 3.9
	Overall accuracy	80.6 ± 1.1	0.2 ± 0.1	19.2 ± 1.0

Remark 7. *The same evaluation using exclusively kinematic features (x^2, x^3, x^4, x^8, x^9, x^{10} in Table P3-3) yields slightly different results with overall $p_{C1} = 79.1\% \pm 1.0\%$, $p_{C2} = 0.2\% \pm 0.1\%$, and $p_{C3} = 20.6\% \pm 1.0\%$ for binary ratings. A purely kinematic feature vector might be important when working with impaired patients, where power/torque features might be an indication of individual impairments rather than a characterization of a physiological gait.*

5.2 Discussion

The rewards predicted with the reward model trained with three ratings (two classification boundaries at 1.5 and 2.5) match the true ratings very closely. The reward model trained with binary ratings (one classification boundary at 2.5) is able to distinguish *good* from *bad* gait patterns confidently with an overall classification accuracy of more than 90%. The steady-state model shows an average error of 5%. As we will show in Section 6, this accuracy suffices for the considered application. For example, the expected error of 3.07% of s^5 translates into an error in treadmill speed of 0.075m/s and the expected error of 5.78% of s^1 translates into an error in hip range of motion of 2.08°, which is less than one input setting step-size, cf., Table P3-1. Even though another model may increase accuracy, it may come at the expense of increased complexity in the computation. Our linear model only requires matrix-vector multiplication, which can easily be implemented on the controller of the Lokomat and is chosen as a suitable compromise of simplicity and accuracy.

The evaluation of both components, the reward model and the steady-state model, in simulation allow us to conclude that they provide suitable models for the considered application. For the overall algorithm, Case 1 is known to result in improved physiology of the gait cycle. Case 3, however, does not imply that the suggested adaptation will not lead to an improved gait cycle as there may be multiple different input adaptations that lead to a physiological gait (not only $\boldsymbol{s}^{\text{ex}}$). In these cases, we do not know if the suggested adaptation would have led to an improvement in gait without closed-loop testing. Hence, the probabilities 80.7% and 80.6% of Case 1 for the two reward models can be interpreted as a lower bound for the overall improvement. The relatively low standard deviation for all settings indicates that the learning is robust against variation in the training data. The use of binary ratings eases the data collection and has been shown to perform similarly well. Therefore, we proceed with closed-loop testing of the algorithm using a reward model trained with binary ratings.

6 Experimental Results - Closed-Loop Testing

The proposed algorithm was implemented as a recommendation system on the Lokomat for closed-loop evaluation. We implemented the algorithm using the reward model trained with binary ratings (good and bad) of the gait cycle. It is important to note, that no data from the respective test person was used for training of the reward or the steady-state model.

6.1 Experiment Setup

We conducted experiments with ten nondisabled subjects and ten different sets of initial non-physiological gait cycles (test scenarios). Table P3-6 describes the ten test scenarios and outlines input adaptations that therapists are expected to make (according to the heuristic guidelines). Scenario 1 through 8 are very common observations of a patient's gait cycle on the Lokomat. Scenario 9 and 10 are combinations of the more common flaws and are included to challenge the algorithm with more complex scenarios.

Table P3-6. Test Scenarios of Experiment

Scenario: Observations	Therapists' heuristic rules (expectation)
1: Limited foot clearance, foot dropping	Increase knee range of motion ($s^3 \uparrow$)
2: Short steps	Increase hip range of motion, speed ($s^1 \uparrow$, $s^5 \uparrow$)
3: Foot dragging	Decrease speed, increase orthosis speed ($s^5 \downarrow$, $s^6 \uparrow$)
4: Large steps, late heel strike	Decrease hip range of motion, hip offset ($s^1 \downarrow$, $s^2 \downarrow$)
5: Short steps, hip extension	Increase hip range of motion, hip offset ($s^1 \uparrow$, $s^2 \uparrow$)
6: Bouncing	Decrease speed, bodyweight support ($s^5 \downarrow$, $s^7 \downarrow$)
7: Foot slipping	Decrease knee range of motion, orthosis speed ($s^3 \downarrow$, $s^6 \downarrow$)
8: Knee buckling	Increase knee range of motion, bodyweight support ($s^3 \uparrow$,$s^7 \uparrow$)
9: Large steps, early heel strike	Decrease hip range of motion, increase hip offset, increase speed ($s^1 \downarrow$, $s^2 \uparrow$, $s^5 \uparrow$)
10: Large steps, late heel strike, foot slipping	Decrease hip range of motion, hip offset, knee range of motion, orthosis speed ($s^1 \downarrow$, $s^2 \downarrow$, $s^3 \downarrow$, $s^6 \downarrow$)

The selected scenarios cover the most common observations of the gait cycle of a passive subject (without muscle activity) on the Lokomat and, therefore, are expected to adequately evaluate the proposed algorithm experimentally (with nondisabled subjects). The initial input settings to achieve non-physiological gait patterns (scenarios and observations in Table P3-6) were chosen manually and purposefully by experienced therapists individually for each subject until the respective observation was present. The guidance force was set to 100% for all trials, i.e., the subjects were walked by the Lokomat. The treadmill speed was varied between 1.4km/h and 2.3km/h.

We conducted 63 experimental trials with the proposed algorithm in closed-loop with the ten nondisabled subjects, where each subject underwent at least five trials. The difference in the number of experimental trials is due to each subject's availability. However, the test scenarios were chosen so that each scenario was tested comparably often. Similarly to the data collection, the subjects were instructed to be as passive as possible. Two therapists assessed the input adaptations suggested by the algorithm and rated whether the gait was physiological. The therapists implemented the input adaptations until the algorithm indicated that a physiological gait cycle had been reached. Additionally, the therapists indicated when they thought that a physiological gait had been reached and the algorithm should be stopped.

6.2 Results

Fig. P3-7 illustrates eight representative trials with the first subject. It contains four types of information and is separated by therapist in columns and by test scenario in rows:

i) The gait cycle rating r_{gait} (y-axis), calculated as the sum of the individual phase ratings $r_{\text{gait}} = r_{\text{HS}} + r_{\text{MS}} + r_{\text{TO}} + r_{\text{SW}}$, over the number of input adaptations (x-axis);

ii) the applied input adaptations and their direction, e.g., $s_1 \uparrow$ represents an increase of Setting 1 by one unit;

iii) a statement from the therapists about the algorithm's suggested input adaptation, i.e., agreement with the suggestion as check mark ✓, disagreement as cross ✗, and uncertainty about the suggestion as question mark **?**; and

iv) the reaching of a physiological gait judged by the therapist with square markers ■ (for the usage as recommendation system) and by the algorithm with diamond markers ⧫ (for the usage as automatic adaptation system).

In all eight illustrated experiments, the algorithm provides a reliable, although not monotonic, improvement in the physiology of the gait. The input adaptations suggested by the algorithm led to a physiological gait for both the usage as recommendation system (square marker) and automatic

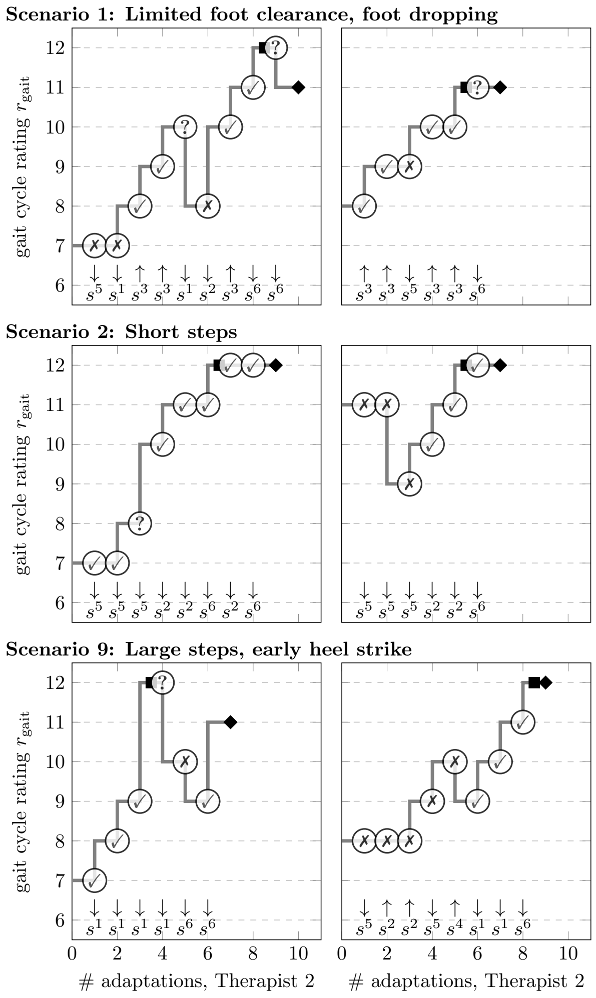


Figure P3-7. Experimental evaluation of the closed-loop recommendation system. Averaged for the eight experiments, a physiological gait was reached after 6.0 input adaptations (until square marker). The diamond marker indicates that the algorithm assessed the gait as physiological (stopping criterion).

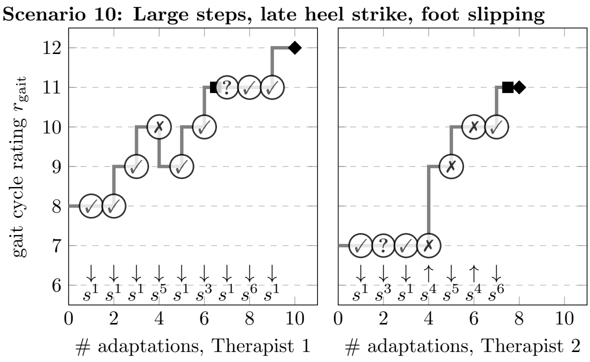


Figure P3-7. (Continued)

adaptation system (diamond marker) in less than 10 adaptations with an overall rating of greater than or equal to 11, where 12 is the maximum possible rating. The input adaptations during the test of Scenario 1 with both therapists (first row) are similar to the heuristic guidelines in Table P3-6, i.e., an increase in the knee range of motion (s^3 ↑). The input adaptations for Scenario 2 (second row) are different from the heuristic guidelines. Here, the algorithm converges to a kinematically different but physiological gait that is achieved through a slower treadmill speed and input settings that are adjusted accordingly. For Scenario 9 and 10 (third and fourth row), the algorithm achieved a physiological gait through adaptations that are similar to the heuristic guidelines. In all illustrated cases, the algorithm converges to a physiological gait.

Table P3-7 summarizes all 63 experimental trials with ten subjects. On average, after a proposed input adaptation, the gait cycle improved in 63% and did not degrade in 93% of adaptations. The latter percentage is important as sometimes, changing an input setting by only one unit is too small to make a noticeable change in the gait cycle and a couple of consecutive adaptations are necessary, e.g., for the orthosis speed (s^6). Most importantly, a physiological gait cycle was reached for all trials within 6.0 $\underline{a}$daptations $\underline{p}$er $\underline{t}$rial (APT) on average for all subjects combined, and between 3.8 and 7.4 for each individual subject.

6.3 Discussion

In general, the algorithm reached a physiological gait cycle within very few adaptations. This is achieved as the proposed algorithm reasons about the gait cycle using the reward model in a higher dimensional feature space

Table P3-7. Summary of all Experimental Trials

				Physiology of gait	
Subject	(body type)	Trials	APT	improved	not degraded
1	(193cm, 93kg, male)	8	6.0	65%	92%
2	(195cm, 100kg, male)	5	3.8	89%	100%
3	(163cm, 53kg, female)	6	6.8	68%	98%
4	(175cm, 85kg, female)	5	4.8	58%	92%
5	(172cm, 68kg, female)	9	4.9	57%	89%
6	(190cm, 85kg, male)	5	7.4	54%	97%
7	(167cm, 85kg, male)	6	7.2	60%	93%
8	(180cm, 75kg, male)	5	7.0	60%	83%
9	(167cm, 64kg, female)	7	5.7	65%	97%
10	(161cm, 48kg, female)	7	6.9	71%	92%
Overall		63	6.0	63%	93%

rather than the space of input settings. As a result, the controller does not rely on trial and error search and, therefore, does not require to be tuned individually for each patient, which makes the approach more efficient, e.g., compared to classical reinforcement learning methods. The majority of times, the therapists agreed with the suggestions from the algorithm, i.e., the suggested adaptations were conform with the heuristic tuning guidelines and their experience. Consequently, the resulting gait cycle was mostly kinematically similar to the one that the therapists would have chosen. In some notable instances, the therapists disagreed or were uncertain about the proposition and were surprised by the improvement in the gait cycle, e.g., Row 1, Therapist 1, Adaptation 6; Row 2, Therapist 1, Adaptation 3; or Row 4, Therapist 2, Adaptation 4 in Fig. P3-7. These instances are examples of situations where the algorithm chooses input adaptations, which were unknown to the therapists. In these cases, the resulting gait cycle was sometimes kinematically different from the heuristic guidelines, e.g., a gait with slower treadmill speed. Table P3-7 shows that the algorithm is able to cope with various body types with similar results for all individuals.

It is worth noting that the differences between similar scenarios with two different therapists in Fig. P3-7 and the same initial input settings do not necessarily lead to the same adjustment of input settings. This observation can be explained as the physiology of the gait does not only depend on the chosen input settings but also on the hardware setup, e.g., the tightness of the straps, which differs slightly between therapists. However, even though the hardware was set up slightly differently by the two therapists, the algorithm managed to find input settings that walk the subject physiologically, indicating certain robustness to slight variations in the hardware.

7 Conclusion and Future Work

This paper has derived a supervised learning-based method utilizing human ratings for learning parameters of a feedback controller. The approach was applied to the Lokomat robotic gait trainer with the goal of automatically adjusting the input settings to reach a physiological gait cycle by encapsulating the therapists' expertise in a reward model. Feedback control was enabled by this reward model and a steady-state model, which allows for converting desired changes in the gait into input adaptations. Experiments with human subjects showed that the therapists' expertise in the form of ratings of four gait phases provides sufficient information to discriminate between physiological and non-physiological gait cycles. Furthermore, the provided adaptations led to an improvement of the gait cycle towards a physiological one within fewer than ten adaptations. The physiological gait cycle was partly reached by changes in input settings that domain experts would not have chosen themselves, suggesting that the proposed method might also be capable of generalizing from ratings and proposing improved settings for unseen scenarios. This observation remains to be confirmed with more data in future work.

Future work involves the data collection, evaluation, and validation of the proposed method with impaired patients. This will include the assessment of asymmetric gait adaptations for the right and left legs, which can readily be achieved by considering one feature vector for each leg. Further, physical limitations and/or constraints in the patients' movements could be assessed online using sensor measurements of the Lokomat and considered for the selection of input settings.

Acknowledgment

We gratefully acknowledge Patricia Andreia Gonçalves Rodrigues, Ursula Costa, and Serena Maggioni for their clinical support and invaluable discussions; Luca Somaini and Nils Reinert for their technical support; and Liliana Pavel for implementing the user interface.

Bibliography

[1] G. Colombo, M. Joerg, R. Schreier, and V. Dietz, "Treadmill training of paraplegic patients using a robotic orthosis", *J. Rehab. Res. and Develop.*, vol. 37, no. 6, pp. 693–700, 2000.

[2] D. J. Reinkensmeyer, J. L. Emken, and S. C. Cramer, "Robotics, motor learning, and neurologic recovery", *Annu. Rev. Biomed. Eng.*, vol. 6, pp. 497–525, 2004.

[3] J. L. Emken and D. J. Reinkensmeyer, "Robot-enhanced motor learning: Accelerating internal model formation during locomotion by transient dynamic amplification", *IEEE Trans. Neural Syst. Rehab. Eng.*, vol. 13, no. 1, pp. 33–39, 2005.

[4] L. Marchal-Crespo and D. J. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury", *J. Neuroeng. and Rehabil.*, vol. 6, no. 1, p. 20, 2009.

[5] O. Lambercy, L. Lünenburger, R. Gassert, and M. Bolliger, "Robots for measurement/clinical assessment", in *Neurorehabil. Technol.* London: Springer, 2012, pp. 443–456.

[6] H. M. Van der Loos, D. J. Reinkensmeyer, and E. Guglielmelli, "Rehabilitation and health care robotics", in *Springer handbook of robotics*, Springer, 2016, pp. 1685–1728.

[7] Hocoma, "Instructor script Lokomat training", Hocoma AG, Tech. Rep., 2015.

[8] S. Jezernik, G. Colombo, and M. Morari, "Automatic gait-pattern adaptation algorithms for rehabilitation with a 4-dof robotic orthosis", *IEEE J. Robot. Automat.*, vol. 20, no. 3, pp. 574–582, 2004.

