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

Menner, Marcel; Berntorp, Karl; Di Cairano, Stefano

TR2020-106 July 16, 2020

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.

Transactions on Intelligent Vehicles

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

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

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

***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.**

***Index Terms*—Machine Learning, Intelligent Control, Autonomous Vehicles**

I. 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 pure 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.

II. 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.

Manuscript received November 14, 2019; revised April 01, 2020; accepted May 30, 2020. M. Menner was partially supported by the Swiss National Science Foundation under Grant PP00P2_157601/1. *(Corresponding author: Marcel Menner.)*

M. Menner and M. N. Zeilinger are with the Institute for Dynamic Systems and Control, ETH Zurich, 8092 Zurich, Switzerland (e-mail: mmenner@ethz.ch; mzeilinger@ethz.ch). M. Menner was an intern at Mitsubishi Electric Research Laboratories (MERL) for part of this work.

K. Berntorp and S. Di Cairano are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, 02139, USA (e-mail: berntorp@merl.com; dicairano@merl.com).

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 pure 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. 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.

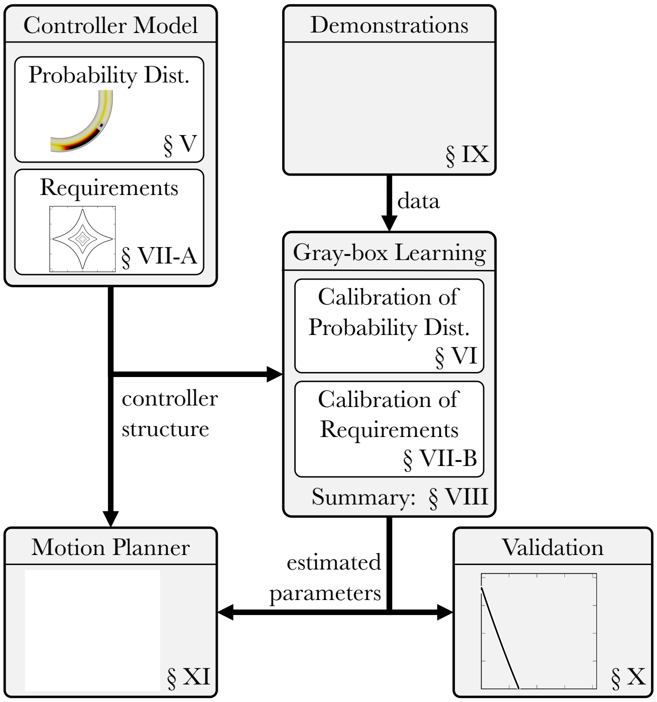


Fig. 1. Core components, their connections in the learning procedure, and their allocation in the paper. The data provided by human drivers are used to calibrate the parameters of the controller. Due to the gray-box learning approach, the estimated parameters are interpretable and thus, can be validated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

III. 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].

A. *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.

B. 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.

C. 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.

IV. 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

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

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 $\mathrm{vec}(\boldsymbol{Z}) = [z_{11}...z_{n1}, z_{12}...z_{n2}, ..., z_{1m}...z_{nm}]^{\mathrm{T}}$. $\boldsymbol{D} = \mathrm{diag}(x_1, ...x_n)$ 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 $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{x}$.

Gradient of a PDF: We use Einstein's summation convention [43] for conciseness, e.g., for $\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_3}$, $\boldsymbol{B} \in \mathbb{R}^{n_1 \times n_2}$, $\boldsymbol{C} \in \mathbb{R}^{n_2 \times n_3}$, $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ 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 $\boldsymbol{x}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{x} = \mathrm{trace}(\boldsymbol{\Sigma}^{-1} \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})$,

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

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 $\boldsymbol{\Sigma}_k$,

$$\begin{aligned} \left(\frac{\partial \log \prod_{k=1}^{T} p(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{Y}}\right)_{ab} &= \sum_{k=1}^{T} \left(\frac{\partial \log p(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{Y}}\right)_{ab} \\ = \sum_{k=1}^{T} \left(\frac{1}{2}\boldsymbol{\Sigma}_k - \frac{1}{2}\boldsymbol{x}_k\boldsymbol{x}_k^{\mathrm{T}}\right)_{zy} \left(\frac{\partial \boldsymbol{\Sigma}_k^{-1}}{\partial \boldsymbol{Y}}\right)_{zyab} &. \qquad (1) \end{aligned}$$

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

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

V. PROBLEM STATEMENT

We consider discrete-time system dynamics of the form

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

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, (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 (3) to ease notation in the following. The behavior of (3) is modeled with respect to requirements $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ with

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

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 (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.

A. 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 (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 (4) are treated as sensor measurements and $\boldsymbol{u}_k$ in (3) is the input (process) disturbance. We model the input disturbance in (3) as Gaussian distributed $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ and the slack in (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,

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

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 $\boldsymbol{H}_k = \partial \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{u}_k)/\partial \boldsymbol{x}|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k}$, $\boldsymbol{D}_k = \partial \boldsymbol{h}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_k, \boldsymbol{u})/\partial \boldsymbol{u}|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_k}$, $\boldsymbol{G}_k = \boldsymbol{g}(\boldsymbol{x}_{k-1})$,

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

where

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

Eq. (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 (6),

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

where the control inputs are

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

with the gain $\boldsymbol{K}_k$ in (6d) and $\boldsymbol{\sigma}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_k)$ with $\boldsymbol{\Sigma}_k$ in (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 (8), in [7], a particle filter extracts the motion plans from (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

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

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 (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.

B. 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 (3)*, the requirement function* (4) *with known parameters* $\boldsymbol{\theta}$*, how motion plans are generated* (7), (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* (8) *that (at least locally) maximize* $p(\boldsymbol{Q}, \boldsymbol{R}|\boldsymbol{Q}^c, \boldsymbol{R}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T})$.

