Robotics and Autonomous Systems 114 (2019) 155–171

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Data-driven human driver lateral control models for developing haptic-shared control advanced driver assist systems

Kazuhide Okamoto [a,*], Panagiotis Tsiotras [b]

[a] School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150, USA
[b] School of Aerospace Engineering and the Institute for Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, , GA 30332-0150, USA

HIGHLIGHTS

- A comparative study of machine-learning-based driver control models is conducted.
- We evaluate them in terms of their suitability to develop haptic-shared ADAS.
- A new unknown input estimation algorithm is proposed to estimate steering torque.
- The algorithm is suitable for low-cost driving simulators that are not equipped with torque sensors.

ARTICLE INFO

Article history:
Available online 11 February 2019

Keywords:
Advanced driver assist system
Human driver control model
Driving simulator study
Driver–vehicle interaction
Unknown parameter and input estimation

ABSTRACT

In order to improve road safety, many advanced driver assist systems (ADAS) have been developed to support human-decision making and reduce driver workload. Currently, the majority of ADAS employ a single, often very simple, driver model to predict human-driver interaction in the immediate future (e.g., next few seconds). However, there is tremendous variability in how each individual drives, necessitating personalized driver models, based on data collected from observed actual driver actions. Yet, because we currently lack sufficient knowledge of the high-level cognitive brain functions, traditional control-theoretic driver models have difficulty accurately predicting driver actions. Recently, machine-learning algorithms have been utilized to predict future driver control actions. We compare several of these algorithms used to predict the lateral control actions of human drivers. Specifically, we compare these algorithms in terms of their suitability to develop haptic-shared ADAS, which share the control force with the human driver. To this end, we need to know how the steering torque is provided by the driver. However, low-cost driving simulators typically measure steering angle but not steering torque. Thus, this work also proposes a methodology to estimate the steering-wheel torque. Using the estimated steering torque, we train several machine learning driver control models and compare the performance using both simulated and real human-driving data sets.

1. Introduction

This paper addresses two problems. One deals with comparing the performance of machine-learning-based human driver control models. The second problem deals with estimating the unknown torque input using steering wheels found in many low-cost driving simulators. In this section, we first provide an overview of the previous works on human driver control modeling, and then we review the works on unknown steering-torque-estimation algorithms.

Advanced driver assist systems (ADAS) that support human drivers have been investigated for many years in an effort to increase road safety. In order to properly assist drivers, ADAS need to accurately predict the immediate actions of the drivers, necessitating a thorough investigation of human driver control models. This raises a serious issue since one should consider the combined vehicle–driver model when designing ADAS systems. However, unlike the modeling of the vehicle, it is difficult to reliably model human driver behavior.

The majority of modern control design methodologies are model-based. That is, in order to design a vehicle control system, a good vehicle model is required that, depending on the complexity of the task, ranges from a simple point-mass model to a complex rolling rigid vehicle model [1]. Parameter identification of these models have been well established [2,3]. On the other hand, similar identification techniques are lacking in the domain of the human driver action models. While several control-theory-based models that explicitly mimic high-level cognitive human

* Corresponding author.
E-mail addresses: kazuhide@gatech.edu (K. Okamoto), tsiotras@gatech.edu (P. Tsiotras).

https://doi.org/10.1016/j.robot.2019.01.020
0921-8890/

brain functions have been proposed over the years (e.g., the two-point visual control model [4–6]), it is still very challenging to accurately predict human driver control actions using traditional modeling methods. Recently, by taking advantage of the development of computational resources and available data, machine-learning algorithms have succeeded in many challenging pattern-recognition and inference tasks. The main objective of this paper is to compare the accuracy of the predicted lateral control actions of human drivers in the immediate future using such machine-learning techniques.

Investigations related to the use of machine-learning algorithms to identify the intentions and actions of human drivers have been attempted in the past. In the late '90s, a hidden Markov model (HMM) was employed by Pentland and Liu to predict driver intention [7], and following that work HMMs have been one of the most popular methods for identifying human driver intention [8–10]. Recently, other algorithms have been employed. Aoude et al. [11], for instance, developed two human-driver-intention-identification algorithms, specifically designed to work at road intersections. The first technique utilizes support vector machines (SVM) and a Bayesian filter, and the other technique uses HMMs. The authors of [11] used real-world traffic data and demonstrated that both methods outperform traditional statistical models. Furthermore, a dynamic Bayesian network employed in [12] was able to identify the fatigue level of human drivers. In addition, based on a k-nearest neighbors (kNN) algorithm and an SVM, driving skills while negotiating curves were classified in [13]. In [14,15] machine-learning techniques were used to decrease the computational cost for vehicle-motion prediction in traffic. Furthermore, a random-forest classifier [16] used in [17] was able to classify each human driver using the controller-area-network (CAN) data found in all ADAS-enabled vehicles. The authors of [17] also warned that this information-extraction capability is a potential threat to privacy. Note that all these prior works are based on classification algorithms.