[9] R. Riener, L. Lünenburger, S. Jezernik, M. Anderschitz, G. Colombo, and V. Dietz, "Patient-cooperative strategies for robot-aided treadmill training: First experimental results", *IEEE Trans. Neural Syst. Rehab. Eng.*, vol. 13, no. 3, pp. 380–394, 2005.

[10] R. Riener, L. Lünenburger, and G. Colombo, "Human-centered robotics applied to gait training and assessment", *J. Rehab. Res. and Develop.*, vol. 43, no. 5, pp. 679–694, 2006.

[11] A. Duschau-Wicke, J. Von Zitzewitz, R. Banz, and R. Riener, "Iterative learning synchronization of robotic rehabilitation tasks", *IEEE Int. Conf. Rehabil. Robot.*, pp. 335–340, 2007.

[12] A. Duschau-Wicke, J. von Zitzewitz, A. Caprez, L. Lünenburger, and R. Riener, “Path control: A method for patient-cooperative robot-aided gait rehabilitation”, *IEEE Trans. Neural Syst. Rehab. Eng.*, vol. 18, no. 1, pp. 38–48, 2010.

[13] S. Maggioni, L. Lünenburger, R. Riener, and A. Melendez-Calderon, “Robot-aided assessment of walking function based on an adaptive algorithm”, *IEEE Int. Conf. Rehabil. Robot.*, pp. 804–809, 2015.

[14] J. L. Emken, R. Benitez, A. Sideris, J. E. Bobrow, and D. J. Reinkensmeyer, “Motor adaptation as a greedy optimization of error and effort”, *J. Neurophysiol.*, vol. 97, no. 6, pp. 3997–4006, 2007.

[15] A. Koenig, X. Omlin, L. Zimmerli, M. Sapa, C. Krewer, M. Bolliger, F. Müller, and R. Riener, “Psychological state estimation from physiological recordings during robot-assisted gait rehabilitation.”, *J. Rehabil. Res. & Develop.*, vol. 48, no. 4, pp. 367–386, 2011.

[16] R. Begg and J. Kamruzzaman, “A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data”, *J. Biomechanics*, vol. 38, no. 3, pp. 401–408, 2005.

[17] J. Wu, J. Wang, and L. Liu, “Feature extraction via KPCA for classification of gait patterns”, *Human Movement Sci.*, vol. 26, no. 3, pp. 393–411, 2007.

[18] M. Yang, H. Zheng, H. Wang, S. McClean, J. Hall, and N. Harris, “A machine learning approach to assessing gait patterns for complex regional pain syndrome”, *Med. Eng. & Physics*, vol. 34, no. 6, pp. 740–746, 2012.

[19] P. Tahafchi, R. Molina, J. A. Roper, K. Sowalsky, C. J. Hass, A. Gunduz, M. S. Okun, and J. W. Judy, “Freezing-of-gait detection using temporal, spatial, and physiological features with a support-vector-machine classifier”, *IEEE Int. Conf. Eng. in Medicine and Biol. Soc.*, pp. 2867–2870, 2017.

[20] R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

[21] W. B. Knox and P. Stone, “Interactively shaping agents via human reinforcement: The TAMER framework”, *5th Int. Conf. Knowledge Capture*, pp. 9–16, 2009.

[22] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, “Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning”, *IEEE Int. Conf. Rehabil. Robot.*, pp. 134–140, 2011.

[23] S. Griffith, K. Subramanian, and J. Scholz, “Policy shaping: Integrating human feedback with reinforcement learning”, *Adv. Neural Inform. Process. Syst.*, pp. 2625–2633, 2013.

[24] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences”, in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 4302–4310.

[25] N. Wilde, D. Kulić, and S. L. Smith, “Learning user preferences in robot motion planning through interaction”, in *IEEE Int. Conf. Robot. and Automat.*, 2018, pp. 619–626.

[26] A. D. D. Dorsa Sadigh, S. Sastry, and S. A. Seshia, “Active preference-based learning of reward functions”, in *Robot.: Sci. and Syst.*, 2017.

[27] C. Basu, M. Singhal, and A. D. Dragan, “Learning from richer human guidance: Augmenting comparison-based learning with feature queries”, in *ACM/IEEE Int. Conf. Human-Robot Interaction*, 2018, pp. 132–140.

[28] M. Menner and M. N. Zeilinger, “A user comfort model and index policy for personalizing discrete controller decisions”, in *Eur. Control Conf.*, 2018, pp. 1759–1765.

[29] R. E. Kalman, “When is a linear control system optimal?”, *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[30] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning”, in *17th Int. Conf. Mach. Learning*, 2000, pp. 663–670.

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control”, *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[32] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration”, *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[33] K. Mombaur, A. Truong, and J.-P. Laumond, “From human to humanoid locomotion—an inverse optimal control approach”, *Auton. Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[34] D. Clever, R. M. Schemschat, M. L. Felis, and K. Mombaur, “Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground”, in *6th IEEE Int. Conf. Biomed. Robot. and Biomechatronics*, 2016, pp. 1192–1199.

[35] A. L. E. N. Kleesattel and K. Mombaur, “Inverse optimal control based enhancement of sprinting motion analysis with and without running-specific prostheses”, in *7th IEEE Int. Conf. Biomed. Robot. and Biomechatronics*, 2018, pp. 556–562.

[36] M. Menner, P. Worsnop, and M. N. Zeilinger, “Constrained inverse optimal control with application to a human manipulation task”, *IEEE Trans. Control Syst. Technol.*, 2019. DOI: 10.1109/TCST.2019.2955663.

[37] J. F.-S. Lin, V. Bonnet, A. M. Panchea, N. Ramdani, G. Venture, and D. Kulić, “Human motion segmentation using cost weights recovered from inverse optimal control”, in *IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 1107–1113.

[38] P. Englert, N. A. Vien, and M. Toussaint, “Inverse KKT: Learning cost functions of manipulation tasks from demonstrations”, *Int. J. Robot. Res.*, vol. 36, no. 13–14, pp. 1474–1488, 2017.

[39] M. Menner, K. Berntorp, M. N. Zeilinger, and S. Di Cairano, “Inverse learning for human-adaptive motion planning”, in *58th IEEE Conf. Decision and Control*, 2019, pp. 809–815.

[40] M. Menner and M. N. Zeilinger, “Convex formulations and algebraic solutions for linear quadratic inverse optimal control problems”, in *Eur. Control Conf.*, 2018, pp. 2107–2112.

[41] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning”, in *21st Int. Conf. Mach. Learning*, 2004, p. 1.

[42] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning”, in *23rd AAAI Conf. Artif. Intell.*, vol. 8, 2008, pp. 1433–1438.

[43] S. Levine, Z. Popovic, and V. Koltun, “Nonlinear inverse reinforcement learning with Gaussian processes”, in *Adv. Neural Inform. Process. Syst.*, 2011, pp. 19–27.

[44] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, “Cooperative inverse reinforcement learning”, in *Adv. Neural Inform. Process. Syst.*, 2016, pp. 3909–3917.

[45] K. Bogert, J. F.-S. Lin, P. Doshi, and D. Kulić, “Expectation-maximization for inverse reinforcement learning with hidden data”, in *Int. Conf. Auton. Agents & Multiagent Syst.*, 2016, pp. 1034–1042.

[46] V. Joukov and D. Kulić, “Gaussian process based model predictive controller for imitation learning”, in *IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 850–855.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning.* Secaucus, NJ: Springer, 2006.

[48] R. Riener, L. Lünenburger, I. C. Maier, G. Colombo, and V. Dietz, “Locomotor training in subjects with sensori-motor deficits: An overview of the robotic gait orthosis lokomat”, *J. Healthcare Eng.*, vol. 1, no. 2, pp. 197–216, 2010.

[49] J. Perry, *Gait Analysis: Normal and Pathological Function.* Thorofare, NJ: SLACK Incorporated Inc., 1992.

[50] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* New York: Springer, 2013, vol. 112.

Part C.

Personalizing Self-Driving Cars using Statistical Estimation

Paper P4

Inverse Learning for Data-driven Calibration of Model-based Statistical Path Planning

Marcel Menner, Karl Berntorp, Melanie N. Zeilinger, and Stefano Di Cairano

Published in *IEEE Transactions on Intelligent Vehicles*

Abstract: This paper presents a method for inverse learning of a control objective defined in terms of requirements and their joint probability distribution from data. The probability distribution characterizes tolerated deviations from the deterministic requirements and is learned using maximum likelihood estimation from data. Further, this paper introduces both parametrized requirements for motion planning in autonomous driving applications and methods for the estimation of their parameters from driving data. Both the parametrized requirements and their joint probability distributions are estimated using a posterior distribution such that the control objective is personalized from a prior as driver data are accumulated. Finally, three variants of the learning method are presented that vary in computational complexity and data storage requirements. Key advantages of the proposed inverse learning method are a relatively low computational complexity, a need for a limited amount of data, and that the data do not have to be segmented into specific maneuvers, which makes the method easily implementable. Learning results using data of five human drivers in a simulation environment suggest that the proposed model for human-conscious driving along with the proposed learning method enable a more natural and personalized driving style of autonomous vehicles for their human passengers.

1 Introduction

Humans cognitive abilities to operate a dynamical system are often difficult to match by an autonomous control design. One reason is that it is difficult to analytically model human knowledge and desires or to incorporate human intent into a control objective. Autonomous driving is one example where humans' capabilities of real-time decision making and trading-off various objectives are hard to achieve by pure model-based control approaches [1], [2]. Calibrating a control objective to achieve human-like behavior of a system as complex as an autonomous vehicle can be a challenging, time-consuming, and expensive task. On the other hand, a purely data-driven approach may require a large amount of data that cover all driving conditions, which is again hard and expensive to gather. As such, the best option appears to be a sensible synthesis of model-based and data-driven approaches using the former to define models and objective classes based on well-established problem knowledge, and the latter to define their relative importance based on drivers' behaviors. Inverse learning methods offer an attractive design paradigm to systematically calibrate a control law or control objective of a model-based approach using data.

The motivation for this work is to automate the calibration of autonomous vehicles to achieve a more natural and personalized driving style for the individual human passenger, while retaining safety and behavioral guarantees of model-based motion-planning algorithms. The desire to personalize autonomous vehicles originates from the fact that the feeling of comfort and cautiousness in traffic varies between individual human passengers. In this paper, we define a driving style as the individual preferences in operating a vehicle as a trade-off between potentially conflicting objectives, such as the time to reach the destination, comfort, and cautiousness. Related definitions are presented in [3], [4], and a thorough review of driving styles related to road safety is presented in [5]. In the context of this paper, we aim at finding the motion-planner parameter tuning that minimizes—in some appropriate sense—the difference between the behavior of the autonomously driven and the manually driven vehicle with respect to some key performance indicators, which we refer to as driving requirements.

2 Key Contributions

In this paper, we propose a method to learn a control objective, which consists of parametrized deterministic requirements and a probability distribution. The deterministic requirements represent goals that a dynamical system aims to satisfy, whereas the probability distribution represents tolerated deviations from the requirements accounting for uncertainties and noise, or that the requirements may not be perfectly achieved. The considered control objective for decision-making has been proposed in [6], [7], where a particle filter extracts the motion plan for autonomous driving from given requirements and their joint probability distribution. This paper considers the inverse problem, where motion plans are generated by a different actor, e.g., a human, who demonstrates how to operate the dynamical system. In the context of the motion-planning application, a human driver demonstrates their preferred driving style by taking full control of the vehicle, which will be possible for autonomous vehicles with autonomy levels until at least SAE Level 4 [8]. Indeed, the learning approach is aimed at improving the experience of riding the autonomous vehicle, while the correct system operation is guaranteed before this calibration step by the motion-planning algorithm.

This paper extends our initial investigation [9] with the following: It i) incorporates a prior for the motion planner's parameters in order to gradually personalize the driving style as data are accumulated; ii) details the algorithm for estimating the parameters of the control objective; and iii) proposes and analyzes three variants of the learning method that vary in computational complexity and storage requirements.

More specifically, we propose a parametrized requirement function to be used for personalized motion planning, along with methods to learn its parameters, where we use four requirements: to stay close to the centerline, to track a nominal velocity, to limit the longitudinal and lateral accelerations for comfort, and to maintain a safety distance from obstacle vehicles. Hence, we personalize the motion planner with respect to these four requirements but additional requirements can be added, similarly. Further, we propose a regularized likelihood maximization method to estimate the probability distribution from demonstrated motion plans, which we model to be the result of an estimation problem in a Kalman-filter framework. The regularized likelihood maximization method can be interpreted as maximum a posteriori estimation, where the prior is given by a common belief and a structural belief. The common belief is used to incorporate commonly used parameters and the structural belief is used to favor structurally beneficial parameters for the motion planner. Simulations with five human drivers suggest that both the probability distribution and the parameters of the requirement function are individual, thereby allowing for tailoring the motion planner to individual preferences. Although we validate the proposed learning method using the particle filter algorithm in [7], the method is generally applicable to calibrate control strategies having a well-defined requirement function, which is the case for most optimization-based control strategies.

The first key advantage of the proposed method is the low computational complexity, where we propose three variants making the method adjustable to the available hardware. For example, one variant only requires a few parameters to be updated recursively and its computational complexity is independent of the data size. The second key advantage is that our method is not maneuver based, i.e., the data need not be segmented prior to learning. These two key results make the algorithm easily implementable on hardware suitable for automotive applications.

Gray-box learning philosophy

While a pure end-to-end learning approach results in a black-box algorithm that is difficult to assess and verify, a purely model-based approach may be easier to assess and verify but is difficult to calibrate for achieving a personalized driving style. Here, we pursue a gray-box learning approach and calibrate the model-based motion-planner in [7] using data, i.e., motion plans generated by human drivers. In this way, the overall driving behavior is guaranteed to have the general properties and guarantees of the motion planner [7], including collision avoidance and specific behaviors in safety-critical decisions. Yet, among the admissible motion plans, the ones that are closer to the driver's natural behavior and that enhance passenger comfort are chosen. Thus, our objective is not to imitate the human's individual driving style but to use the motion plans demonstrated by the individual driver to calibrate the parameters of the motion planner such that the autonomous vehicle behavior is as close as possible—in an appropriate sense—to that of the human driver. Since we learn the parameters of a given algorithm, rather than the algorithm itself as in black-box learning, the autonomous vehicle behavior will still satisfy the invariant motion-planner constraints, e.g., the safety constraints. As a consequence, the calibrated motion planner will try to be as close as possible to the driver behavior, within the space of admissible, i.e., safe, behaviors. In particular, the resulting motion planner will not imitate negative behaviors such as unsafe maneuvers or violations of the rules of the road, e.g., exceeding the speed limit. Due to the more limited scope of learning, a reduced amount of data is sufficient for the learning algorithm to operate, since we are learning parameters, rather than the entire algorithm. Fig. P4-1 illustrates the core components of the approach, i.e., the controller, the data generation, the learning algorithm, the validation of the estimated parameters, and the motion planner.

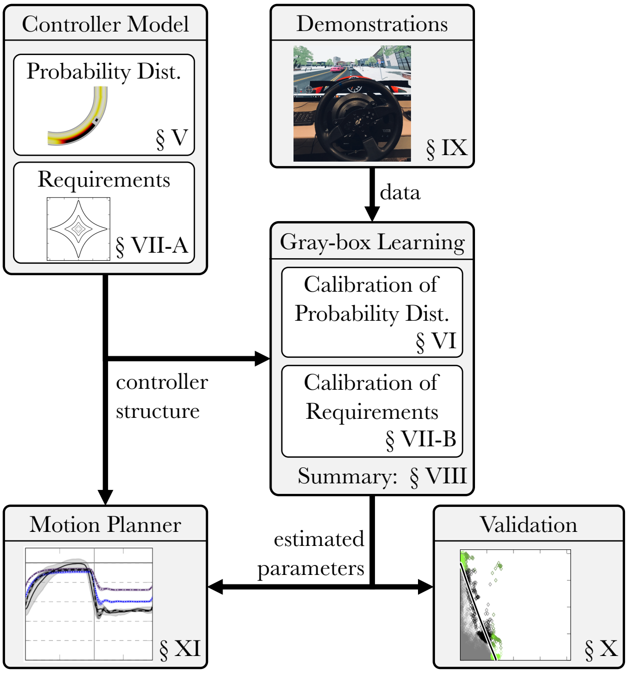


Figure P4-1. Core components, their connections in the learning procedure, and their allocation in the paper. The data provided by human drivers are used to calibrate the parameters of the controller. Due to the gray-box learning approach, the estimated parameters are interpretable and thus, can be validated prior to their employment in the motion planner.