Problem 2: Given motion model (3)*, the parametric requirement function* (4)*, the assumption on the motion plans* (7), (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 VI and Section VII, respectively. Section VIII summarizes the learning procedure and presents three variants of the proposed method and their hardware requirements. Fig. 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.

VI. 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 (8).

A. Optimization Problem Setup

We estimate the covariance matrices from the distribution

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

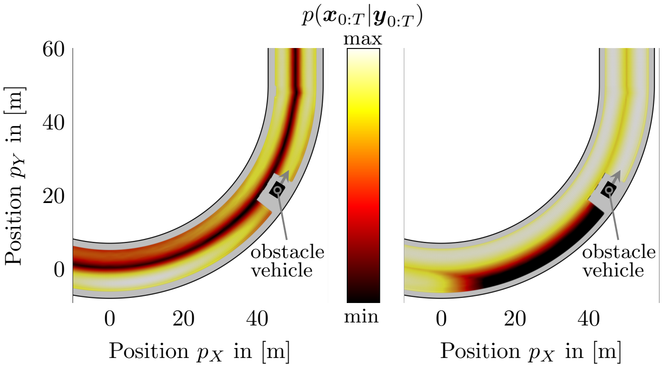


Fig. 2. Illustration of two driver profiles and their probability distribution. Left: Probability distribution resulting from $\boldsymbol{Q}^c, \boldsymbol{R}^c$, and parameters $\boldsymbol{\theta}^c$. Right: Probability distribution from a personalized motion planner. The personalized motion planner 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).

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 (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 V-A being generated by the inputs in (8) rather than by $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$.

1) Likelihood $p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R})$*:* Consider the system dynamics (3) and the measurement (requirement) equation (4). If the control inputs are as in (8), then

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

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 (10) is essential for the efficiency of the likelihood maximization algorithm detailed in Section VI-B.

Theorem 1: Consider (3) and let $p(\boldsymbol{x}_0, \boldsymbol{y}_0) = 1$. If $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ in (4) and $\boldsymbol{u}_k$ as in (8) with $\boldsymbol{K}_k$ as in (6d) and $\boldsymbol{\sigma}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_k)$, then (10) holds.

Proof: The joint probability on the left hand side of (10) is equal to the product of the probabilities of all random variables (Markov property),

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R}) = \prod_{k=1}^{T} p(\boldsymbol{u}_k|\hat{\boldsymbol{u}}_k, \boldsymbol{\Sigma}_k) p(\boldsymbol{v}_k|\boldsymbol{0}, \boldsymbol{R}). \tag{11}$$

Reformulating (11) using Lemma 1 shows (10). ■

Lemma 1: Let $\boldsymbol{u}_k$ as in (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 $\boldsymbol{K}_k^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{K}_k = \boldsymbol{R}^{-1} - \boldsymbol{\Gamma}_k^{-1}$ and $\boldsymbol{\Sigma}_k^{-1}\boldsymbol{K}_k = \boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{R}^{-1}$ and (ii) follows from:

$$\begin{aligned}|\boldsymbol{\Sigma}_k| &= |\boldsymbol{Q}||\boldsymbol{I}_{n_u} - \boldsymbol{K}_k\boldsymbol{J}_k| = |\boldsymbol{Q}||\boldsymbol{I}_{n_u} - \boldsymbol{Q}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{\Gamma}_k^{-1}\boldsymbol{J}_k| \\ &= |\boldsymbol{Q}||\boldsymbol{I}_{n_y} - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{\Gamma}_k^{-1}|,\end{aligned}$$

where the last equality is Sylvester's determinant identity [46]. Finally,

$$|\boldsymbol{I}_{n_y} - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{\Gamma}_k^{-1}| = |\boldsymbol{\Gamma}_k - \boldsymbol{J}_k\boldsymbol{Q}\boldsymbol{J}_k^{\mathrm{T}}||\boldsymbol{\Gamma}_k^{-1}| = |\boldsymbol{R}||\boldsymbol{\Gamma}_k^{-1}|$$

shows (ii). ■

Corollary 1: Let the data be generated by purely stochastic control inputs $\boldsymbol{u}_k = \boldsymbol{\sigma}_k$. Then, the sample covariances $\boldsymbol{Q} = 1/T\sum_{k=1}^{T}\boldsymbol{u}_k\boldsymbol{u}_k^{\mathrm{T}}$ and $\boldsymbol{R} = 1/T\sum_{k=1}^{T}\boldsymbol{v}_k\boldsymbol{v}_k^{\mathrm{T}}$ 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 $\hat{\boldsymbol{\Sigma}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{x}_k\boldsymbol{x}_k^{\mathrm{T}}$ 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 (8), a convergence analysis is more challenging and omitted here as the resulting distribution in (10) is not Gaussian.

2) 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,

$$p\left(\boldsymbol{t}_c\left(\boldsymbol{Q}, \boldsymbol{R}\right)\middle|\boldsymbol{0}, \sigma_c^2\boldsymbol{I}\right) p\left(\boldsymbol{t}_s\left(\boldsymbol{Q}, \boldsymbol{R}\right)\middle|\boldsymbol{0}, \sigma_s^2\boldsymbol{I}\right), \tag{12}$$

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.

3) Logarithmic likelihood maximization: Overall, we estimate $\boldsymbol{Q}$ and $\boldsymbol{R}$ by maximizing the log-likelihood,

$$\begin{aligned}\max_{\boldsymbol{Q},\boldsymbol{R}} \quad & \log\left(p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R})p(\boldsymbol{Q}, \boldsymbol{R}|\boldsymbol{Q}^c, \boldsymbol{R}^c)\right) && \text{(13a)} \\ \text{s.t.} \quad & \boldsymbol{Q} \in \mathcal{C}_Q, \boldsymbol{R} \in \mathcal{C}_R, && \text{(13b)}\end{aligned}$$

where $\mathcal{C}_Q, \mathcal{C}_R$ can be used to enforce constraints on $\boldsymbol{Q}, \boldsymbol{R}$, e.g., $\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{T}} \succeq \boldsymbol{0}, \boldsymbol{R} = \boldsymbol{R}^{\mathrm{T}} \succeq \boldsymbol{0}$. We optimize (13) with the projected gradient method outlined in Section VI-B.