The attempt to learn control inputs from human demonstrations, called learning from demonstration (LfD), has been an active research topic in the robotics community [18]. Several researchers compared the LfD performance. For example, Nguyen-Tuong et al. [19] compared several regression methods to learn the inverse dynamics of a robot arm and concluded that GP regression and support vector regression achieve higher learning accuracy than locally weighted projection regression (LWPR). Several LfD works have been proposed in the vehicle-control area, such as a Gaussian Process (GP) regression method [20] that models human-driver braking actions and Gaussian Mixture Models (GMMs) that models the longitudinal control actions of human drivers [21]. Also, the performance of a GMM-based driver model was compared against a stimulus–response and an optimal velocity models [22]. Furthermore, the authors of [23] utilized the maximum entropy inverse reinforcement learning algorithm [24] to learn driving styles. Lefèvre et al. [25] compared the performance of driver longitudinal control models and observed that simple parametric models exhibited sound predictions for short-term horizons, while, with long-term prediction horizons, non-parametric models, such as HMM with Gaussian mixture regression [26], outperform parametric models. Finally, but equally importantly, Wei et al. [27] compared the performance between receding horizon controller (RHC)-based and artificial neural network (ANN)-based models to predict a professional driver's driving actions and found that the ANN-based model outperforms the RHC-based model. Recently, researchers have utilized these human driver control models to improve the performance of ADAS.

Based on identified driver control models, several newly developed ADAS predict the future behavior of a human driver, and compute the necessary corrections to achieve higher performance than solely-human-driven vehicles. For instance, Di Cairano et al. [28] developed a Markov chain that predicts the future throttle-power request by the human driver. The predicted power request is then fed into a model predictive controller (MPC) as an input. The resulting controller improved the fuel efficiency of a hybrid electrical vehicle. The ADAS developed by Lefèvre et al. [29] controls the longitudinal motion of a vehicle. The control signal is computed so that the signal is similar to the predicted desired action of the human driver and, at the same time, satisfies the safety constraints. In addition, Liu et al. [30] developed an ADAS that corrects the steering angle of the front wheel of the vehicle to increase safety when the human driver is distracted. This ADAS is activated only when it predicts that the human driver will violate the prespecified safety constraints. A similar situation is dealt with by the ADAS proposed Shia et al. [31], which corrects the steering wheel angle only when the safety constraints are predicted to be violated.

The ADAS introduced above share the input to the vehicle system with the human driver. Thus, they are categorized as "input-mixing shared control," which directly changes the input to the controlled system [32]. Driver control models that output steering angles are, for instance, [4,27,33–36]. According to [32], there exist another category, called the "haptic shared control," which shares the input forces with the human operator. Such ADAS are introduced in [37,38], which employ driver control models that output steering torque and compute correction torque based on the predicted torque generated by the human driver control model.

Driver control models that output steering torque have been proposed by several researchers [5,6,39]. According to [40], haptic shared control is more beneficial than input-mixing control, because human drivers can always remain in control and continuously receive feedback from the ADAS. We conjecture that in emergency situations, input-mixing-shared-control-based ADAS is more suitable because the top priority is to avoid accidents, while in other situations, such as lane keeping on highways, the haptic-shared-control approach is more suitable to tailor the system output to the preference of the human driver. Thus, in this work we perform a comparative study of machine-learning-based human driver control models which output steering-wheel torque. However, while the training of such models require torque data, conventional steering wheels for driving simulators are not equipped with torque sensors. We conjecture that this limitation leads to the limited number of available research works on haptic-shared control ADAS compared to input-mixing control ADAS. Thus, in this work we propose an approach to estimate the unknown steering-torque input to the steering wheel of a driving simulator in Section 3.

Many researchers employ driving simulators to verify their algorithms because use of real vehicles involves high operational cost and risk of accidents. Indeed, many realistic, low-cost driving simulators have been developed either with a fixed [39,41,42] or moving base [13,33]. Thanks to the recent development of computer graphics, these simulators offer a realistic experience to the driver. However, these simulators may not be suitable for testing haptic-shared-control-based ADAS since – most often than not – they use steering wheels that measure steering-wheel *angle* but not steering wheel *torque*. This which implies neglecting the interactions between the human-driver and the steering-wheel dynamics. In order to capture this interaction between a human driver and the vehicle, it is imperative to infer the actual driver control command, namely, steering torque, not steering angle [43].