3 Qualitative Comparison with Related Work

Some recent research directions present interesting relations with the techniques proposed here and are worth some discussions. Recent reviews provide additional background on learning objectives from demonstrations [10], [11].

3.1 Inverse Optimal Control (IOC)

IOC methods model observed data to be the result of an optimal control problem [12]–[23] and often aim at transferring human expertise to an autonomous system, e.g., for humanoid locomotion [14], [15], identifying human movements [16]–[18], or robot manipulation tasks [19]. Typically, IOC methods such as [18], [19] use the Karush-Kuhn-Tucker (KKT) conditions and assume a deterministic control objective and the resulting control actions are deterministic. Under this assumption, the performance may deteriorate in the presence of imperfect information such as noisy or suboptimal data. Some notable exceptions are [20], where a bi-level optimization approach is proposed to address imperfect data; [21], where a risk-metric model is introduced to circumvent risk-neutral, deterministic objective functions; [22], where policies are constructed for scenarios with multiple future outcomes; and [23], where the concept of active learning is incorporated into the risk-sensitive framework in [21] enabling an agent to query demonstrations from an expert.

Distinction from IOC

The systematic difference between our method and IOC [12]–[23] lies in the stochasticity of the control objective, which models decision-making explicitly as nondeterministic and suboptimal. In the context of autonomous driving, this model is especially relevant due to the presence of uncertainty in the environment, modeling errors, and sensing and localization errors. The advantage of KKT-based inverse learning paradigms is the consideration of operational constraints, which also facilitates the learning of constraints, e.g., in [18], [24]. However, these methods require the data to be segmented into specific maneuvers in which the assumptions on the data—expressed by the KKT conditions—are satisfied. Automatic segmentation of demonstrations into maneuvers or subtasks is addressed, e.g., in [25]–[27], and the reader is referred to [10] for a detailed discussion. In contrast, for our method, the data need not be segmented into maneuvers prior to learning. On the other hand, while the motion planner is subject to constraints, they are not explicitly considered in the learning procedure. However, for the motion-planning application, this does not pose an issue as operational constraints neither should be changed by the learning algorithm to ensure safety, e.g., the road boundary, nor are actually reached in normal driving conditions, e.g., the peak lateral and longitudinal accelerations of the autonomous vehicle.

3.2 Inverse Reinforcement Learning (IRL)

IRL methods [28]–[37] also learn an objective function from demonstrations and model data to be the result of probabilistic decision-making in a Markov decision process. IRL typically considers a finite state and action space and can be formulated either as a model-based or a model-free approach, depending on prior knowledge. In model-based IRL, which is the closest to the approach considered here, the transition probabilities between states are assumed to be known. A notable parallel development to our work using maximum entropy IRL with a finite state and action space is [37], where reward functions are tuned based on human driving data. In order to overcome the limitation of finite discrete actions, [37] uses high-resolution sampling of time-continuous actions leading to a high-dimensional state-space representation.

Distinction from IRL

The main difference between the proposed method and IRL [28]–[37] is in the formulation of the control problem using a continuous state space in a Kalman filter framework rather than a Markov decision process with a potentially finite state and action space. This difference in the control problem yields a systematic difference in the corresponding inverse problem. For example, [37] uses a high-dimensional state space representation to make IRL applicable to learning driving behaviors, whereas we choose a low-dimensional, continuous state space. Furthermore, the amount of data often tends to be proportional to the size of the space, and hence high-dimensional state spaces may require a considerable amount of time for the learning process to succeed.

3.3 Imitation Learning & Supervised Learning

Further related—but conceptually different—approaches are to learn policies rather than an objective function [38], [39], which is often referred to as imitation learning, or to use labeled data in order to learn an objective function, e.g., using supervised learning [40]–[42]. Notably, [40] uses semi-supervised learning with a similar motivation, where drivers are classified into aggressive and normal driving styles based on a few labeled data points.

Distinction from imitation learning & supervised learning

Compared to policy-based imitation learning methods [38], [39], which tend to replicate the demonstrated motion plans, we learn parameters of an algorithm to obtain the closest behavior among the allowed ones. Therefore, we are more constrained in the solution but need less data and have properties that are invariant through the learning process, which we can use to enforce safe behaviors while learning. Compared to supervised learning methods [40]–[42], we do not require labeled data in order to learn a control objective. On the other hand, inverse learning methods that use unlabeled data, such as IOC, IRL, and our method, require the assumption that the data represent desirable behavior.

4 Notation & Preliminaries

$p(\boldsymbol{x}_{0:T}|\boldsymbol{y}) := p(\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_T|\boldsymbol{y})$ denotes the conditional probability density function (PDF) of $\boldsymbol{x}_k \in \mathbb{R}^n$ at time-steps $k = 0, ..., T$, conditioned on $\boldsymbol{y}$. Given mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ stand for the Gaussian distribution and PDF, respectively, with

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\tfrac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)^\mathsf{T} \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)\right).$$

The notation $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ means $\boldsymbol{x}$ sampled from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the expected value of x is $\mathbb{E}[x]$, and $\propto$ reads proportional to. For a matrix $\boldsymbol{Z} \in \mathbb{R}^{n\times m}$, $z_{ij} = (\boldsymbol{Z})_{ij}$ is the element in the ith row and jth column and vec is the vectorization operator with $\mathrm{vec}(\boldsymbol{Z}) = [z_{11}...z_{n1}, z_{12}...z_{n2}, ..., z_{1m}...z_{nm}]^\mathsf{T}$. $\boldsymbol{D} = \mathrm{diag}(x_1, ...x_n)$ is a diagonal matrix with $d_{ii} = x_i$. We define $\mathcal{Y} = \mathrm{setmax}_{\mathrm{norm}}(\mathcal{X}, M)$ and $\mathcal{Y} = \mathrm{setmin}_{\mathrm{norm}}(\mathcal{X}, M)$ as operators that return the set $\mathcal{Y}$ containing the M largest/smallest elements in the set $\mathcal{X}$ with respect to a given norm; $x = \mathrm{med}(\mathcal{X})$ and $y = \max(\mathcal{X})$ as the median and the maximum value of the scalar elements in $\mathcal{X}$; and $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}} = \boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$.

Gradient of a PDF

We use Einstein's summation convention [43] for conciseness, e.g., for $\boldsymbol{A} \in \mathbb{R}^{n_1\times n_3}$, $\boldsymbol{B} \in \mathbb{R}^{n_1\times n_2}$, $\boldsymbol{C} \in \mathbb{R}^{n_2\times n_3}$, $\boldsymbol{A} = \boldsymbol{BC}$ can be expressed as

$$(\boldsymbol{A})_{ab} = (\boldsymbol{B})_{az}(\boldsymbol{C})_{zb} \Leftrightarrow (\boldsymbol{A})_{ab} = \sum_{z=1}^{n_2}(\boldsymbol{B})_{az}(\boldsymbol{C})_{zb}\ .$$

Let $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ be a function of $\boldsymbol{Y} \in \mathbb{R}^{m\times m}$ and $p(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{\Sigma})$ be a Gaussian PDF. Then,

$$\left(\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{\Sigma})}{\partial \boldsymbol{Y}}\right)_{ab} = \left(\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}^{-1}}\right)_{zy}\left(\frac{\partial \boldsymbol{\Sigma}^{-1}}{\partial \boldsymbol{Y}}\right)_{zyab}$$

and, using $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x} = \mathrm{trace}(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\boldsymbol{x}^\mathsf{T})$,

$$\begin{aligned}\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}^{-1}} &= \frac{\partial(\frac{1}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2}\mathrm{trace}(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\boldsymbol{x}^\mathsf{T}))}{\partial \boldsymbol{\Sigma}^{-1}} \\ &= \frac{1}{2}\boldsymbol{\Sigma} - \frac{1}{2}\boldsymbol{x}\boldsymbol{x}^\mathsf{T},\end{aligned}$$

cf., (57) and (101) in [44]. For $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_k)$ with $k = 1, ..., T$ and time-varying $\boldsymbol{\Sigma}_k$,

$$\left(\frac{\partial \log \prod_{k=1}^{T} p(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{Y}}\right)_{ab} = \sum_{k=1}^{T} \left(\frac{\partial \log p(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{Y}}\right)_{ab}$$

$$= \sum_{k=1}^{T} \left(\frac{1}{2}\boldsymbol{\Sigma}_k - \frac{1}{2}\boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}}\right)_{zy} \left(\frac{\partial \boldsymbol{\Sigma}_k^{-1}}{\partial \boldsymbol{Y}}\right)_{zyab}. \tag{P4-1}$$

Let $\hat{\boldsymbol{\Sigma}}_T = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}}$ be the sample covariance. If $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$ (time-invariant) for all k, (P4-1) simplifies to

$$\frac{T}{2}\left(\boldsymbol{\Sigma} - \hat{\boldsymbol{\Sigma}}_T\right)_{zy} \left(\frac{\partial \boldsymbol{\Sigma}^{-1}}{\partial \boldsymbol{Y}}\right)_{zyab}. \tag{P4-2}$$

5 Problem Statement

We consider discrete-time system dynamics of the form

$$\boldsymbol{x}_k = \boldsymbol{f}\left(\boldsymbol{x}_{k-1}\right) + \boldsymbol{g}\left(\boldsymbol{x}_{k-1}\right)\boldsymbol{u}_k, \tag{P4-3}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are in general nonlinear functions, $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the state at time k, and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denotes the input applied from discrete time-step $k-1$ to k. In the context of autonomous driving, (P4-3) represents the motion model of the vehicle. Note that we deviate from the standard literature by using $\boldsymbol{u}_k$ with index k instead of $k-1$ in (P4-3) to ease notation in the following. The behavior of (P4-3) is modeled with respect to requirements $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ with

$$\boldsymbol{y}_k = \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k, \tag{P4-4}$$

where we call $\boldsymbol{h}_{\boldsymbol{\theta}}$ the requirement function and $\boldsymbol{v}_k$ is the slack, with $\boldsymbol{y}_k = \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ if all requirements are obeyed perfectly. Based on our definition in (P4-3), the requirements are a function of the current control $\boldsymbol{u}_k$ and the predicted state achieved by applying that control, $\boldsymbol{x}_k$. This allows us to quantitatively model several key performance indicators as a control objective.

5.1 Motion Planner & Modeling Assumptions

The requirement function $\boldsymbol{h}_{\boldsymbol{\theta}}$ is modeled as deterministic with potentially unknown parameters $\boldsymbol{\theta}$. On the other hand, the tolerated deviations from the requirements, represented by the slack $\boldsymbol{v}_k$, are modeled as probabilistic and, therefore inflict a probability distribution upon the requirements.

Using the probability distribution and the requirement function, the motion planner considered in [7] constructs the state trajectory PDF given the requirements—i.e., $p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T})$ with $\boldsymbol{y}_k$ in (P4-4) from time $k=0$ to $k=T$—and extracts the state trajectory from the PDF. The extracted state trajectory is then used as motion plan. In this context, the motion-planning problem is formulated as a statistical estimation problem, in which the requirements $\boldsymbol{y}_k$ in (P4-4) are treated as sensor measurements and $\boldsymbol{u}_k$ in (P4-3) is the input (process) disturbance. We model the input disturbance in (P4-3) as Gaussian distributed $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ and the slack in (P4-4) as $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$.

Control inputs & motion plan

In order to find the control inputs that result from this model, we use independence of the random variables to write the joint probability recursively,

$$p\left(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T}\right) \propto \prod_{k=1}^{T} p\left(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{x}_{k-1}\right) \tag{P4-5}$$

and, at the first order, $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) \approx \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{D}_k\boldsymbol{u}_k$ with $\boldsymbol{H}_k = \partial\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u}_k)/\partial\boldsymbol{x}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}$, $\boldsymbol{D}_k = \partial\boldsymbol{h}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_k, \boldsymbol{u})/\partial\boldsymbol{u}|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_k}$, $\boldsymbol{G}_k = \boldsymbol{g}(\boldsymbol{x}_{k-1})$,

$$p\left(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{x}_{k-1}\right) \approx \mathcal{N}\left(\hat{\boldsymbol{x}}_k, \boldsymbol{G}_k\boldsymbol{\Sigma}_k\boldsymbol{G}_k^{\mathsf{T}}\right), \tag{P4-6a}$$

where

$$\hat{\boldsymbol{x}}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{G}_k\hat{\boldsymbol{u}}_k \tag{P4-6b}$$
$$\hat{\boldsymbol{u}}_k = \boldsymbol{K}_k\left(\boldsymbol{y}_k - \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{f}\left(\boldsymbol{x}_{k-1}\right), \boldsymbol{0})\right) \tag{P4-6c}$$
$$\boldsymbol{K}_k = \boldsymbol{Q}\left(\boldsymbol{H}_k\boldsymbol{G}_k + \boldsymbol{D}_k\right)^{\mathsf{T}}\boldsymbol{\Gamma}_k^{-1} \tag{P4-6d}$$
$$\boldsymbol{\Gamma}_k = (\boldsymbol{H}_k\boldsymbol{G}_k + \boldsymbol{D}_k)\boldsymbol{Q}\left(\boldsymbol{H}_k\boldsymbol{G}_k + \boldsymbol{D}_k\right)^{\mathsf{T}} + \boldsymbol{R} \tag{P4-6e}$$
$$\boldsymbol{\Sigma}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k(\boldsymbol{H}_k\boldsymbol{G}_k + \boldsymbol{D}_k)\right)\boldsymbol{Q}. \tag{P4-6f}$$

Eq. (P4-6) is derived using the conditional Gaussian distribution of $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ and is similar to a measurement update of an extended Kalman filter, where $\hat{\boldsymbol{x}}_k$ is the optimal state estimate, $\boldsymbol{K}_k$ is the optimal Kalman gain, $\boldsymbol{\Gamma}_k$ is the innovation covariance, and $\boldsymbol{\Sigma}_k$ is the estimate covariance.

The resulting motion plan—i.e., the assumption on the data—is obtained from (P4-6),

$$\boldsymbol{x}_{0:T} \quad \text{with} \quad \boldsymbol{x}_k \sim \mathcal{N}\left(\hat{\boldsymbol{x}}_k, \boldsymbol{G}_k\boldsymbol{\Sigma}_k\boldsymbol{G}_k^{\mathsf{T}}\right), \tag{P4-7}$$

where the control inputs are

$$\boldsymbol{u}_k = \boldsymbol{K}_k \left(\boldsymbol{y}_k - \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{f}\left(\boldsymbol{x}_{k-1}\right), \boldsymbol{0}) \right) + \boldsymbol{\sigma}_k \tag{P4-8}$$

with the gain $\boldsymbol{K}_k$ in (P4-6d) and $\boldsymbol{\sigma}_k \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma}_k\right)$ with $\boldsymbol{\Sigma}_k$ in (P4-6f). The resulting motion plan and the control inputs are therefore entirely specified by the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ along with $\boldsymbol{y}_k$ and $\boldsymbol{h}_{\boldsymbol{\theta}}$.

Remark 1. *In contrast to* (P4-8)*, in [7], a particle filter extracts the motion plans from* (P4-5)*, where the control inputs become*

$$\boldsymbol{u}_k^N = \hat{\boldsymbol{u}}_k + \frac{\sum_{i=1}^N w_{k,i} \boldsymbol{\sigma}_{k,i}}{\sum_{i=1}^N w_{k,i}}$$

with $\boldsymbol{\sigma}_{k,i} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_k)$ and weights $w_{k,i}$ of the N particles computed as

$$w_{k,i} = \left\| \boldsymbol{y}_k - \boldsymbol{h}\left(\boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{g}(\boldsymbol{x}_{k-1})(\hat{\boldsymbol{u}}_k + \boldsymbol{\sigma}_{k,i}) \right) \right\|_{\boldsymbol{\Gamma}_k^{-1}} .$$

For $N \to \infty$, $\boldsymbol{u}_k^N$ yields asymptotically the optimal (deterministic) motion planner, i.e., $\boldsymbol{u}_k^N \to \arg\max_{\boldsymbol{u}_k} p(\boldsymbol{x}_k | \boldsymbol{y}_k, \boldsymbol{x}_{k-1})$. Although the optimal motion planner would be desirable for autonomous systems, it is in general prohibitive due to the limited amount of computational resources on automotive micro-controllers [45]. Furthermore, it is an ill-suited model for learning from (human) data, which naturally are subject to noise and other sources of suboptimalities. The benefits of the model in (P4-8)*—i.e., $\boldsymbol{u}_k \sim \mathcal{N}(\hat{\boldsymbol{u}}_k, \boldsymbol{\Sigma}_k)$—are that nondeterministic decision-making is considered explicitly and its Gaussian distribution facilitates computationally tractable maximum likelihood estimation, which can be implemented on computational platforms suitable for automotive applications.*

5.2 Conceptual Idea & Problem Definition

In this paper, we learn the probability distribution of the requirement function, defined by $\boldsymbol{Q}$ and $\boldsymbol{R}$, as well as the parameters $\boldsymbol{\theta}$. For human passengers who have not demonstrated their individual preferred driving style, the motion planner uses common covariance matrices $\boldsymbol{Q}^c$ and $\boldsymbol{R}^c$ along with common parameters $\boldsymbol{\theta}^c$. Personalization is achieved through adapting the parameters $\boldsymbol{\theta}$ and the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, where $\boldsymbol{\theta}^c$, $\boldsymbol{Q}^c$, and $\boldsymbol{R}^c$ are used as a prior. This conceptual idea is stated formally with the following two problems.