B. Optimization Algorithm: Projected Gradient Descent

Let $\boldsymbol{\xi}^i = [\mathrm{vec}(\boldsymbol{Q}^i)^{\mathrm{T}} \ \mathrm{vec}(\boldsymbol{R}^i)^{\mathrm{T}}]^{\mathrm{T}}$ as the vectorized representation of the covariance matrices at iteration i of the optimization algorithm. Further, let

$$c(\boldsymbol{\xi}^i) = \log\left(p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R})p(\boldsymbol{Q}, \boldsymbol{R}|\boldsymbol{Q}^c, \boldsymbol{R}^c)\right) \tag{14}$$

be the log-likelihood as in (13a). Algorithm 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 $\mathrm{d}\boldsymbol{Q}$ and $\mathrm{d}\boldsymbol{R}$, compute the step size l (Line 4–6), and project the updated matrices (Line 7) onto the constraint set in (13b) (Line 8).

1) Gradient computation: The gradient of the prior belief $\log p(\boldsymbol{Q}, \boldsymbol{R}|\boldsymbol{Q}^c, \boldsymbol{R}^c)$ in (14) depends on the functions $\boldsymbol{t}_c$ and $\boldsymbol{t}_s$ in (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 (10),

$$\begin{aligned}&\log\, p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}|\boldsymbol{Q}, \boldsymbol{R}) = \\ &\quad \sum_{k=1}^{T}\log p(\boldsymbol{u}_k|\boldsymbol{0}, \boldsymbol{Q}) + \sum_{k=1}^{T}\log p(\boldsymbol{v}_k|\boldsymbol{0}, \boldsymbol{R}) \\ &\quad + \sum_{k=1}^{T}\log p(\boldsymbol{w}_k|\boldsymbol{0}, \boldsymbol{R}) - \sum_{k=1}^{T}\log p(\boldsymbol{e}_k|\boldsymbol{0}, \boldsymbol{\Gamma}_k).\end{aligned} \tag{15}$$

The gradient of the first three terms in (15) is computationally cheap to evaluate as it does not scale with the time duration of the collected data T, cf., (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., (1), as the covariance matrix $\boldsymbol{\Gamma}_k$ is time-varying.

2) 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 (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 $i+1$.

3) 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],

$$\boldsymbol{X} = \boldsymbol{V}\mathrm{diag}(\lambda_1, ..., \lambda_n)\boldsymbol{V}^{\mathrm{T}},$$

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

$$\mathrm{proj}_{\boldsymbol{S}_n^+}(\boldsymbol{X}) = \boldsymbol{V}\mathrm{diag}(\max(\varepsilon_\lambda, \lambda_1), ..., \max(\varepsilon_\lambda, \lambda_n))\boldsymbol{V}^{\mathrm{T}}$$

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 X-A.

VII. Requirements for Autonomous Driving and their Personalization

The requirements for the motion-planning application are

- to follow the centerline of a target lane,

Algorithm 1 Estimation of $\boldsymbol{Q}$, $\boldsymbol{R}$

```
$\boldsymbol{Q}^0 = \boldsymbol{Q}^c$, $\boldsymbol{R}^0 = \boldsymbol{R}^c$, $l = 1$, $i = 0$
while $l \geq \varepsilon$                                  ▷ $\varepsilon = 10^{-8}$ in our case
    $\mathrm{d}\boldsymbol{Q}, \mathrm{d}\boldsymbol{R} \leftarrow$ getGrad
    while $c(\boldsymbol{\xi}^i) - c(\boldsymbol{\xi}^i + l\nabla_{\boldsymbol{\xi}} c(\boldsymbol{\xi}^i)) > -\frac{l}{2}|\nabla_{\boldsymbol{\xi}} c(\boldsymbol{\xi}^i)|^2$
        $l \leftarrow \gamma l$                                 ▷ $\gamma = 0.7$ in our case
    end while
    $\boldsymbol{Q} = \boldsymbol{Q}^i + l \cdot \mathrm{d}\boldsymbol{Q}$; $\boldsymbol{R} = \boldsymbol{R}^i + l \cdot \mathrm{d}\boldsymbol{R}$
    $\boldsymbol{Q}^{i+1}, \boldsymbol{R}^{i+1} \leftarrow$ project($\boldsymbol{Q}, \boldsymbol{R}$)
    $l \leftarrow l/\gamma^{n_\alpha}$,   $i \leftarrow i + 1$          ▷ $n_\alpha = 3$ in our case
end while
```

- 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 (4) at time k as are formalized as

$$\boldsymbol{y}_k = \begin{bmatrix} 0 \\ v_{\mathrm{nom}} \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} h^l(p_{X,k}, p_{Y,k}) \\ v_{x,k} \\ h^c_{\boldsymbol{\theta}}(a_{x,k}, a_{y,k}) \\ h^o_{\boldsymbol{\theta}}(d_k, v_{x,k}) \end{bmatrix},$$

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.

A. Individual Requirements

1) 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

$$h^c_{\boldsymbol{\theta}}(a_x, a_y) = \bar{a} \cdot \frac{c_{\boldsymbol{\theta}}(a_x, a_y) - c^0}{c^1 - c^0}, \tag{16}$$

with $c^1 = \left(\sqrt{(\bar{a}^2 + \epsilon)}^{n_c} + \sqrt{\epsilon}^{n_c}\right)^{\frac{1}{n_c}}$, $c^0 = \left(2\sqrt{\epsilon}^{n_c}\right)^{\frac{1}{n_c}}$, 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 (16) such that, for higher n_c, the level sets are more circular, see the left plot in Fig. 3. For $\epsilon = 0$, $s = 1$, (16) is the n_c-(pseudo)norm for $[a_x\ a_y]^{\mathrm{T}}$. 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. 3.