Problem 1. *Given motion model* (P4-3)*, the requirement function* (P4-4) *with known parameters $\boldsymbol{\theta}$, how motion plans are generated* (P4-7)*,* (P4-8)*, and a set $\mathcal{D}$ of human driven vehicle data $\mathcal{D} = \{(\boldsymbol{x}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{x}_T, \boldsymbol{y}_T)\}$, determine $\boldsymbol{Q}$, $\boldsymbol{R}$ in* (P4-8) *that (at least locally) maximize $p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T})$.*

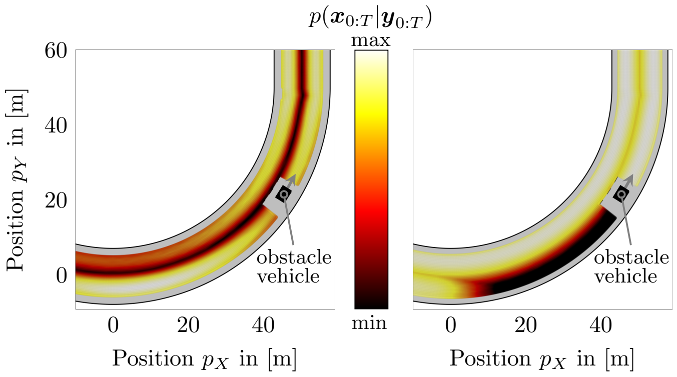


Figure P4-2. Illustration of two driver profiles and their probability distribution. Left: Probability distribution resulting from $\boldsymbol{Q}^c$, $\boldsymbol{R}^c$, and parameters $\boldsymbol{\theta}^c$. Right: Probability distribution from a personalized motion planner. The personalized motion planner is likely to overtake the obstacle vehicle during the turn (low probability of trailing the obstacle vehicle), whereas the motion planner with the common parameters is more likely to stay behind the obstacle vehicle (low probability of changing lanes).

Problem 2. *Given motion model* (P4-3)*, the parametric requirement function* (P4-4)*, the assumption on the motion plans* (P4-7)*,* (P4-8)*, and the data $\mathcal{D}$, determine $\boldsymbol{\theta}$ maximizing $p(\boldsymbol{\theta}|\boldsymbol{\theta}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T})$.*

Problem 1 and Problem 2 are addressed in Section 6 and Section 7, respectively. Section 8 summarizes the learning procedure and presents three variants of the proposed method and their hardware requirements. Fig. P4-2 illustrates the concept of state trajectory PDFs and how decision-making can differ between humans. It displays the probability distributions $p(\boldsymbol{x}_{0:T}|\boldsymbol{y}_{0:T})$ for two realizations of parameters in a traffic scenario with one moving obstacle vehicle, where the color map indicates initial conditions that are more or less likely.

6 Estimation of Covariance Matrices

In this section, we present our method for estimating the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ given the requirement function $\boldsymbol{h}_{\boldsymbol{\theta}}$ and data generated as in (P4-8).

6.1 Optimization Problem Setup

We estimate the covariance matrices from the distribution

$$\begin{aligned} &p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}) \\ &\propto p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{Q}^c, \boldsymbol{R}^c) p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c), \end{aligned} \tag{P4-9}$$

where $p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{Q}^c, \boldsymbol{R}^c) = p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R})$ is the likelihood of the observations and $p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)$ is a prior distribution. The two distributions are further specified in the following. In (P4-9) and what follows, $\boldsymbol{x}_{0:T}$ refers to observations of closed-loop driving and differs from the open-loop motion planner in Section 5.1 being generated by the inputs in (P4-8) rather than by $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$.

Likelihood $p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R})$

Consider the system dynamics (P4-3) and the measurement (requirement) equation (P4-4). If the control inputs are as in (P4-8), then

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}) = \prod_{k=1}^{T} p(\boldsymbol{u}_k | \boldsymbol{0}, \boldsymbol{Q}) p(\boldsymbol{v}_k | \boldsymbol{0}, \boldsymbol{R}) \frac{p(\boldsymbol{w}_k | \boldsymbol{0}, \boldsymbol{R})}{p(\boldsymbol{e}_k | \boldsymbol{0}, \boldsymbol{\Gamma}_k)} \tag{P4-10}$$

with $\boldsymbol{e}_k = \boldsymbol{y}_k - \boldsymbol{h}(\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{0})$, $\boldsymbol{w}_k = \boldsymbol{e}_k - \boldsymbol{J}_k \boldsymbol{u}_k$, and $\boldsymbol{J}_k = \boldsymbol{H}_k \boldsymbol{G}_k + \boldsymbol{D}_k$, which is formally shown in Theorem 1. The reformulation in (P4-10) is essential for the efficiency of the likelihood maximization algorithm detailed in Section 6.2.

Theorem 1. *Consider* (P4-3) *and let* $p(\boldsymbol{x}_0, \boldsymbol{y}_0) = 1$. *If* $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ *in* (P4-4) *and* $\boldsymbol{u}_k$ *as in* (P4-8) *with* $\boldsymbol{K}_k$ *as in* (P4-6d) *and* $\boldsymbol{\sigma}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_k)$, *then* (P4-10) *holds.*

Proof. The joint probability on the left hand side of (P4-10) is equal to the product of the probabilities of all random variables (Markov property),

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}) = \prod_{k=1}^{T} p(\boldsymbol{u}_k | \hat{\boldsymbol{u}}_k, \boldsymbol{\Sigma}_k) p(\boldsymbol{v}_k | \boldsymbol{0}, \boldsymbol{R}). \tag{P4-11}$$

Reformulating (P4-11) using Lemma 1 shows (P4-10). □

Lemma 1. *Let* $\boldsymbol{u}_k$ *as in* (P4-8) *and* $\hat{\boldsymbol{u}}_k = \boldsymbol{K}_k \boldsymbol{e}_k$. *Then,*

$$p(\boldsymbol{u}_k | \hat{\boldsymbol{u}}_k, \boldsymbol{\Sigma}_k) = p(\boldsymbol{u}_k | \boldsymbol{0}, \boldsymbol{Q}) \frac{p(\boldsymbol{w}_k | \boldsymbol{0}, \boldsymbol{R})}{p(\boldsymbol{e}_k | \boldsymbol{0}, \boldsymbol{\Gamma}_k)}.$$

Proof. We need to show that (i):

$$\|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k\|_{\boldsymbol{\Sigma}_k^{-1}} = \|\boldsymbol{u}_k\|_{\boldsymbol{Q}^{-1}} + \|\boldsymbol{w}_k\|_{\boldsymbol{R}^{-1}} - \|\boldsymbol{e}_k\|_{\boldsymbol{\Gamma}_k^{-1}}$$

and (ii): $|\boldsymbol{\Sigma}_k| = |\boldsymbol{Q}|\frac{|\boldsymbol{R}|}{|\boldsymbol{\Gamma}_k|}$.

The identity (i) can be shown by using $\boldsymbol{K}_k^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{K}_k = \boldsymbol{R}^{-1} - \boldsymbol{\Gamma}_k^{-1}$ and $\boldsymbol{\Sigma}_k^{-1}\boldsymbol{K}_k = \boldsymbol{J}_k^\mathsf{T}\boldsymbol{R}^{-1}$ and (ii) follows from:

$$\begin{aligned}|\boldsymbol{\Sigma}_k| &= |\boldsymbol{Q}||\boldsymbol{I}_{n_u} - \boldsymbol{K}_k\boldsymbol{J}_k| = |\boldsymbol{Q}||\boldsymbol{I}_{n_u} - \boldsymbol{Q}\boldsymbol{J}_k^\mathsf{T}\boldsymbol{\Gamma}_k^{-1}\boldsymbol{J}_k| \\ &= |\boldsymbol{Q}||\boldsymbol{I}_{n_y} - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^\mathsf{T}\boldsymbol{\Gamma}_k^{-1}|,\end{aligned}$$

where the last equality is Sylvester's determinant identity [46]. Finally,

$$|\boldsymbol{I}_{n_y} - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^\mathsf{T}\boldsymbol{\Gamma}_k^{-1}| = |\boldsymbol{\Gamma}_k - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^\mathsf{T}||\boldsymbol{\Gamma}_k^{-1}| = |\boldsymbol{R}||\boldsymbol{\Gamma}_k^{-1}|$$

shows (ii). □

Corollary 1. *Let the data be generated by purely stochastic control inputs* $\boldsymbol{u}_k = \boldsymbol{\sigma}_k$. *Then, the sample covariances* $\boldsymbol{Q} = 1/T\sum_{k=1}^{T}\boldsymbol{u}_k\boldsymbol{u}_k^\mathsf{T}$ *and* $\boldsymbol{R} = 1/T\sum_{k=1}^{T}\boldsymbol{v}_k\boldsymbol{v}_k^\mathsf{T}$ *are the maximum likelihood solutions to* $\max_{\boldsymbol{Q},\boldsymbol{R}} p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q},\boldsymbol{R})$.

Proof. This follows directly from the Markov property with $\hat{\boldsymbol{u}}_k = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_k = \boldsymbol{Q}$, for which the sample covariance is the maximum likelihood estimator [47]. □

Remark 2. *Let* $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ *with* $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$ *for* $k = 1, ..., T$. *The probability that the sample covariance* $\hat{\boldsymbol{\Sigma}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{x}_k\boldsymbol{x}_k^\mathsf{T}$ *confidently estimates* $\boldsymbol{\Sigma}$ *is*

$$p\left(\|\boldsymbol{\Sigma} - \hat{\boldsymbol{\Sigma}}_T\|_2 \leq \epsilon\|\boldsymbol{\Sigma}\|_2\right) \geq 1 - \delta, \quad \text{if } T \geq \frac{3n_x^2}{\epsilon\delta}$$

for any $\epsilon > 0$ *and* $\delta \in (0,1)$ *[48], [49]. This confidence bound for the sample covariance holds for purely stochastic control inputs as shown in Corollary 1. For control inputs as in* (P4-8)*, a convergence analysis is more challenging and omitted here as the resulting distribution in* (P4-10) *is not Gaussian.*

Prior belief $p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)$

We use the notion of a prior belief to incorporate both a *common belief* and a *structural belief.* The *common belief* defines deviations from the common covariance matrices $\boldsymbol{Q}^c$ and $\boldsymbol{R}^c$, whereas the *structural belief* favors structurally beneficial covariance matrices. We model $p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)$ as a Gaussian distribution,

$$p\left(\boldsymbol{t}_c\left(\boldsymbol{Q}, \boldsymbol{R}\right) \middle| \boldsymbol{0}, \sigma_c^2 \boldsymbol{I}\right) p\left(\boldsymbol{t}_s\left(\boldsymbol{Q}, \boldsymbol{R}\right) \middle| \boldsymbol{0}, \sigma_s^2 \boldsymbol{I}\right), \tag{P4-12}$$

where the functions $\boldsymbol{t}_c$ and $\boldsymbol{t}_s$ along with the variances σ_c^2 and σ_s^2 are used for the common belief and the structural belief. In this context, σ_c^2 and σ_s^2 trade off prior belief and the likelihood of the observations.

Logarithmic likelihood maximization

Overall, we estimate $\boldsymbol{Q}$ and $\boldsymbol{R}$ by maximizing the log-likelihood,

$$\max_{\boldsymbol{Q}, \boldsymbol{R}} \quad \log\left(p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}) p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)\right) \tag{P4-13a}$$

$$\text{s.t.} \quad \boldsymbol{Q} \in \mathcal{C}_Q, \boldsymbol{R} \in \mathcal{C}_R, \tag{P4-13b}$$

where $\mathcal{C}_Q, \mathcal{C}_R$ can be used to enforce constraints on $\boldsymbol{Q}, \boldsymbol{R}$, e.g., $\boldsymbol{Q} = \boldsymbol{Q}^\mathsf{T} \succeq \boldsymbol{0}, \boldsymbol{R} = \boldsymbol{R}^\mathsf{T} \succeq \boldsymbol{0}$. We optimize (P4-13) with the projected gradient method outlined in Section 6.2.

6.2 Optimization Algorithm: Projected Gradient Descent

Let $\boldsymbol{\xi}^i = [\text{vec}(\boldsymbol{Q}^i)^\mathsf{T} \ \text{vec}(\boldsymbol{R}^i)^\mathsf{T}]^\mathsf{T}$ as the vectorized representation of the covariance matrices at iteration i of the optimization algorithm. Further, let

$$c(\boldsymbol{\xi}^i) = \log\left(p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T} | \boldsymbol{Q}, \boldsymbol{R}) p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)\right) \tag{P4-14}$$

be the log-likelihood as in (P4-13a). Algorithm P4-1 summarizes the estimation procedure for $\boldsymbol{Q}$ and $\boldsymbol{R}$. We initialize $\boldsymbol{Q} = \boldsymbol{Q}^c$ and $\boldsymbol{R} = \boldsymbol{R}^c$ (Line 1). At each iteration, we compute the gradient with respect to $\boldsymbol{Q}$ and $\boldsymbol{R}$ (Line 3), denoted d$\boldsymbol{Q}$ and d$\boldsymbol{R}$, compute the step size l (Line 4–6), and project the updated matrices (Line 7) onto the constraint set in (P4-13b) (Line 8).

Gradient computation

The gradient of the prior belief $\log p(\boldsymbol{Q}, \boldsymbol{R}|\boldsymbol{Q}^c, \boldsymbol{R}^c)$ in (P4-14) depends on the functions $\boldsymbol{t}_c$ and $\boldsymbol{t}_s$ in (P4-12). For computing the gradients of $\log p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R})$ with respect to $\boldsymbol{Q}$ and $\boldsymbol{R}$, we use (P4-10),

$$\begin{aligned}\log\, p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R}) =& \\ &\sum_{k=1}^{T} \log p(\boldsymbol{u}_k|\boldsymbol{0}, \boldsymbol{Q}) + \sum_{k=1}^{T} \log p(\boldsymbol{v}_k|\boldsymbol{0}, \boldsymbol{R}) \\ &+ \sum_{k=1}^{T} \log p(\boldsymbol{w}_k|\boldsymbol{0}, \boldsymbol{R}) - \sum_{k=1}^{T} \log p(\boldsymbol{e}_k|\boldsymbol{0}, \boldsymbol{\Gamma}_k).\end{aligned} \tag{P4-15}$$

The gradient of the first three terms in (P4-15) is computationally cheap to evaluate as it does not scale with the time duration of the collected data T, cf., (P4-2). The computational complexity of the last term scales linearly with T with $\mathcal{O}(T(n_y^4(n_y^2 + n_u^2)))$, cf., (P4-1), as the covariance matrix $\boldsymbol{\Gamma}_k$ is time-varying.

Selection of step-size

The step-size l does not affect the optimal solution but the convergence rate of the learning algorithm and is sometimes referred to as the learning rate in the literature [50]. We select the step-size by backtracking line search [51], where the idea is to reduce the step-size from an initial value l until a strict improvement is achieved (Line 4–6). The approach can be implemented efficiently for the considered application as $c(\boldsymbol{\xi}^i)$ in (P4-14) is relatively cheap to evaluate. The step-size for the next iteration $i + 1$ is initialized adaptively as in Line 9. The rationale behind this initialization is that the magnitude of the gradient does not change too abruptly between iterations i and $i + 1$.

Projection

The projection is used to enforce $\boldsymbol{Q} \in \mathcal{C}_Q, \boldsymbol{R} \in \mathcal{C}_R$. We enforce the constraint of positive definite covariance matrices using the spectral decomposition of a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ as in [52], [53],

$$\boldsymbol{X} = \boldsymbol{V}\mathrm{diag}(\lambda_1, ..., \lambda_n)\boldsymbol{V}^\mathsf{T},$$

where $\boldsymbol{V}$ comprises the eigenvectors of $\boldsymbol{X}$ and λ_i are its n eigenvalues. The projection of $\boldsymbol{X}$ onto the cone of positive definite matrices $\boldsymbol{S}_n^+$, denoted as $\mathrm{proj}_{\boldsymbol{S}_n^+}(\boldsymbol{X})$, is

$$\mathrm{proj}_{\boldsymbol{S}_n^+}(\boldsymbol{X}) = \boldsymbol{V}\mathrm{diag}(\max(\varepsilon_\lambda, \lambda_1), ..., \max(\varepsilon_\lambda, \lambda_n))\boldsymbol{V}^\mathsf{T}$$

with a small $\varepsilon_\lambda > 0$ to ensure strict positive definiteness. The computational complexity of the spectral decomposition is $\mathcal{O}(n^3)$, which is feasible for the considered problem dimension since it applies to $\boldsymbol{Q}$ and $\boldsymbol{R}$, and not to the leaning data. Additional constraints, which are specific to the application can also be included as it will be shown in Section 10.1.

Algorithm P4-1 Estimation of $\boldsymbol{Q}$, $\boldsymbol{R}$

```
1: $\boldsymbol{Q}^0 = \boldsymbol{Q}^c$, $\boldsymbol{R}^0 = \boldsymbol{R}^c$, $l = 1$, $i = 0$
2: while $l \geq \varepsilon$                                   ▷ $\varepsilon = 10^{-8}$ in our case
3:     $\mathrm{d}\boldsymbol{Q}, \mathrm{d}\boldsymbol{R} \leftarrow$ getGrad
4:     while $c(\boldsymbol{\xi}^i) - c(\boldsymbol{\xi}^i + l\nabla_{\boldsymbol{\xi}} c(\boldsymbol{\xi}^i)) > -\frac{l}{2}|\nabla_{\boldsymbol{\xi}} c(\boldsymbol{\xi}^i)|^2$
5:         $l \leftarrow \gamma l$                                ▷ $\gamma = 0.7$ in our case
6:     end while
7:     $\boldsymbol{Q} = \boldsymbol{Q}^i + l \cdot \mathrm{d}\boldsymbol{Q}$; $\boldsymbol{R} = \boldsymbol{R}^i + l \cdot \mathrm{d}\boldsymbol{R}$
8:     $\boldsymbol{Q}^{i+1}, \boldsymbol{R}^{i+1} \leftarrow$project($\boldsymbol{Q}, \boldsymbol{R}$)
9:     $l \leftarrow l/\gamma^{n_\alpha}$,   $i \leftarrow i + 1$                 ▷ $n_\alpha = 3$ in our case
10: end while
```

7 Requirements for Autonomous Driving and their Personalization

The requirements for the motion-planning application are

- to follow the centerline of a target lane,
- to maintain a nominal velocity,
- to limit longitudinal and lateral acceleration, and
- to maintain a safety distance from surrounding obstacles.

The requirements $\boldsymbol{y}_k$ in (P4-4) at time k as are formalized as

$$\boldsymbol{y}_k = \begin{bmatrix} 0 \\ v_{\text{nom}} \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} h^l(p_{X,k}, p_{Y,k}) \\ v_{x,k} \\ h^c_{\boldsymbol{\theta}}(a_{x,k}, a_{y,k}) \\ h^o_{\boldsymbol{\theta}}(d_k, v_{x,k}) \end{bmatrix},$$

where $h^l(p_X, p_Y)$ denotes the squared distance from the centerline at vehicle position p_X, p_Y, v_{nom} and v_x are the nominal and current velocity, respectively, $h^c_{\boldsymbol{\theta}}(a_x, a_y)$ is the passenger comfort requirement with longitudinal acceleration a_x and lateral acceleration a_y acting on the vehicle, and $h^o_{\boldsymbol{\theta}}(d, v_x)$ is the obstacle avoidance requirement with separation distance d

between ego vehicle (EV) and the obstacle vehicles (OVs). As both the centerline and the velocity tracking requirements are physical quantities, they are modeled as invariant and only their relative importance is learned. However, both the passenger comfort and obstacle avoidance requirements are expected to structurally vary between drivers and are introduced next.

7.1 Individual Requirements

Passenger comfort requirement

We model the passenger comfort requirement as a penalty for longitudinal and lateral accelerations, as well as their coupling. The magnitude of accelerations and their coupling are well known to relate to the individual driving style [54], where performance drivers may achieve simultaneous longitudinal and lateral acceleration, and more cautious drivers tend to do either one or the other. The passenger comfort requirement is formalized as

$$h^c_{\boldsymbol{\theta}}(a_x, a_y) = \bar{a} \cdot \frac{c_{\boldsymbol{\theta}}(a_x, a_y) - c^0}{c^1 - c^0}, \tag{P4-16}$$

with $c^1 = (\sqrt{(\bar{a}^2 + \epsilon)}^{n_c} + \sqrt{\epsilon}^{n_c})^{\frac{1}{n_c}}$, $c^0 = (2\sqrt{\epsilon}^{n_c})^{\frac{1}{n_c}}$, and

$$c_{\boldsymbol{\theta}}(a_x, a_y) = \left(\sqrt{(a_x^2 + \epsilon)}^{n_c} + \sqrt{((s \cdot a_y)^2 + \epsilon)}^{n_c}\right)^{\frac{1}{n_c}}$$

and a small $\epsilon > 0$. The parameter s defines a unilateral scaling—i.e., for $s \neq 1$, the comfort requirement is not commutative $h^c_{\boldsymbol{\theta}}(a_x, a_y) \neq h^c_{\boldsymbol{\theta}}(a_y, a_x)$—and c^1, c^0 are normalization constants ensuring that $h^c_{\boldsymbol{\theta}}(0,0) = 0$ and $h^c_{\boldsymbol{\theta}}(\bar{a}, 0) = h^c_{\boldsymbol{\theta}}(0, \bar{a}/s) = \bar{a}$. The exponent n_c shapes the level sets of (P4-16) such that, for higher n_c, the level sets are more circular, see the left plot in Fig. P4-3. For $\epsilon = 0$, $s = 1$, (P4-16) is the n_c-(pseudo)norm for $[a_x \; a_y]^\mathsf{T}$. Here, we introduce $\epsilon > 0$ (and c^1, c^0 as a consequence) for two reasons: First, $h^c_{\boldsymbol{\theta}}(a_x, a_y)$ becomes differentiable with respect to its inputs a_x, a_y for all n_c. Second, the penalty for combined longitudinal and lateral accelerations is reduced for smaller values, i.e., the level set is more circular around the origin, see the right plot in Fig. P4-3.

Obstacle avoidance requirement

The obstacle avoidance requirement is modeled as a piecewise linear function

$$h^o_{\boldsymbol{\theta}}(d, v_x) = \begin{cases} \frac{1}{t_s}\left(d_{\min} + t_s v_x - d\right) & \text{if } d_{\min} + t_s v_x \geq d \\ 0 & \text{else,} \end{cases} \tag{P4-17}$$

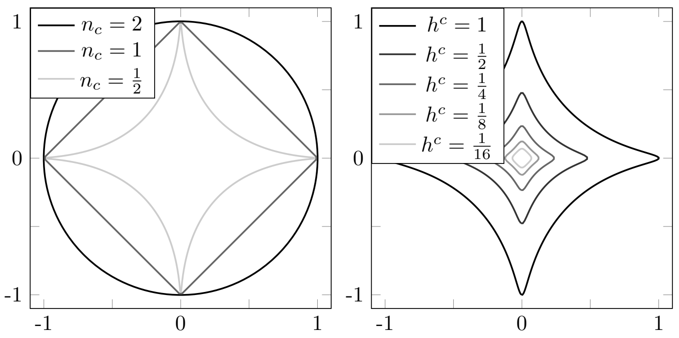


Figure P4-3. Left: Level sets $h^c = 1$ with varying n_c, $s = 1$, and $\epsilon = 0$. Right: Varying level sets h^c with $n_c = \frac{1}{2}$, $s = 1$, and $\epsilon = 0.01$.

where $d_{\min}$ is the minimum distance to be kept from the OVs and $t_s v_x$ is the traveled distance of the EV within the safety time t_s at velocity v_x and considers that the safety distance from the OVs is velocity dependent. The scaling $1/t_s$ is introduced to obtain a comparable magnitude of $h^o_{\boldsymbol{\theta}}$ for different t_s, which is important for estimating $\boldsymbol{Q}, \boldsymbol{R}$ due to the prior $p(\boldsymbol{Q}, \boldsymbol{R} | \boldsymbol{Q}^c, \boldsymbol{R}^c)$. Lateral obstacle constraints are considered through the motion planner as discussed in [7].

7.2 Estimation of Requirement Parameters

We estimate the personalized requirement parameters

$$\boldsymbol{\theta} = \begin{bmatrix} s & n_c & d_{\min} & t_s \end{bmatrix}^\mathsf{T}$$

from the distribution $p(\boldsymbol{\theta} | \boldsymbol{\theta}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T})$, which we model as $p(\boldsymbol{\theta} | \boldsymbol{\theta}^p, \boldsymbol{I}) p(\boldsymbol{\theta} | \boldsymbol{\theta}^c, \sigma_\theta^2 \boldsymbol{I})$, with the $\underline{c}$ommon parameters $\boldsymbol{\theta}^c = [s^c \; n_c^c \; d_{\min}^c \; t_s^c]^\mathsf{T}$, the $\underline{p}$ersonal parameters $\boldsymbol{\theta}^p = [s^p \; n_c^p \; d_{\min}^p \; t_s^p]^\mathsf{T}$ estimated from $\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}$, and the variance σ_θ^2. Then,

$$\boldsymbol{\theta} = \arg\max_{\tilde{\boldsymbol{\theta}}} \; p(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}^p, \boldsymbol{I}) p(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}^c, \sigma_\theta^2 \boldsymbol{I}) \tag{P4-18a}$$

$$= \frac{\sigma_\theta^2}{\sigma_\theta^2 + 1} \boldsymbol{\theta}^p + \frac{1}{\sigma_\theta^2 + 1} \boldsymbol{\theta}^c. \tag{P4-18b}$$

Next, we present algorithms for estimating $\boldsymbol{\theta}^p$ from data of human driving.

Estimation of passenger comfort requirement

The estimation of the parameters n_c^p and s^p is achieved by Algorithm P4-2 and described next. Let

$$\begin{aligned}\mathcal{A} &= \left\{ \begin{bmatrix} a_{x,0} \\ a_{y,0} \end{bmatrix}, ..., \begin{bmatrix} a_{x,T} \\ a_{y,T} \end{bmatrix} \right\} \\ \mathcal{A}_\mathrm{c} &= \{a_{x,0}, ..., a_{x,T}, a_{y,0}, ..., a_{y,T}\} \\ \mathcal{A}_x &= \{a_{x,0}, ..., a_{x,T}\}\,,\ \mathcal{A}_y = \{a_{y,0}, ..., a_{y,T}\}\,.\end{aligned}$$

Estimation of s^p

We compute the unilateral scaling parameter s^p as the ratio between longitudinal and lateral accelerations, each represented by the median as a measure of central tendency. We use the median, rather than the mean, as a robust estimator in the presence of outliers. More specifically, s^p results from the median of the M largest longitudinal accelerations and divided by the median of the M largest lateral accelerations (Line 1 in Algorithm P4-2).

Estimation of n_c^p

In order to estimate n_c^p, we first estimate a_max denoting the value of the level set of the largest accelerations. We compute a_max as the median of the M largest elements in absolute value of the set defined by $\mathcal{A}_x \cup (s \cdot \mathcal{A}_y)$, where $s \cdot \mathcal{A}_y = \{s \cdot a_{y_0}, ..., s \cdot a_{y_T}\}$ (Line 2). Then, n_c^p is obtained by estimating the shape of the level set a_max using the passenger comfort requirement as (pseudo)norm. Thereby, we compute a set with a strong coupling of longitudinal and lateral accelerations using a small $n_{c,0}$, denoted $\mathcal{F}_0$. Finally, the exponent n_c^p is increased iteratively until the median comfort level in $\mathcal{F}_0$ is greater than or equal to a_max (Line 5–9).

Estimation of obstacle avoidance requirement

We use a system identification-like approach similar to [55] to estimate the parameters d_min^p and t_s^p, as described in Algorithm P4-3 and explained next. The intuitive idea is that the observed data originate from either of two models: driving with or without traffic. Considering the piecewise linear $h_{\boldsymbol{\theta}}^o$ in (P4-17), we want the switch between the two models to coincide with $d_\mathrm{min}^p = d - t_s^p v_x$, where $d_\mathrm{min}^p < d - t_s^p v_x$ indicates traffic-free and $d_\mathrm{min}^p > d - t_s^p v_x$ is traffic-affected driving.

Algorithm P4-2 Estimation of n_c^p, s^p given ε

```
1: $s^p = \mathrm{med}(\mathcal{A}_x^\star)/\mathrm{med}(\mathcal{A}_y^\star)$ with $\mathcal{A}_x^\star = \mathrm{setmax}_{|\cdot|}(\mathcal{A}_x, M)$, $\mathcal{A}_y^\star = \mathrm{setmax}_{|\cdot|}(\mathcal{A}_y, M)$
2: $a_{\max} = \mathrm{med}(A_c^\star)$, $A_c^\star = \mathrm{setmax}_{|\cdot|}(\mathcal{A}_x \cup (s \cdot \mathcal{A}_y), M)$
3: Choose smallest candidate exponent $n_{c,0}$.
4: $\mathcal{F}_0 = \mathrm{setmax}_{h^c}(\mathcal{A}, M)$ using $n_{c,0}$ and set $n_c^p = n_{c,0}$
5: do
6:    Compute comfort levels $\mathcal{C}_0$ of elements in $\mathcal{F}_0$ with $n_c^p$
7:    $\bar{\delta}_0^c = \mathrm{med}(\mathcal{C}_0)$
8:    $n_c^p \leftarrow n_c^p + \delta n_c$.                          ▷ $\delta n_c = 0.01$ in our case
9: while $\bar{\delta}_0^c \geq a_{\max}$
```

Estimation of $d_{\min}^p$

We estimate $d_{\min}^p$ as the maximum value of the M smallest observed distances with $\mathcal{D} = \{d_0, ..., d_T\}$ (Line 1 in Algorithm P4-3). In other words, $d_{\min}^p$ is designed as the Mth smallest observed distance, which is a more robust and conservative estimator than, e.g., the smallest distance.

Estimation of t_s^p

Let

$$\boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} h^l(p_X, p_Y) & v_x & h_{\boldsymbol{\theta}}^c(a_x, a_y) \end{bmatrix}^\mathsf{T} \tag{P4-19a}$$

$$\boldsymbol{H}_k^r = \frac{\partial \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}, \hat{\boldsymbol{u}}_k)}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}, \; \boldsymbol{D}_k^r = \frac{\partial \boldsymbol{h}_{\boldsymbol{\theta}}^r(\hat{\boldsymbol{x}}_k, \boldsymbol{u})}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_k} \tag{P4-19b}$$

$$\boldsymbol{v}_k^r = \begin{bmatrix} 0 & v_{\mathrm{nom}} & 0 \end{bmatrix}^\mathsf{T} - \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{P4-19c}$$

$$\boldsymbol{e}_k^r = \begin{bmatrix} 0 & v_{\mathrm{nom}} & 0 \end{bmatrix}^\mathsf{T} - \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{0}), \tag{P4-19d}$$

where r denotes reduced (by the obstacle avoidance requirement). If the parameters of $h_{\boldsymbol{\theta}}^c$ are known (Section 7.2), we can use Algorithm P4-1 to estimate $\boldsymbol{Q}^r$ and $\boldsymbol{R}^r$ using (P4-19) for traffic-free driving (Line 2 in Algorithm P4-3). Further, in (P4-8), let $\boldsymbol{K}_k = \boldsymbol{K}_k^{\mathrm{TF}} + \boldsymbol{K}_k^{\mathrm{TA}}$, where $\boldsymbol{K}_k^{\mathrm{TF}} = \boldsymbol{K}_k^r \boldsymbol{T}$ is the gain matrix in the absence of traffic (traffic-free) with $\boldsymbol{T} = [\boldsymbol{I}_3 \; \boldsymbol{0}_{3\times 1}]$,

$$\boldsymbol{K}_k^r = \boldsymbol{Q}^r \boldsymbol{J}_k^{r\mathsf{T}} \left(\boldsymbol{J}_k^r \boldsymbol{Q}^r \boldsymbol{J}_k^{r\mathsf{T}} + \boldsymbol{R}^r \right)^{-1}, \; \boldsymbol{J}_k^r = \boldsymbol{H}_k^r \boldsymbol{G}_k + \boldsymbol{D}_k^r,$$

and $\boldsymbol{K}_k^{\mathrm{TA}}$ is the residual gain matrix (traffic-affected). This decomposition of $\boldsymbol{K}_k$ in (P4-8) yields

$$\boldsymbol{\sigma}_k = \boldsymbol{u}_k - \boldsymbol{K}_k^{\mathrm{TF}} \boldsymbol{e}_k \sim \mathcal{N}\left(\boldsymbol{K}_k^{\mathrm{TA}} \boldsymbol{e}_k, \boldsymbol{\Sigma}_k \right).$$

Hence, in the absence of traffic $\boldsymbol{K}_k^{\mathrm{TA}}\boldsymbol{e}_k = \boldsymbol{0}$, $\boldsymbol{\sigma}_k = \boldsymbol{u}_k - \boldsymbol{K}_k^{\mathrm{TF}}\boldsymbol{e}_k$ is sampled from a distribution with zero mean and, in the presence of traffic $\boldsymbol{K}_k^{\mathrm{TA}}\boldsymbol{e}_k \neq \boldsymbol{0}$, $\boldsymbol{u}_k - \boldsymbol{K}_k^{\mathrm{TF}}\boldsymbol{e}_k$ is sampled from a distribution with mean $\boldsymbol{K}_k^{\mathrm{TA}}\boldsymbol{e}_k$.