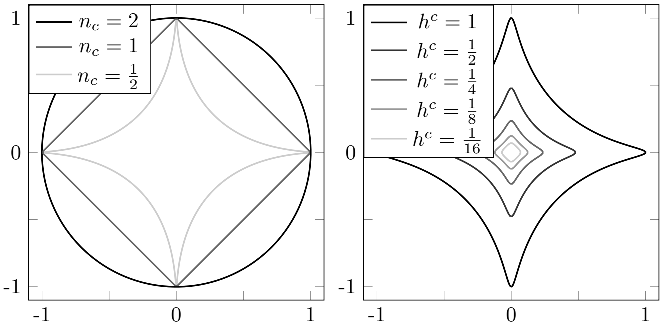


Fig. 3. Left: Level sets $h^c = 1$ with varying n_c, $s = 1$, and $\epsilon = 0$. Right: Varying level sets h^c with $n_c = \frac{1}{2}$, $s = 1$, and $\epsilon = 0.01$.

2) Obstacle avoidance requirement: The obstacle avoidance requirement is modeled as a piecewise linear function

$$h^o_{\boldsymbol{\theta}}(d, v_x) = \begin{cases} \frac{1}{t_s}\left(d_{\min} + t_s v_x - d\right) & \text{if } d_{\min} + t_s v_x \geq d \\ 0 & \text{else,} \end{cases} \tag{17}$$

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].

B. Estimation of Requirement Parameters

We estimate the personalized requirement parameters

$$\boldsymbol{\theta} = \begin{bmatrix} s & n_c & d_{\min} & t_s \end{bmatrix}^{\mathrm{T}}$$

from the distribution $p(\boldsymbol{\theta}|\boldsymbol{\theta}^c, \boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T})$, which we model as $p(\boldsymbol{\theta}|\boldsymbol{\theta}^p, \boldsymbol{I})p(\boldsymbol{\theta}|\boldsymbol{\theta}^c, \sigma^2_\theta \boldsymbol{I})$, with the common parameters $\boldsymbol{\theta}^c = [s^c\ n^c_c\ d^c_{\min}\ t^c_s]^{\mathrm{T}}$, the personal parameters $\boldsymbol{\theta}^p = [s^p\ n^p_c\ d^p_{\min}\ t^p_s]^{\mathrm{T}}$ estimated from $\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:T}$, and the variance σ^2_θ. Then,

$$\boldsymbol{\theta} = \arg\max_{\tilde{\boldsymbol{\theta}}}\ p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^p, \boldsymbol{I})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^c, \sigma^2_\theta \boldsymbol{I}) \tag{18a}$$

$$= \frac{\sigma^2_\theta}{\sigma^2_\theta + 1}\boldsymbol{\theta}^p + \frac{1}{\sigma^2_\theta + 1}\boldsymbol{\theta}^c. \tag{18b}$$

Next, we present algorithms for estimating $\boldsymbol{\theta}^p$ from data of human driving.

1) Estimation of passenger comfort requirement: The estimation of the parameters n_c^p and s^p is achieved by Algorithm 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 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 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 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).

Algorithm 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 $\mathcal{A}_x^\star = \mathrm{setmax}_{|\cdot|}\left(\mathcal{A}_x, M\right)$,
   $\mathcal{A}_y^\star = \mathrm{setmax}_{|\cdot|}\left(\mathcal{A}_y, M\right)$
2: $a_{\max} = \mathrm{med}(A_{\mathrm{c}}^\star)$, $A_{\mathrm{c}}^\star = \mathrm{setmax}_{|\cdot|}\left(\mathcal{A}_x \cup (s \cdot \mathcal{A}_y), M\right)$
3: Choose smallest candidate exponent $n_{c,0}$.
4: $\mathcal{F}_0 = \mathrm{setmax}_{h^c}\left(\mathcal{A}, M\right)$ 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 $\bar{\delta}_0^c \geq a_{\max}$
```

2) 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 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 (17), we want the switch between the two models to coincide with $d_{\min}^p = d - t_s^p v_x$, where $d_{\min}^p < d - t_s^p v_x$ indicates traffic-free and $d_{\min}^p > d - t_s^p v_x$ is traffic-affected driving.

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 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

$$\boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} h^l(p_X, p_Y) & v_x & h_{\boldsymbol{\theta}}^c(a_x, a_y) \end{bmatrix}^{\mathrm{T}} \tag{19a}$$

$$\boldsymbol{H}_k^r = \left.\frac{\partial \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}, \hat{\boldsymbol{u}}_k)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k},\ \boldsymbol{D}_k^r = \left.\frac{\partial \boldsymbol{h}_{\boldsymbol{\theta}}^r(\hat{\boldsymbol{x}}_k, \boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_k} \tag{19b}$$

$$\boldsymbol{v}_k^r = \begin{bmatrix} 0 & v_{\mathrm{nom}} & 0 \end{bmatrix}^{\mathrm{T}} - \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{19c}$$

$$\boldsymbol{e}_k^r = \begin{bmatrix} 0 & v_{\mathrm{nom}} & 0 \end{bmatrix}^{\mathrm{T}} - \boldsymbol{h}_{\boldsymbol{\theta}}^r(\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{0}), \tag{19d}$$

where r denotes reduced (by the obstacle avoidance requirement). If the parameters of $h_{\boldsymbol{\theta}}^c$ are known (Section VII-B1), we can use Algorithm 1 to estimate $\boldsymbol{Q}^r$ and $\boldsymbol{R}^r$ using (19) for traffic-free driving (Line 2 in Algorithm 3). Further, in (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 $\boldsymbol{T} = [\boldsymbol{I}_3\ \boldsymbol{0}_{3\times 1}]$,

$$\boldsymbol{K}_k^r = \boldsymbol{Q}^r \boldsymbol{J}_k^{r\,\mathrm{T}} \left(\boldsymbol{J}_k^r \boldsymbol{Q}^r \boldsymbol{J}_k^{r\,\mathrm{T}} + \boldsymbol{R}^r \right)^{-1},\ \boldsymbol{J}_k^r = \boldsymbol{H}_k^r \boldsymbol{G}_k + \boldsymbol{D}_k^r,$$

and $\boldsymbol{K}_k^{\mathrm{TA}}$ is the residual gain matrix (traffic-affected). This decomposition of $\boldsymbol{K}_k$ in (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

$$t_s^p = \arg \min_{\tilde{t}_s^p, \mathcal{T}, \boldsymbol{a}_i} \sum_{k \in \mathcal{T}} I_k + \sum_{k \notin \mathcal{T}} J_k \tag{20a}$$

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

$$\mathcal{T} = \{ i \mid d_{\min}^p + \tilde{t}_s^p v_{x,i} \leq d_i \}, \tag{20b}$$

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 (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 (20) by enumerating t_s^p in Δt_{inc} increments (Line 3–6).

Algorithm 3 Estimation of $d_{\min}^p$, t_s^p