We use this change in mean for estimating t_s^p with

$$t_s^p = \arg\min_{\tilde{t}_s^p, \mathcal{T}, \boldsymbol{a}_i} \sum_{k\in\mathcal{T}} I_k + \sum_{k\notin\mathcal{T}} J_k \tag{P4-20a}$$

with $I_k = \|\boldsymbol{u}_k - \boldsymbol{K}_k^r\boldsymbol{e}_k^r\|_2^2$, $J_k = \|\boldsymbol{u}_k - \boldsymbol{K}_k^r\boldsymbol{e}_k^r - \boldsymbol{p}(d_k, v_{x,k})\|_2^2$, and

$$\mathcal{T} = \{i \mid d_{\min}^p + \tilde{t}_s^p v_{x,i} \leq d_i\}, \tag{P4-20b}$$

where $\boldsymbol{p}(d, v_x) = \sum_{i=0}^{1} \boldsymbol{a}_i \left(d_{\min}^p + \tilde{t}_s^p v_x - d\right)^i$ with the coefficients $\boldsymbol{a}_i \in \mathbb{R}^{n_u}$ is used to approximate the nonzero mean. Note that (P4-20) is a combinatorial problem, however, for a fixed t_s^p, it reduces to a convex least squares problem in $\boldsymbol{a}_i$. We solve (P4-20) by enumerating t_s^p in Δt_{inc} increments (Line 3–6).

Algorithm P4-3 Estimation of $d_{\min}^p$, t_s^p

```
1: $d_{\min}^p = \max(\mathcal{D}^\star)$ with $\mathcal{D}^\star = \mathrm{setmin}_{|\cdot|}(\mathcal{D}, M)$
2: Get $\boldsymbol{Q}^r, \boldsymbol{R}^r$ with Algorithm 1 for traffic-free driving
3: for all $t_s^p \in [0\mathrm{s}\ t_{s,\max}]$ in $\Delta t_{\mathrm{inc}}$ increments
4:     Compute $\mathcal{T}$ in (P4-20b)
5:     Solve (P4-20) with fixed $t_s^p$ and $\mathcal{T}$ for $\boldsymbol{a}_i$
6: end for                    ▷ $t_{s,\max} = 10\mathrm{s}$, $\Delta t_{\mathrm{inc}} = 0.01\mathrm{s}$ in our case
7: Choose $t_s^p$ with smallest cost (P4-20a)
```

8 Overall Algorithm, Variants, and Computational Requirements

In this section, we summarize the overall inverse learning algorithm, where Problem 1 and Problem 2 as stated in Section 5 were addressed as follows:

Result 1. *Algorithm P4-1 solves Problem 1.*

Result 2. *Eq.* (P4-18) *using $\boldsymbol{\theta}^p$ estimated with Algorithm P4-2 and Algorithm P4-3 solves Problem 2.*

Furthermore, we present three variants for the practical implementation of the proposed inverse learning method in this section. The three variants have different computational complexities and data storage requirements, as well as model assumptions and approximations. Variant I is the unmodified algorithm as presented in Section 6 and Section 7. Variant II uses an

Table P4-1. Variants of Algorithm

	Data Storage	Computational Complexity
Variant I	$\mathcal{O}(Tn_{\mathcal{D}})$	$\mathcal{O}(T(n_y^4(n_y^2+n_u^2)))$
Variant II	$\mathcal{O}(Tn_{\mathcal{D}})$	$\mathcal{O}(n_y^4(n_y^2+n_u^2))$
Variant III	$\mathcal{O}(3n_y^2+n_u^2)$	$\mathcal{O}(n_y^4(n_y^2+n_u^2))$

approximation of the time-varying covariance $\mathbf{\Gamma}_k$ as time-invariant $\mathbf{\Gamma}$. Variant III uses the same approximation $\mathbf{\Gamma}_k \approx \mathbf{\Gamma}$ and, additionally, models the parameters of the requirement function as constant $\boldsymbol{\theta} = \boldsymbol{\theta}^c$. Table P4-1 specifies the expected data storage and computational complexity for the three variants.

Variant I

The implementation of Variant I requires the storage of data that scale linearly with T, where $n_{\mathcal{D}}$ in Table P4-1 is the number of numerical values to be stored at each time-step. For the considered application, not all data need to be stored, e.g., the road data and both the EV's and an OV's positions are sufficiently represented by the centerline tracking error and separation distance d between the EV and the OV. The computational complexity is linear in T, see Section 6.2. Algorithm P4-4 outlines the estimation procedure for the parameters $\boldsymbol{Q}$, $\boldsymbol{R}$, and $\boldsymbol{\theta}$.

Variant II

The implementation of Variant II has the same storage requirements as Variant I. However, Variant II has lower computational complexity due to approximating $\mathbf{\Gamma}_k \approx \mathbf{\Gamma} = (\boldsymbol{HG}+\boldsymbol{D})\boldsymbol{Q}(\boldsymbol{HG}+\boldsymbol{D})^{\mathsf{T}}+\boldsymbol{R}$ with some constant $\boldsymbol{H}$, $\boldsymbol{G}$, and $\boldsymbol{D}$. Algorithm P4-4 outlines the estimation procedure, where Line 2 and 4 are considerably less computationally demanding.

Variant III

In addition to approximating $\mathbf{\Gamma}_k \approx \mathbf{\Gamma}$, in Variant III we model the requirement function parameters as constant $\boldsymbol{\theta} = \boldsymbol{\theta}^c$, i.e., the requirement function is not personalized. Then, the sample covariance matrices

$$\hat{\boldsymbol{U}}_T = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{u}_k \boldsymbol{u}_k^\mathsf{T},\ \hat{\boldsymbol{V}}_T = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{v}_k \boldsymbol{v}_k^\mathsf{T},$$
$$\hat{\boldsymbol{W}}_T = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{w}_k \boldsymbol{w}_k^\mathsf{T},\ \hat{\boldsymbol{E}}_T = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{e}_k \boldsymbol{e}_k^\mathsf{T}$$

define a sufficient statistic for the distribution in (P4-10), i.e., the data can be compressed into the sample covariance matrices without losing information. As a result, the data size to be stored is independent of T as the sample covariance matrices can be updated recursively, e.g.,

$$\hat{\boldsymbol{U}}_T = \frac{1}{T}\boldsymbol{u}_T \boldsymbol{u}_T^\mathsf{T} + \frac{T-1}{T}\hat{\boldsymbol{U}}_{T-1}.$$

For Variant III, the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are estimated as in Line 4 in Algorithm P4-4 using $\boldsymbol{h}_{\boldsymbol{\theta}^c}$. In this case, the closed-loop behavior is still personalized but only through $\boldsymbol{Q}$, $\boldsymbol{R}$.

Algorithm P4-4 Overall estimation procedure for $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{\theta}$

```
1: Get $s, n_c$ in (P4-18) using $s^p, n_c^p$ estimated with Alg. 2
2: Get $\boldsymbol{Q}^{\mathrm{TF}}, \boldsymbol{R}^{\mathrm{TF}}$ using Alg. 1 with $\boldsymbol{h}_{\boldsymbol{\theta}}^r$
3: Get $d_{\min}, t_s$ in (P4-18) using $d_{\min}^p, t_s^p$ estimated with Alg. 3
4: Get $\boldsymbol{Q}, \boldsymbol{R}$ using Alg. 1 with $\boldsymbol{h}_{\boldsymbol{\theta}}$
```

9 Simulation Setup with Human Driver

We carried out simulations with human participants who controlled a vehicle in CarSim using the torque-feedback Thrustmaster T300RS gaming steering wheel with a MATLAB interface, see Fig. P4-4. We constructed a two-lane oval circuit with two straight segments of 200 m connected by two 180° turns with radius 53.6 m at the centerline of the right lane, resulting in $(2 \cdot 200 + 2\pi \cdot 53.6)$ m length. Both lanes were 3.6 m in width. The ego vehicle and the obstacle vehicles drove anti-clockwise. Each test driver completed the following:

Task 0

The driver familiarized themself with the driving simulator and was prepared for Task 1 and Task 2. No data were recorded during this task.

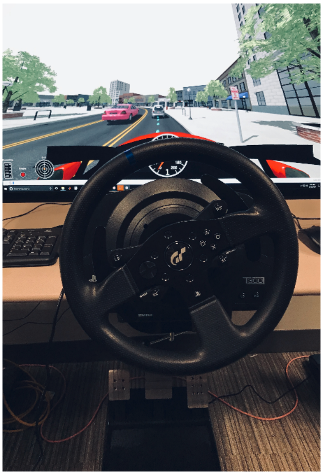


Figure P4-4. Simulation setup for learning from data of human driver.

Table P4-2. Obstacle Vehicles' Initial Positions & Velocities

Vehicle ID	1	2	3	4	5	6	7
Initial Position [m]	100	300	500	600	350	550	600
Lane	right	right	right	right	left	left	left
Velocity [km/h]	19.8	19.8	19.8	19.8	16.2	16.2	16.2

Task 1 (15 min recorded)

The driver was instructed to stay in the right lane and that the nominal velocity is 50 km/h. This task did not involve OVs.

Task 2 (15 min recorded)

The driver was allowed to use both lanes and the nominal velocity was again 50 km/h. We added 7 OVs, as specified in Table P4-2, where the initial position is the distance along the track and the start at 0 m is the beginning of a straight segment and is the initial position of the EV. The OVs drove slowly to increase the number of following and overtake actions the driver will demonstrate.

10 Learning Results & Hardware Requirements

Five human drivers participated in the study. Four of them were normal drivers and one (Driver 3) had professional test driving training, and aimed at exercising a performance driving style. The driving frequency and experience of the five participants are stated in Table P4-3. We present the parameter estimation results in this section and the behavior of the motion planner that uses such parameters in Section 11.

10.1 Design Choices

We consider the kinematic single track vehicle model [56]

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{p}_X \\ \dot{p}_Y \\ \dot{\psi} \\ \dot{v}_x \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v_x \cos(\psi+\beta)/\cos(\beta) \\ v_x \sin(\psi+\beta)/\cos(\beta) \\ v_x \tan(\delta)/L \\ u_1 \\ u_2 \end{bmatrix}$$

represented in discrete time with the sampling period $T_s = 0.5$ s, where p_X and p_Y mark the EV's position in the world frame, ψ is the heading (yaw) angle, v_x is the longitudinal velocity, δ is the steering angle of the front wheel, $L = l_f + l_r$ is the wheel base, and $\beta = \arctan(l_r \tan(\delta)/L)$ is the kinematic body-slip angle. Accelerations are computed as $a_x = \dot{v}_x$ and $a_y = v_x\dot{\psi}$. The inputs u_1 and u_2 are the longitudinal acceleration and the steering rate.

Estimation of requirement function parameters

We choose the variance over the common parameters as $\sigma_\theta^2 = (T/T_{1/2})^2$ such that $\boldsymbol{\theta} \to \boldsymbol{\theta}^p$ for $T \to \infty$, $\boldsymbol{\theta} = 0.5\boldsymbol{\theta}^p + 0.5\boldsymbol{\theta}^c$ for $T_{1/2} = T$, and $\partial\boldsymbol{\theta}/\partial T|_{T=0} = \mathbf{0}$, i.e., not transitioning too quickly from $\boldsymbol{\theta}^c$. We choose $T_{1/2} = 5 \cdot 60/T_s$, i.e., $T_{1/2} = T$ corresponds to five minutes of driving. Further, we choose $M = \mathrm{round}(0.01N)$ for estimating $\boldsymbol{\theta}^p$ and design the passenger comfort requirement with $\bar{a} = 5$ and $\epsilon = 0.01$.

Constraints $\mathcal{C}_Q, \mathcal{C}_R$

We require $\boldsymbol{Q}$ to be diagonal because, if $\boldsymbol{Q}$ had nonzero off-diagonal elements, the longitudinal acceleration and steering rate would more likely be coupled. For instance, if $q_{12} > 0$ ($\mathbb{E}[u_1 u_2] > 0$), accelerating ($\dot{v}_{x,k} = u_1 > 0$) would imply a preference on steering to the right ($\dot{\delta} = u_2 > 0$), which is unnatural. Further, we impose $\boldsymbol{Q} = \boldsymbol{Q}^\mathsf{T} \succeq \varepsilon_\lambda \cdot \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{R}^\mathsf{T} \succeq \varepsilon_\lambda \cdot \boldsymbol{I}$ with $\varepsilon_\lambda = 10^{-3}$ and model the centerline tracking as independent of the other requirements with $r_{12} = r_{13} = r_{14} = 0$ to avoid oscillations, e.g., of the velocity with the centerline tracking error ($\mathbb{E}[(v_{\text{nom}} - v_x)(0 - h^l)] = 0$).

Prior (structural belief)

We design the structural belief to accommodate particle-filter algorithms, which we use to solve the motion-planning problem. In this context, the signal-to-noise ratio $\|\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\mathsf{T}\|/\|\boldsymbol{R}\|$ with $\boldsymbol{J} = \boldsymbol{H}\boldsymbol{G} + \boldsymbol{D}$ is to be chosen close to one [57]. We choose $\boldsymbol{t}_s(\boldsymbol{Q}, \boldsymbol{R}) = \text{vec}(\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^\mathsf{T} - \boldsymbol{R})$ and $\sigma_s^2 = \frac{1}{T}$ with $\boldsymbol{G} = \boldsymbol{g}(\boldsymbol{x}^\star)$, $\boldsymbol{H} = \partial \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{0})/\partial \boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{x}^\star}$, and $\boldsymbol{D} = \partial \boldsymbol{h}(\boldsymbol{x}^\star, \boldsymbol{u})/\partial \boldsymbol{u}|_{\boldsymbol{u}=\boldsymbol{0}}$, where $\boldsymbol{x}^\star$ denotes a nominal state where all requirements are fulfilled ($v_x = 50\text{km/h}$, $\delta = 0$).