```
1: $d_{\min}^p = \max(\mathcal{D}^\star)$ with $\mathcal{D}^\star = \mathrm{setmin}_{|\cdot|}\left(\mathcal{D}, M\right)$
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 (20b)
5:     Solve (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 (20a)
```

VIII. 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 V were addressed as follows:

Result 1: Algorithm 1 solves Problem 1.

TABLE I
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	$\mathcal{O}(3n_y^2+n_u^2)$	$\mathcal{O}(n_y^4(n_y^2+n_u^2))$

Result 2: Eq. (18) *using* $\boldsymbol{\theta}^p$ *estimated with Algorithm 2 and Algorithm 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 VI and Section VII. Variant II uses an approximation of the time-varying covariance $\boldsymbol{\Gamma}_k$ as time-invariant $\boldsymbol{\Gamma}$. Variant III uses the same approximation $\boldsymbol{\Gamma}_k \approx \boldsymbol{\Gamma}$ and, additionally, models the parameters of the requirement function as constant $\boldsymbol{\theta} = \boldsymbol{\theta}^c$. Table I specifies the expected data storage and computational complexity for the three variants.

Variant I: The implementation of Variant I requires storage of data that scale linearly with T, where $n_{\mathcal{D}}$ in Table I 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 VI-B1. Algorithm 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 $\boldsymbol{\Gamma}_k \approx \boldsymbol{\Gamma} = (\boldsymbol{H}\boldsymbol{G}+\boldsymbol{D})\boldsymbol{Q}(\boldsymbol{H}\boldsymbol{G}+\boldsymbol{D})^{\mathrm{T}}+\boldsymbol{R}$ with some constant $\boldsymbol{H}$, $\boldsymbol{G}$, and $\boldsymbol{D}$. Algorithm 4 outlines the estimation procedure, where Line 2 and 4 are considerably less computationally demanding.

Variant III: In addition to approximating $\boldsymbol{\Gamma}_k \approx \boldsymbol{\Gamma}$, 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

$$\hat{\boldsymbol{U}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{u}_k\boldsymbol{u}_k^{\mathrm{T}},\ \hat{\boldsymbol{V}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{v}_k\boldsymbol{v}_k^{\mathrm{T}},$$
$$\hat{\boldsymbol{W}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{w}_k\boldsymbol{w}_k^{\mathrm{T}},\ \hat{\boldsymbol{E}}_T = \frac{1}{T}\sum_{k=1}^{T}\boldsymbol{e}_k\boldsymbol{e}_k^{\mathrm{T}}$$

define a sufficient statistic for the distribution in (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.,

$$\hat{\boldsymbol{U}}_T = \frac{1}{T}\boldsymbol{u}_T\boldsymbol{u}_T^{\mathrm{T}} + \frac{T-1}{T}\hat{\boldsymbol{U}}_{T-1}.$$

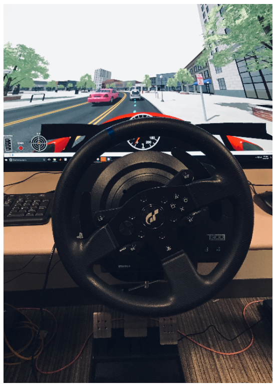


Fig. 4. Simulation setup for learning from data of human driver.

TABLE II
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

For Variant III, the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are estimated as in Line 4 in Algorithm 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 4 Overall estimation procedure for $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{\theta}$

```
Get $s, n_c$ in (18) using $s^p, n_c^p$ estimated with Alg. 2
Get $\boldsymbol{Q}^{\mathrm{TF}}, \boldsymbol{R}^{\mathrm{TF}}$ using Alg. 1 with $\boldsymbol{h}_{\boldsymbol{\theta}}^r$
Get $d_{\min}, t_s$ in (18) using $d_{\min}^p, t_s^p$ estimated with Alg. 3
Get $\boldsymbol{Q}, \boldsymbol{R}$ using Alg. 1 with $\boldsymbol{h}_{\boldsymbol{\theta}}$
```

IX. 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. 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 $(2\cdot200+2\pi\cdot53.6)$ 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.

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 II, 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.

X. 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 III. We present the parameter estimation results in this section and the behavior of the motion planner that uses such parameters in Section XI.

A. 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 $\boldsymbol{Q} = \boldsymbol{Q}^{\mathrm{T}} \succeq \varepsilon_\lambda \cdot \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{R}^{\mathrm{T}} \succeq \varepsilon_\lambda \cdot \boldsymbol{I}$ 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 ($\mathbb{E}[(v_{\mathrm{nom}} - v_x)(0 - h^l)] = 0$).

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 $\|\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^{\mathrm{T}}\|/\|\boldsymbol{R}\|$ with $\boldsymbol{J} = \boldsymbol{H}\boldsymbol{G} + \boldsymbol{D}$ is to be chosen close to one [57]. We choose $\boldsymbol{t}_s(\boldsymbol{Q},\boldsymbol{R}) = \mathrm{vec}(\boldsymbol{J}\boldsymbol{Q}\boldsymbol{J}^{\mathrm{T}} - \boldsymbol{R})$ and $\sigma_s^2 = \frac{1}{T}$ with $\boldsymbol{G} = \boldsymbol{g}(\boldsymbol{x}^\star)$, $\boldsymbol{H} = \partial\boldsymbol{h}(\boldsymbol{x},\mathbf{0})/\partial\boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{x}^\star}$, and $\boldsymbol{D} = \partial\boldsymbol{h}(\boldsymbol{x}^\star,\boldsymbol{u})/\partial\boldsymbol{u}|_{\boldsymbol{u}=\mathbf{0}}$, where $\boldsymbol{x}^\star$ denotes a nominal state where all requirements are fulfilled ($v_x = 50$km/h, $\delta = 0$).

Prior (common belief): We choose $\sigma_c^2 = \sigma_\theta^2$ and $\boldsymbol{t}_c(\boldsymbol{Q},\boldsymbol{R}) = [\mathrm{vec}(\boldsymbol{Q} - \boldsymbol{Q}^c)^{\mathrm{T}}\ \mathrm{vec}(\boldsymbol{R} - \boldsymbol{R}^c)^{\mathrm{T}}]^{\mathrm{T}}$. Then,

$$p\left(\boldsymbol{t}_c(\boldsymbol{Q},\boldsymbol{R})|\mathbf{0},\sigma_c^2\boldsymbol{I}\right) = \prod_{\forall ij} p\left(q_{ij}|q_{ij}^c,\sigma_c^2\right) \prod_{\forall ij} p\left(r_{ij}|r_{ij}^c,\sigma_c^2\right)$$

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,

$$\boldsymbol{Q}^c = \mathrm{diag}(16.1,\ 0.0016) \tag{21a}$$

$$\boldsymbol{R}^c = \begin{bmatrix} 0.105 & 0 & 0 & 0 \\ & 16.2 & -4.10 & -14.7 \\ & & 5.02 & 4.38 \\ & & & 17.4 \end{bmatrix} \tag{21b}$$

$$\boldsymbol{\theta}^c = \begin{bmatrix} s & n_c & d_{\min} & t_s \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 8 & 5 \end{bmatrix}^{\mathrm{T}}. \tag{21c}$$

B. 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 VIII. Fig. 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 $\boldsymbol{\Gamma}_k \approx \boldsymbol{\Gamma}$ 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} ($\mathbb{E}[(v_{\mathrm{nom}} - v_x)(0 - h^o)]$).

Personalized parameters (30 min of driving): Table III 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 $h_{\boldsymbol{\theta}}^c(a_x, a_y) = a_{\max}$ 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

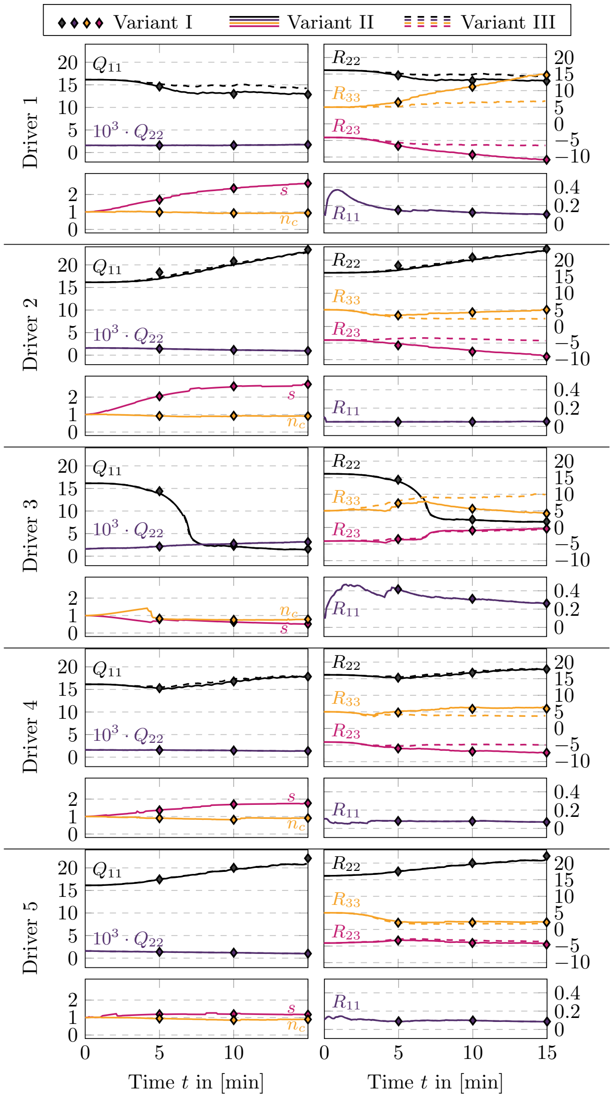
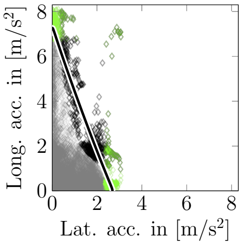
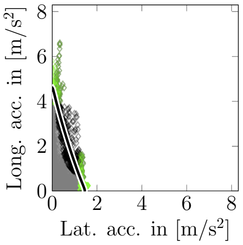
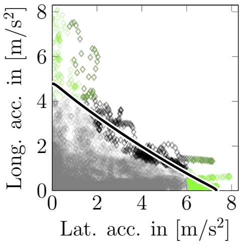
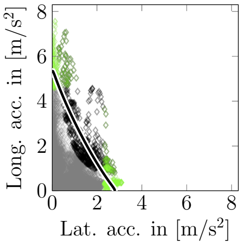
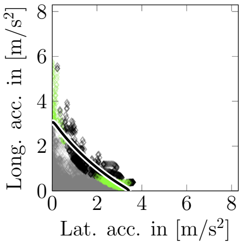


Fig. 5. Parameter estimated as data are collected over time with Variant I (diamond symbols), Variant II (solid lines), and Variant III (dashed lines) for five drivers. The parameters $\boldsymbol{\theta}, \boldsymbol{Q}, \boldsymbol{R}$ are obtained by solving Algorithm 4 using different amount of data from time 0min through time t (x-axis).

TABLE III
Estimated Parameters & Acceleration Scatter Plots

Driver 1 Occasional driver $\boldsymbol{Q} = \mathrm{diag}(13.6,\ 0.0017)$ $\boldsymbol{R} = \begin{bmatrix} 0.085 & 0 & 0 & 0 \\ & 13.1 & -6.39 & -11.4 \\ & & 6.68 & 5.29 \\ & & & 14.3 \end{bmatrix}$ $s = 2.71$, $n_c = 0.95$ $d_{\min} = 7.32$ m, $t_s = 4.10$ s	