Prior (common belief)

We choose $\sigma_c^2 = \sigma_\theta^2$ and $\boldsymbol{t}_c(\boldsymbol{Q}, \boldsymbol{R}) = [\text{vec}(\boldsymbol{Q} - \boldsymbol{Q}^c)^\mathsf{T}\ \text{vec}(\boldsymbol{R} - \boldsymbol{R}^c)^\mathsf{T}]^\mathsf{T}$. Then,

$$p\left(\boldsymbol{t}_c(\boldsymbol{Q}, \boldsymbol{R}) | \boldsymbol{0}, \sigma_c^2 \boldsymbol{I}\right) = \prod_{\forall ij} p\left(q_{ij} | q_{ij}^c, \sigma_c^2\right) \prod_{\forall ij} p\left(r_{ij} | r_{ij}^c, \sigma_c^2\right)$$

defines a Gaussian distribution over each element of $\boldsymbol{Q}^c, \boldsymbol{R}^c$. The common parameters are chosen as the estimate of Variant III using all data from the five drivers combined,

$$\boldsymbol{Q}^c = \text{diag}(16.1,\ 0.0016) \tag{P4-21a}$$

$$\boldsymbol{R}^c = \begin{bmatrix} 0.105 & 0 & 0 & 0 \\ & 16.2 & -4.10 & -14.7 \\ & & 5.02 & 4.38 \\ & & & 17.4 \end{bmatrix} \tag{P4-21b}$$

$$\boldsymbol{\theta}^c = \begin{bmatrix} s & n_c & d_{\min} & t_s \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 1 & 1 & 8 & 5 \end{bmatrix}^\mathsf{T}. \tag{P4-21c}$$

10.2 Estimation Results

Personalizing parameters over time (0 min – 15 min of driving)

First, we analyze the gradual personalization of both the requirement function parameters and the covariance matrices as data are collected over time using the three variants proposed in Section 8. Fig. P4-5 shows the transition of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{Q}, \boldsymbol{R}$ from the initial common parameters $\boldsymbol{\theta}^c$ and $\boldsymbol{Q}^c, \boldsymbol{R}^c$ (y-axis) for Task 1 (traffic-free scenario), over time (x-axis). The variances σ_θ^2, σ_c^2, and σ_s^2 are chosen such that the parameters do not transition too quickly from the common ones, but also that the personalization does not require too much training time, which can best be seen for Driver 3. The first observation is that the parameters estimated with

Variant I and Variant II do not deviate much. This suggests that the approximation $\mathbf{\Gamma}_k \approx \mathbf{\Gamma}$ of Variant II is acceptable for the motion-planning application. For Variant III, the parameters of the requirement function are kept constant $\boldsymbol{\theta} = \boldsymbol{\theta}^c$ and, as a result, the covariance matrices deviate between Variant II and Variant III, most noticeable for R_{33} ($\mathbb{E}[(0 - h^o)^2]$) and R_{23} ($\mathbb{E}[(v_{\text{nom}} - v_x)(0 - h^o)]$).

Personalized parameters (30 min of driving)

Table P4-3 specifies the covariance matrices $\boldsymbol{Q}$, $\boldsymbol{R}$ and the parameters s, n_c of the passenger comfort requirement as well as $d_{\min}, t_s$ of the obstacle avoidance requirement for the five drivers, estimated with Variant II. It also shows scatter plots of accelerations (absolute values) demonstrated by the five drivers. All demonstrated accelerations of each driver are displayed in gray. The level set $h^c_{\boldsymbol{\theta}}(a_x, a_y) = a_{\max}$ is displayed as black line, where s is estimated using the green data points (median of largest longitudinal divided by the largest lateral accelerations) and n_c is such that the median of the black data points is $a_{\max}$. The scatter plots show that Driver 3's driving style yields high lateral accelerations with $a_{\max}/s = 7.32$ m/s^2 relative to Driver 1, 2, 4, and 5 with $a_{\max}/s = 2.68$ m/s^2, 1.45 m/s^2, 2.79 m/s^2, and 3.41 m/s^2, respectively. It indicates that, in general, Driver 1 and Driver 4 exhibit similar accelerations, whereas Driver 2 and Driver 5 avoid larger accelerations. Driver 2 is the most conservative keeping a minimum distance to OVs of 13.6 m, compared to $d_{\min} < 9$ m for the other drivers, and reacting to OVs at $t_s = 7.00$ s. Driver 3 is the least conservative with $d_{\min} = 5.89$ m and $t_s = 3.21$ s. The covariance matrices can be interpreted as follows: Low values represent a low tolerance of violating the respective requirement, e.g., $r_{22} = 1.63$ of Driver 3 versus $r_{22} > 10$ for all other drivers indicates that Driver 3 is more reluctant to deviate from the nominal velocity. Further, low off-diagonal values relative to their diagonal counterparts represent little coupling of the respective two requirements, e.g., $r_{23} = -0.725$ of Driver 3 indicates that Driver 3 is not as likely to reduce their velocity for the sake of reducing lateral accelerations on the vehicle. An important off-diagonal element is r_{24}, which represents the covariance of the velocity and obstacle avoidance requirement. For $r_{24} < 0$ ($\mathbb{E}[(v_{\text{nom}} - v_x)(0 - h^o)] < 0$), the driver is more likely to reduce the velocity when encountering traffic on the target lane. Similarly, for $r_{34} > 0$ ($\mathbb{E}[(0 - h^c)(0 - h^o)] > 0$), the driver is more likely to sacrifice comfort in traffic.

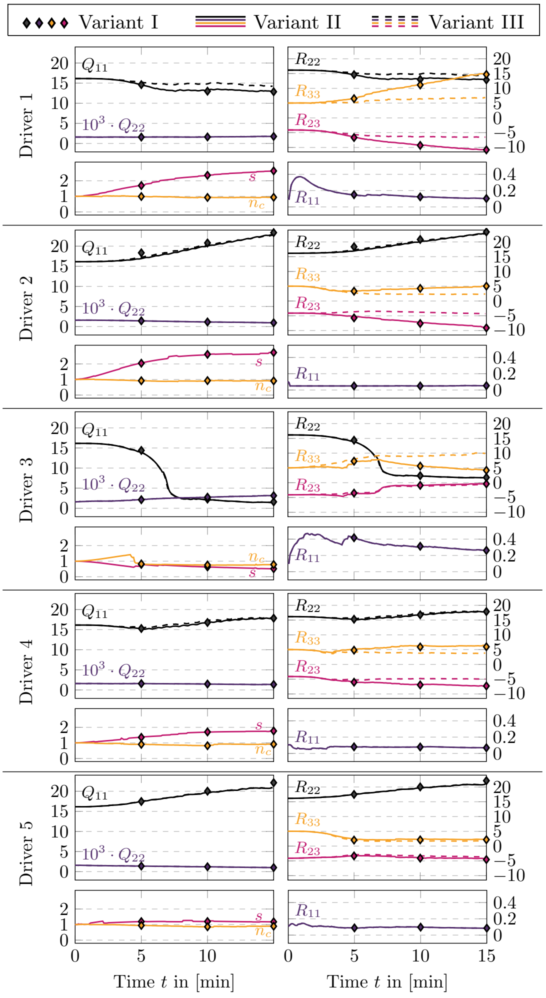


Figure P4-5. Parameter estimated as data are collected over time with Variant I (diamond symbols), Variant II (solid lines), and Variant III (dashed lines) for five drivers. The parameters $\boldsymbol{\theta}, \boldsymbol{Q}, \boldsymbol{R}$ are obtained by solving Algorithm P4-4 using different amount of data from time 0min through time t (x-axis).

Table P4-3. Estimated Parameters & Acceleration Scatter Plots

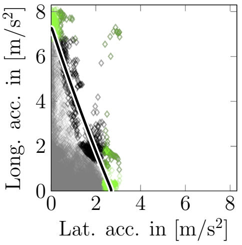
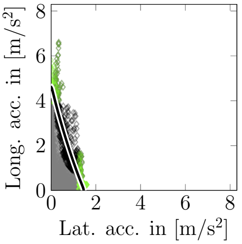
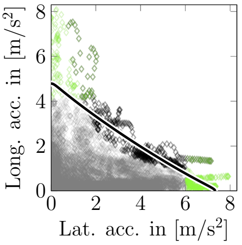
Driver 1 Occasional driver $\boldsymbol{Q} = \text{diag}(13.6,\ 0.0017)$ $\boldsymbol{R} = \begin{bmatrix} 0.085 & 0 & 0 & 0 \\ & 13.1 & -6.39 & -11.4 \\ & & 6.68 & 5.29 \\ & & & 14.3 \end{bmatrix}$ $s = 2.71,\ n_c = 0.95$ $d_{\text{min}} = 7.32$ m, $t_s = 4.10$ s	
Driver 2 Occasional driver $\boldsymbol{Q} = \text{diag}(30.3,\ 0.0009)$ $\boldsymbol{R} = \begin{bmatrix} 0.055 & 0 & 0 & 0 \\ & 27.0 & -5.31 & -24.9 \\ & & 2.69 & 3.94 \\ & & & 28.8 \end{bmatrix}$ $s = 3.17,\ n_c = 0.89$ $d_{\text{min}} = 13.6$ m, $t_s = 7.00$ s	
Driver 3 Experienced driver with professional training $\boldsymbol{Q} = \text{diag}(1.37,\ 0.0032)$ $\boldsymbol{R} = \begin{bmatrix} 0.257 & 0 & 0 & 0 \\ & 1.63 & -0.725 & -0.214 \\ & & 10.1 & 0.0856 \\ & & & 1.54 \end{bmatrix}$ $s = 0.65,\ n_c = 0.9$ $d_{\text{min}} = 5.89$ m, $t_s = 3.21$ s	

Table P4-3. (Continued)

Driver 4

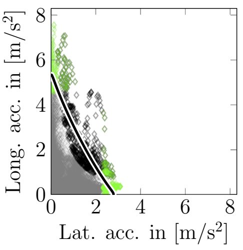
Everyday standard driver, naturally cautious

$\boldsymbol{Q} = \mathrm{diag}(19.5,\ 0.0013)$

$$\boldsymbol{R} = \begin{bmatrix} 0.066 & 0 & 0 & 0 \\ & 19.0 & -5.22 & -17.5 \\ & & 3.87 & 3.78 \\ & & & 20.0 \end{bmatrix}$$

$s = 1.92$, $n_c = 0.86$

$d_{\min} = 8.52$ m, $t_s = 5.87$ s



Driver 5

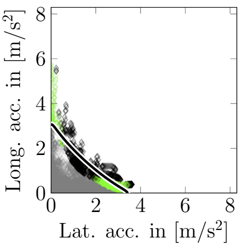
Occasional driver with professional training

$\boldsymbol{Q} = \mathrm{diag}(23.7,\ 0.0010)$

$$\boldsymbol{R} = \begin{bmatrix} 0.083 & 0 & 0 & 0 \\ & 23.7 & -4.06 & -21.9 \\ & & 1.79 & 2.95 \\ & & & 25.3 \end{bmatrix}$$

$s = 0.9$, $n_c = 0.81$

$d_{\min} = 6.42$ m, $t_s = 6.35$ s



10.3 Hardware Requirements & Computational Cost

Table P4-4 shows the hardware requirements for the three variants. The data storage requirements of the algorithms using `mex` functions created with MATLAB are 982 kB for Variant I and Variant II, and 307 kB for Variant III. Storing the data of 30 min driving requires 1440 kB for Variant I and Variant II, however, Variant III requires only a few parameters that can be updated online (recursively). On average, Variant I requires 408 s, Variant II requires around 3 s, and Variant III requires less than 0.5 s, using MATLAB with the hardware configuration: 3.1 GHz Intel Core i7, 16 GB 1867 MHz DDR3, and Intel Iris Graphics 6100 1536 MB.

Variant I and Variant II yield very similar parameters for the motion-planning application, see Fig. P4-5. As Variant II is comparably cheap, we

Table P4-4. Hardware Requirements for the considered Application

	Data Storage		CPU Time
	Algorithm	30 min Driving	
Variant I	982 kB	1440 kB	408 s
Variant II	982 kB	1440 kB	3.63 s
Variant III	307 kB	0.44 kB	0.356 s

favor Variant II over Variant I and employ only Variant II and Variant III in the following.

11 Personalized Motion Planning

We use the particle-filter algorithm in [7] with the proposed (personalized) requirement function and estimated covariance matrices for validation, where we refer to the motion planner trained with the data obtained from Driver x as Planner x, and Planner C refers to the motion planner using the common parameters in (P4-21). In what follows, we show results of planners obtained by specific combinations of variants and drivers to make the features of the method more evident.

11.1 Results

Fig. P4-6 and Fig. P4-7 show the trajectories of driving without OVs (Task 1) as mean and standard deviation over all laps of Driver 1–5 and Planner 1–5. Fig. P4-6 compares the autonomous motion planners trained with Variant II and Variant III and Fig. P4-7 shows the planners for different durations of training with Variant II. The figures display the velocity, the distance to the centerline ΔCL, and the lateral acceleration over the track position from 0 to 50% of the track, where the first segment refers to the straight and the second segment is the 180° turn.

Personalized planners (30 min of driving)

Consider first Planner 1–5 trained with Variant II and all available data (dashed black)—i.e., 30 min of driving—for a comparison with their respective drivers (solid black). It can be seen that the velocity and the lateral acceleration of the drivers and their respective planners match relatively closely with some notable exceptions. Driver 1, 3, 4 exceeded the nominal velocity of 50 km/h on average on the straight segment. The path

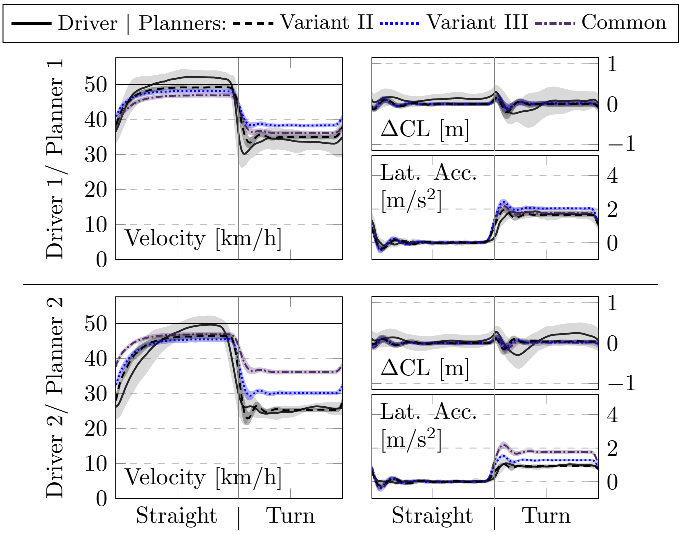


Figure P4-6. Comparison of Variant II and Variant III. The velocity, centerline tracking error, and lateral acceleration are displayed as functions of the track position for Driver x (solid black), Planner C (dashdotted purple), Variant II (dashed black), and Variant III (dashed blue).

planner avoids almost always exceeding the nominal velocity by design as it is easier to fulfill the other requirements using a lower velocity, which is desirable for autonomous driving from a safety perspective. Planner 5 matches the lateral accelerations of Driver 5 relatively closely. Driver 5 appears to be suboptimal in many directions: The driver did not reach the nominal velocity on the straight segment and has a large error for tracking the centerline. Thus, despite the learning, the path planner decides not to imitate these negative behaviors, and it only follows the one that makes physical sense, i.e., the reduced lateral acceleration. Further, the planners track the centerline more closely than the drivers, which is due to the optimization of the requirements.

Comparison of Variant II and Variant III

Fig. P4-6 shows a comparison between Variant II and Variant III for Driver 1 and Driver 2, who tend to avoid higher lateral accelerations

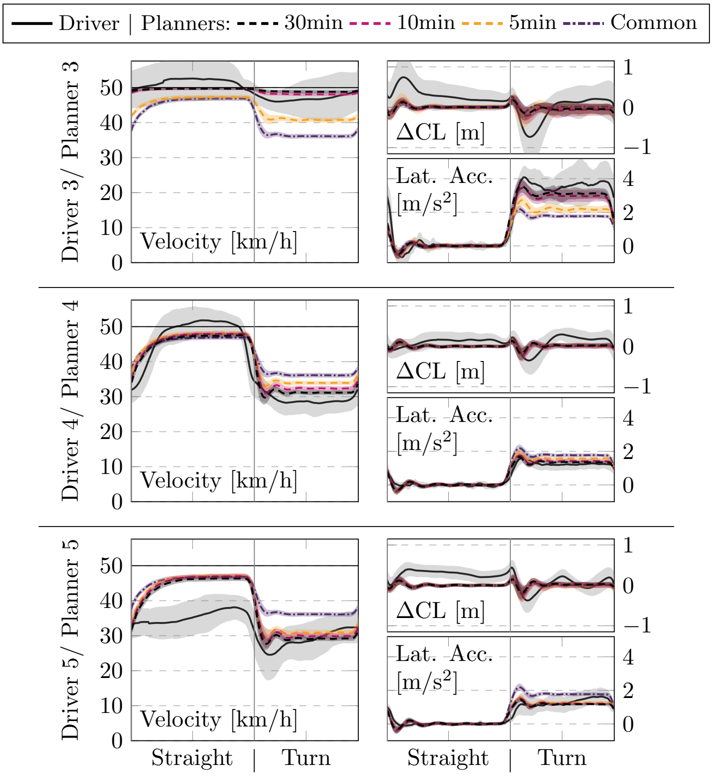


Figure P4-7. Personalizing planners over time. The velocity, centerline tracking error, and lateral acceleration are displayed as functions of the track position for Driver x (solid), Planner C (dashdotted purple), Variant II with 5 min of driving data (dashed yellow), Variant II with 10 min of driving data (dashed magenta), and Variant II with 30 min of driving data (dashed black).

compared to longitudinal accelerations ($s > 2$ in (P4-16)). The planners trained with Variant II with personalized requirement function match the experiments very closely for both drivers. The planners trained with Variant III capture the individual driving behavior, too, but deviate from their respective drivers slightly more. For example, Driver 2 avoids larger lateral accelerations with $|a_y| \approx 1$ m/s^2 during turns, which the Planner 2 trained with Variant II matches closely. Also, Variant III avoids the higher lateral accelerations of the motion planner with the common parameters ($|a_y| \approx 1.8$ m/s^2) to maintain lateral accelerations with $|a_y| \approx 1.3$ m/s^2 during the turn. For Driver 3–5 (not displayed), the difference between Variant II and Variant III is less significant, which is due to the parameters s and n_c being closer to one.