Driver 2 Occasional driver $\boldsymbol{Q} = \mathrm{diag}(30.3,\ 0.0009)$ $\boldsymbol{R} = \begin{bmatrix} 0.055 & 0 & 0 & 0 \\ & 27.0 & -5.31 & -24.9 \\ & & 2.69 & 3.94 \\ & & & 28.8 \end{bmatrix}$ $s = 3.17$, $n_c = 0.89$ $d_{\min} = 13.6$ m, $t_s = 7.00$ s	
Driver 3 Experienced driver with professional training $\boldsymbol{Q} = \mathrm{diag}(1.37,\ 0.0032)$ $\boldsymbol{R} = \begin{bmatrix} 0.257 & 0 & 0 & 0 \\ & 1.63 & -0.725 & -0.214 \\ & & 10.1 & 0.0856 \\ & & & 1.54 \end{bmatrix}$ $s = 0.65$, $n_c = 0.9$ $d_{\min} = 5.89$ m, $t_s = 3.21$ s	
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	

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 $r_{24} < 0$ ($\mathbb{E}[(v_{\mathrm{nom}} - v_x)(0 - h^o)] < 0$), 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.

C. Hardware Requirements & Computational Cost

Table IV 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. 5. As Variant II is comparably cheap, we favor Variant II over Variant I and employ only Variant II and Variant III in the following.

TABLE IV
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

XI. 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 (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.

A. Results

Fig. 6 and Fig. 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. 6 compares the autonomous motion planners trained with Variant II and Variant III and Fig. 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 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 decide 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. 6 shows a comparison between Variant II and Variant III for Driver 1 and Driver 2, who tend to avoid higher lateral accelerations compared to longitudinal accelerations ($s > 2$ in (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

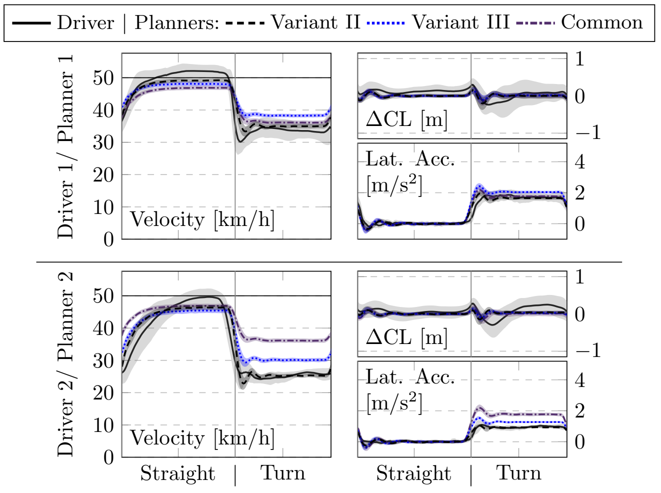


Fig. 6. Comparison of Variant II and Variant III. The velocity, centerline tracking error, and lateral acceleration are displayed as functions of the track position for Driver x (solid black), Planner C (dashdotted purple), Variant II (dashed black), and Variant III (dashed blue).

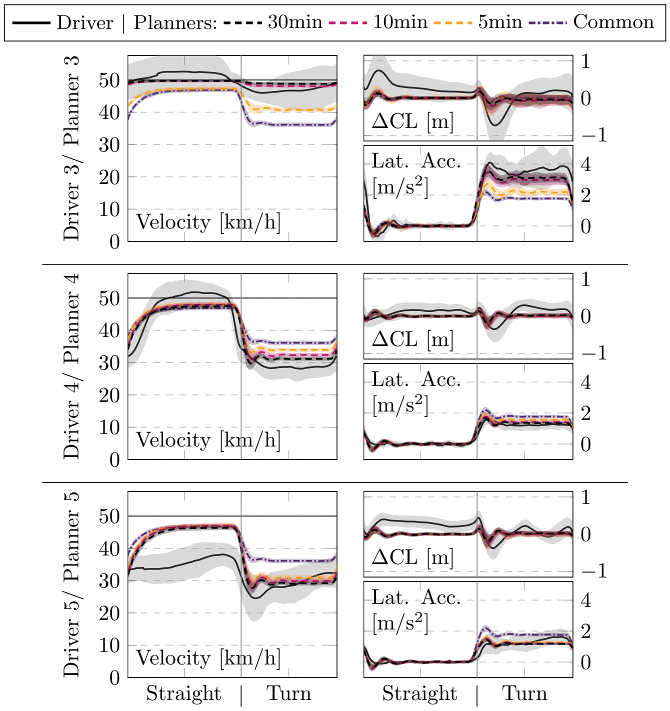


Fig. 7. Personalizing planners over time. The velocity, centerline tracking error, and lateral acceleration are displayed as functions of the track position for Driver x (solid), Planner C (dashdotted purple), Variant II with 5 min of driving data (dashed yellow), Variant II with 10 min of driving data (dashed magenta), and Variant II with 30 min of driving data (dashed black).

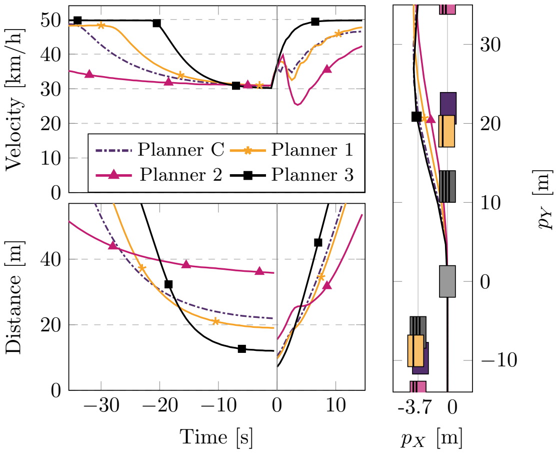


Fig. 8. Path planning in traffic. Top left: Velocity profiles. Bottom left: Distance from OVs in target lane. Right: Lane-change trajectory with OVs frozen at time of decision, where the OVs' positions for Planner x are marked with x stripes.

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. 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. 8 illustrates the resulting trajectories of Planner 1–3 as well as for the planner with the common parameters (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 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 V 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.

TABLE V
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

B. 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.

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. 6 and Fig. 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. 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. 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.

XII. 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.

References

[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, 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, 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. Machine Learning*, 2004, p. 1.

[29] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conf. Artif. Intell.*, 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 *Advances 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. Machine Learning*, 2012, pp. 475–482.

[33] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances 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. Grojohann, 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.*, Nov 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,” *IEEE Trans. Robot.*, 2020, doi: 10.1109/TRO.2020.2964147.

[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*. New York: Springer, 2006.

[51] L. Armijo, “Minimization of functions having lipschitz continuous first partial derivatives,” *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, 1966.

[52] N. C. Schwertman and D. M. Allen, “Smoothing an indefinite variance-covariance matrix,” *J. Statistical Computation and Simul.*, vol. 9, no. 3, pp. 183–194, 1979.

[53] N. J. Higham, “Computing a nearest symmetric positive semidefinite matrix,” *Linear algebra and its appl.*, 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.

Marcel Menner (Student Member, IEEE) received the M.Sc. degree with high distinction in mechanical engineering from the Technische Universität München, Munich, Germany, in 2016, for which he was awarded the Degree of Excellence. He is currently working toward the Ph.D. degree with the Institute for Dynamic Systems and Control, ETH Zurich, Switzerland.

He was a visiting student with the Massachusetts Institute of Technology, Cambridge, MA, USA (2015–2016) and a Research Intern with the Mitsubishi Electric Research Laboratories, Cambridge, MA, USA (2018–2019). His research interests include intelligent systems and data-driven control with a focus on learning from human interactions.

Karl Berntorp (SM’20) received the M.Sc. degree in Engineering Physics in 2009 and the Ph.D. degree in Automatic Control in 2014, from Lund University, Lund, Sweden. In 2008 he was a visiting researcher at Daimler AG in Sindelfingen, Germany. Since 2014 he has worked with Mitsubishi Electric Research Laboratories in Cambridge, MA. His research is on statistical signal processing, Bayesian inference, sensor fusion, and optimization-based control, with applications to automotive, transportation, navigation, and communication systems. His work includes design and implementation of estimation, constrained control, motion-planning, and learning algorithms. Dr. Berntorp is the author of more than 70 papers in journals and conferences and has more than 10 granted patents.

Melanie N. Zeilinger (Member, IEEE) received the Diploma degree in engineering cybernetics from the University of Stuttgart, Germany, in 2006, and the Ph.D. degree with honors in electrical engineering from ETH Zurich, Switzerland, in 2011.

She is an Assistant Professor with the Department of Mechanical and Process Engineering, ETH Zurich, Switzerland. From 2011 to 2012, she was a Postdoctoral Fellow with the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. From 2012 to 2015, she was a Marie Curie Fellow and Postdoctoral Researcher in a joint program with the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, CA, USA, and with the Max Planck Institute for Intelligent Systems, Tübingen, Germany. From 2018 to 2019, she was a Professor with the University of Freiburg, Freiburg im Breisgau, Germany. Her research interests include distributed control and optimization, as well as safe learning-based control, with applications to human-in-the-loop systems.

Stefano Di Cairano (SM’15) received the Master’s (Laurea) and the Ph.D. degrees in information engineering in 2004 and 2008, respectively, from the University of Siena, Italy. During 2008-2011, he was with Powertrain Control R&A, Ford Research and Advanced Engineering, Dearborn, MI, USA. Since 2011, he is with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he is currently the Senior Team Leader for optimization-based control, and a Distinguished Researcher in Control and Dynamical Systems. His research focuses on optimization-based control and decision-making strategies for complex mechatronic systems, in automotive, factory automation, transportation systems, and aerospace. His research interests include model predictive control, constrained control, path planning, hybrid systems, optimization, and particle filtering. He has authored/coauthored more than 150 peer-reviewed papers in journals and conference proceedings and 35 patents.

Dr. Di Cairano was the Chair of the IEEE CSS Technical Committee on Automotive Controls and of the IEEE CSS Standing Committee on Standards. He is the inaugural Chair of the IEEE CCTA Editorial Board and was an Associate Editor of the IEEE TRANS. CONTROL SYSTEMS TECHNOLOGY.