Personalizing planners over time (0 min – 30 min)

Fig. P4-7 shows the planners' trajectories for different training durations. It displays the trajectories with 0 min of training (common parameters), 5 min, 10 min, and all available data with 30 min. The gradual personalization can best be seen for Planner 3 with velocities during the turn of 36 km/h, 41 km/h, 48 km/h, and 49 km/h for the increasing training durations. For Driver 1 and Driver 2 (not displayed), the gradual personalization of the motion planners is similar as for the displayed drivers.

Planners in obstacle situation (Variant II)

Next, we consider a motion-planning scenario where both lanes are initially blocked by two slow OVs, velocities 30 km/h and 25 km/h on the right and left lane, respectively. Approaching the blockage, the planner must slow down and wait for the opportunity to overtake. Fig. P4-8 illustrates the resulting trajectories of Planner 1–3 as well as for the planner with the common parameters (P4-21). It displays the EV's velocity, the minimum distance to the other vehicles in the target lane, and a snapshot of the path, where the positions of the two OVs are frozen at the time of lane-change decision. It shows that Planner 2 is the most conservative starting to decelerate early ($v_x < 35$ km/h) and keeping the largest distance from the OVs ($d > 40$ m and $d > 15$ m on the right and left lane, respectively). Also, Planner 2's lane-change trajectory shows the smallest curvature, which is expected from its lower tolerance for lateral accelerations, and is consistent with Driver 2 being the most cautious of the test subjects. Planner 3 is the least conservative, decelerating later than the others (see velocity and distance for $t < -30$ s) and its trajectory exhibits the highest curvature, which is consistent with Driver 3's performance driving style.

Generalization to other city-driving scenarios (Variant II)

Due to combining data-based—i.e., learning—and model-based—i.e., particle-filter motion planning—approaches, the planners generalize well to scenarios different from the training track. For example, the motion

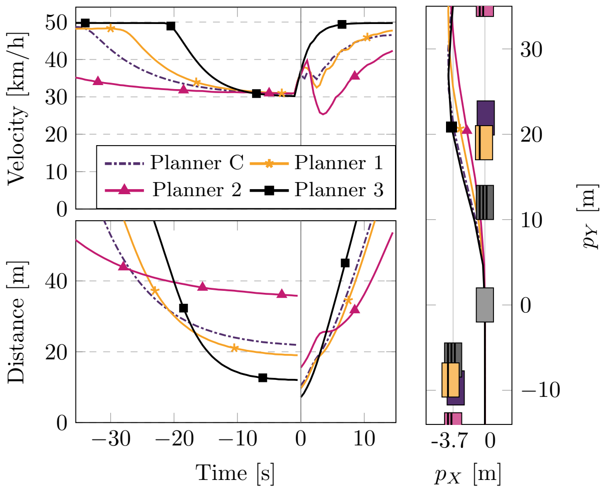


Figure P4-8. Path planning in traffic. Top left: Velocity profiles. Bottom left: Distance from OVs in target lane. Right: Lane-change trajectory with OVs frozen at time of decision, where the OVs' positions for Planner x are marked with x stripes.

planners are able to execute turns with different radii and adjust their velocities accordingly, even though the training data only cover turns with one radius. In order to show the proposed method's generalization properties, we consider circular tracks with different radii. Table P4-5 reports the mean of the velocities of Planner C and 1–5 on the circular track. It shows that all planners decrease their velocities with decreasing radius, which appears natural for human drivers. Specifically, Planner 2 and 5, which generally avoid higher lateral accelerations, slow down the most.

11.2 Discussion

The motion planners trained with both Variant II and Variant III achieve a personalized driving style. Variant II offers the advantage of increased personalization compared to Variant III, however, Variant III is an attractive solution as the hardware requirements are very limited. Due to combining data-based design and model-based algorithm, the motion planners exhibit similarities as well as individual components.

Table P4-5. Circular Track - Averaged Velocities

Radius	500 m	100 m	50 m	25 m
Planner C	46.4 km/h	41.0 km/h	35.6 km/h	28.3 km/h
Planner 1	46.0 km h	36.4 km/h	29.7 km/h	23.2 km/h
Planner 2	40.0 km h	26.0 km/h	19.4 km/h	13.8 km/h
Planner 3	49.7 km/h	49.3 km/h	49.0 km/h	48.2 km/h
Planner 4	44.6 km h	34.5 km/h	27.7 km/h	20.8 km/h
Planner 5	44.3 km h	34.1 km/h	27.5 km/h	20.7 km/h

Similarities of planners & deviation from drivers

The similarities of the planners and deviations from their respective drivers are mainly caused by reasons related to increased safety and fulfillment of requirements. The fulfillment of requirements results in consistency of the planners, which can be seen for instance by the low variance of the planners and the constant velocities during the turn in Fig. P4-6 and Fig. P4-7. Further, the motion planners avoid to exceed the nominal velocity and achieve a relatively small centerline tracking error. These are examples where the planners are not implemented to imitate the driving style of the human entirely, but to achieve a more natural and personalized driving style.

Individuality of planners

In the traffic-free driving scenario in Fig. P4-7, the individuality of the planners can be best identified in the velocity profile and its resulting lateral acceleration. Planner 1, 2, 4, and 5 trained with Variant II take the turn at 34 km/h, 25 km/h, 29 km/h, and 29 km/h with lateral accelerations of 1.8 m/s^2, 1.0 m/s^2, 1.3 m/s^2, and 1.3 m/s^2, which matches their respective drivers' velocities very closely. Planner 3 turns at a slightly higher velocity than Driver 3 in the experiments, however, due to the optimization in the planning algorithm, the velocity is more constant during the turn and hence Planner 3 exhibits lower lateral accelerations than Driver 3, thereby fulfilling both requirements more closely. Also, the traffic-affected scenario in Fig. P4-8 shows highly individual components. Planner 3's velocity is monotonically increasing during the overtake maneuver, which means that longitudinal accelerations continue even during steering operation, as Driver 3 is comfortable with combined longitudinal-lateral accelerations. Planner C, 1, and 2 exhibit a brief drop in velocity for time > 0 s. This drop appears at the peak curvature of the path when the EV turns right to align with the left lane and is caused by the tolerance for lateral accelerations.

Expected limitation of estimated parameters

For significantly different driving scenarios, both the drivers and the planners may behave differently. For instance, in high-speed freeway driving, lane-change maneuvers may be slower (higher r_{11}) and/or velocities more constant (smaller r_{22}). This might prompt mode-dependent parameter sets for each planner, which will be addressed in future work.

12 Conclusion

This paper presented an inverse learning method to calibrate/personalize autonomous vehicles from data of human driving. It proposed a deterministic requirement function with a priori unknown parameters and an algorithm for their estimation. Further, it presented a likelihood maximization method to estimate the probability distribution defining tolerated deviations from the requirements. Three variants of the proposed inverse learning algorithm were presented that vary in computational complexity and storage requirements, as well as their level of approximation, making the inverse learning method adjustable to the available hardware. Simulations with five drivers showed that the estimates are different for each individual, thus resulting in the motion planner generating different motions that, while satisfying the intrinsic properties of the planning algorithm, had a behavior similar to the corresponding drivers.

Bibliography

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles”, *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016.

[2] E. Frazzoli, M. A. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles”, *J. Guid., Control, and Dyn.*, vol. 25, no. 1, pp. 116–129, 2002.

[3] J. Elander, R. West, and D. French, “Behavioral correlates of individual differences in road-traffic crash risk: An examination of methods and findings.”, *Psychological bulletin*, vol. 113, no. 2, pp. 279–294, 1993.

[4] T. Lajunen and T. Özkan, “Self-report instruments and methods”, in *Handbook of traffic psychology*, 2011, pp. 43–59.

[5] F. Sagberg, Selpi, G. F. Bianchi Piccinini, and J. Engström, “A review of research on driving styles and road safety”, *Human factors*, vol. 57, no. 7, pp. 1248–1275, 2015.

[6] K. Berntorp and S. Di Cairano, “Particle filtering for online motion planning with task specifications”, in *Amer. Control Conf.*, 2016, pp. 2123–2128.

[7] K. Berntorp, T. Hoang, and S. Di Cairano, “Motion planning of autonomous road vehicles by particle filtering”, *IEEE Trans. Intell. Vehicles*, vol. 4, no. 2, pp. 197–210, 2019.

[8] SAE, “Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems”, *Standard No. J3016*, Jan. 2014.

[9] M. Menner, K. Berntorp, M. N. Zeilinger, and S. Di Cairano, “Inverse learning for human-adaptive motion planning”, in *58th IEEE Conf. Decision and Control*, 2019, pp. 809–815.

[10] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration”, *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 297–330, 2020.

[11] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control”, *Annu. Rev. Control, Robot., and Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[12] R. E. Kalman, “When is a linear control system optimal?”, *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[13] M. Menner and M. N. Zeilinger, “Convex formulations and algebraic solutions for linear quadratic inverse optimal control problems”, in *Eur. Control Conf.*, 2018, pp. 2107–2112.

[14] K. Mombaur, A. Truong, and J.-P. Laumond, “From human to humanoid locomotion—an inverse optimal control approach”, *Auton. Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[15] D. Clever, R. M. Schemschat, M. L. Felis, and K. Mombaur, “Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground”, in *6th IEEE Int. Conf. Biomed. Robot. and Biomechatronics*, 2016, pp. 1192–1199.

[16] J. F.-S. Lin, V. Bonnet, A. M. Panchea, N. Ramdani, G. Venture, and D. Kulić, “Human motion segmentation using cost weights recovered from inverse optimal control”, in *IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 1107–1113.

[17] A. L. E. N. Kleesattel and K. Mombaur, “Inverse optimal control based enhancement of sprinting motion analysis with and without running-specific prostheses”, in *7th IEEE Int. Conf. Biomed. Robot. and Biomechatronics*, 2018, pp. 556–562.

[18] M. Menner, P. Worsnop, and M. N. Zeilinger, “Constrained inverse optimal control with application to a human manipulation task”, *IEEE Trans. Control Syst. Technol.*, 2019. DOI: 10.1109/TCST.2019.2955663.

[19] P. Englert, N. A. Vien, and M. Toussaint, “Inverse KKT: Learning cost functions of manipulation tasks from demonstrations”, *Int. J. Robot. Res.*, vol. 36, no. 13–14, pp. 1474–1488, 2017.

[20] P. M. Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn, “Data-driven inverse optimization with imperfect information”, *Math. Program.*, vol. 167, no. 1, pp. 191–234, 2018.

[21] A. Majumdar, S. Singh, A. Mandlekar, and M. Pavone, “Risk-sensitive inverse reinforcement learning via coherent risk models”, in *Robot.: Sci. and Syst.*, 2017.

[22] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, “Multimodal probabilistic model-based planning for human-robot interaction”, in *IEEE Int. Conf. Robot. and Automat.*, 2018, pp. 1–9.

[23] R. Chen, W. Wang, Z. Zhao, and D. Zhao, “Active learning for risk-sensitive inverse reinforcement learning”, *ArXiv:1909.07843*, 2019.

[24] G. Chou, N. Ozay, and D. Berenson, “Learning constraints from locally-optimal demonstrations under cost function uncertainty”, *IEEE Robot. and Automat. Lett.*, 2020. DOI: 10.1109/LRA.2020.2974427.

[25] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, “Movement segmentation using a primitive library”, in *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2011, pp. 3407–3412.

[26] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, “Probabilistic segmentation applied to an assembly task”, in *IEEE-RAS 15th Int. Conf. Humanoid Robots (Humanoids)*, 2015, pp. 533–540.

[27] A. Baisero, Y. Mollard, M. Lopes, M. Toussaint, and I. Lütkebohle, “Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations”, in *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2015, pp. 478–484.

[28] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning”, in *21st Int. Conf. Mach. Learning*, 2004, p. 1.

[29] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning", in *23rd AAAI Conf. Artif. Intell.*, vol. 8, 2008, pp. 1433–1438.

[30] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration", *Robot. and Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[31] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with Gaussian processes", in *Adv. Neural Inform. Process. Syst.*, 2011, pp. 19–27.

[32] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples", in *29th Int. Conf. Mach. Learning*, 2012, pp. 475–482.

[33] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning", in *Adv. Neural Inform. Process. Syst.*, 2016, pp. 3909–3917.

[34] K. Bogert, J. F.-S. Lin, P. Doshi, and D. Kulić, "Expectation-maximization for inverse reinforcement learning with hidden data", in *Int. Conf. Auton. Agents & Multiagent Syst.*, 2016, pp. 1034–1042.

[35] V. Joukov and D. Kulić, "Gaussian process based model predictive controller for imitation learning", in *IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 850–855.

[36] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization", in *33rd Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[37] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving", in *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2019, pp. 2658–2665.

[38] S. Schaal, "Is imitation learning the route to humanoid robots?", *Trends in Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[39] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation", *Philos. Trans. Roy. Soc. London. Ser. B: Biol. Sci.*, vol. 358, no. 1431, pp. 537–547, 2003.

[40] W. Wang, J. Xi, A. Chong, and L. Li, "Driving style classification using a semisupervised support vector machine", *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 5, pp. 650–660, 2017.

[41] C. Basu, M. Singhal, and A. D. Dragan, “Learning from richer human guidance: Augmenting comparison-based learning with feature queries”, in *ACM/IEEE Int. Conf. Human-Robot Interaction*, 2018, pp. 132–140.

[42] M. Menner, L. Neuner, L. Lünenburger, and M. N. Zeilinger, “Using human ratings for feedback control: A supervised learning approach with application to rehabilitation robotics”, vol. 36, no. 3, pp. 789–801, 2020.

[43] A. Einstein *et al.*, “The foundation of the general theory of relativity”, *Annalen der Physik*, vol. 49, no. 7, pp. 769–822, 1916.

[44] K. B. Petersen and M. S. Pedersen, *The matrix cookbook (version: Nov. 15, 2012)*, 2012.

[45] S. Di Cairano and I. V. Kolmanovsky, “Real-time optimization and model predictive control for aerospace and automotive applications”, in *Amer. Control Conf.*, 2018, pp. 2392–2409.

[46] A. G. Akritas, E. K. Akritas, and G. I. Malaschonok, “Various proofs of Sylvester’s (determinant) identity”, *Math. and Comput. in Simul.*, vol. 42, no. 4–6, pp. 585–593, 1996.

[47] P. S. Dwyer, “Some applications of matrix derivatives in multivariate analysis”, *J. Amer. Statist. Assoc.*, vol. 62, no. 318, pp. 607–625, 1967.

[48] R. Adamczak, A. Litvak, A. Pajor, and N. Tomczak-Jaegermann, “Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles”, *J. Amer. Math. Soc.*, vol. 23, no. 2, pp. 535–561, 2010.

[49] R. Vershynin, “How close is the sample covariance matrix to the actual covariance matrix?”, *J. Theor. Probability*, vol. 25, no. 3, pp. 655–686, 2012.

[50] C. M. Bishop, *Pattern Recognition and Machine Learning.* Secaucus, NJ: Springer, 2006.

[51] L. Armijo, “Minimization of functions having Lipschitz continuous first partial derivatives”, *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[52] N. C. Schwertman and D. M. Allen, “Smoothing an indefinite variance-covariance matrix”, *Journal of Statistical Computation and Simulation*, vol. 9, no. 3, pp. 183–194, 1979.

[53] N. J. Higham, “Computing a nearest symmetric positive semidefinite matrix”, *Linear algebra and its applications*, vol. 103, pp. 103–118, 1988.

[54] R. S. Rice, “Measuring car-driver interaction with the gg diagram”, SAE Technical Paper, Tech. Rep., 1973.

[55] M. Lavielle, “Using penalized contrasts for the change-point problem”, *Signal Process.*, vol. 85, no. 8, pp. 1501–1510, 2005.

[56] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, “Automated driving: The role of forecasts and uncertainty - a control perspective”, *Eur. J. Control*, vol. 24, pp. 14–32, 2015.

[57] F. Gustafsson, “Particle filter theory and practice with positioning applications”, *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 53–82, 2010.