In order to estimate the unknown steering-torque input, we also need to estimate the unknown steering-wheel system parameters. In this work we assume the availability of a steering wheel that: (a) measures steering wheel angle only and not steering wheel torque; and (b) has force-feedback functionality. Note that many real vehicles measure the steering-wheel torque to provide

power steering. Thus, the methods discussed in this paper may not be necessary for cases that employ real vehicles with steering-torque sensors.

A preliminary version of this work was presented in [44]. The current, and more elaborated version of the work in [44] contains a more extensive discussions on the results, and includes a discussion for estimating the delivered driver torque using only steering angle feedback.

The remainder of this paper is organized as follows. Section 2 introduces the methods we employ to model the lateral control actions of human drivers. The output of these methods are predicted values of the human steering torque. Thus, to evaluate the performance of these algorithms, we need the ground-truth of the steering input torque, which is unknown if a steering wheel is not equipped with a torque sensor is used, thus necessitating a suitable estimation algorithm. To this end, Section 3 formulates the unknown steering torque input estimation problem, introduces previous methods, and proposes such an algorithm. Next, in Section 4, we verify the steering-torque-estimation algorithm using numerical simulations and real human driving data. After establishing the soundness of the proposed steering torque input estimation algorithm, in Section 5, we evaluate the performance of the machine-learning methods to model human driver lateral control actions introduced in Section 2 using synthetic data generated from a human control model, as well as real data obtained from a driving simulator driven by a human subject. We discuss the results of these experiments in Section 6.

2. Driver steering torque prediction

We wish to predict the driver's torque at the current time step given some feature values at the current and several previous time steps. This section introduces the feature vector and the machine-learning approaches we use to identify the driver's steering torque.

Note that, while many deep-learning-based approaches such as [45] and [46] have been proposed in the literature, we do not use such approaches in this work, because in order to train a deep neural network, one needs a significant amount of data. For instance, the work in [45] collected 12 hours of simulated driving data, and the work in [46] collected 72 hours of real-world driving data. Obtaining such large amounts of data may be time-consuming and expensive. By properly designing the feature vector, this work shows that we can achieve good prediction performance without the need to collect a massive amount of data.

2.1. Feature vector

This work follows the modeling framework of the sensorimotor two-point visual control model [5], which is depicted in Fig. 1. The system consists of five subsystems: the road geometry, the perception subsystem, the driver, the steering column, and the vehicle dynamics subsystem. The road geometry outputs road curvature ρ. The inputs of the perception subsystem are ρ, side-slip angle β, yaw rate r, and the outputs are two view-ahead angles called θ_{near} and θ_{far}. The driver subsystem processes these two angles to compute the steering torque T_{dr}. As it is difficult to measure θ_{near} and θ_{far}, the perception and the driver subsystems are combined in this work, and we call the combined subsystem the "human driver" subsystem, as shown inside the dashed box in Fig. 1. By introducing this human driver subsystem, we do not have to consider the angles θ_{near} and θ_{far}. Instead, we employ the more easily measurable inputs and outputs to design regression methods that identify the steering torque T_{dr}. The inputs to the human driver subsystem are ρ, β, r and the steering-wheel angle δ_s and the output is T_{dr}. Thus, our objective is to find the relationship between these inputs and the output.

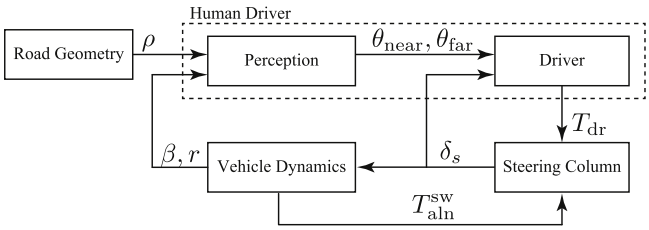


Fig. 1. The sensorimotor two-point visual control model framework.

In order to incorporate sequential changes of the inputs for the last $\ell\Delta t$ s, where $\ell = 1, 2, \ldots$ and $\Delta t > 0$ is the sampling interval, we introduce the following feature vector $z(t_k) \in \mathbb{R}^d$:

$$z(t_k) = [\rho(t_k), \beta(t_k), r(t_k), \delta_s(t_k), T_{\text{dr}}(t_{k-1}), \rho(t_{k-1}), \beta(t_{k-1}), \ldots, \delta_s(t_{k-\ell})]^\top, \tag{1}$$

where $k = 1, 2, \ldots$ is the time-step index. Note that this work assumes that the input variables at the current time step are available, eliminating the modeling error of other subsystems (i.e., vehicle and steering column dynamics). This assumption enables direct comparison of the estimation performance of T_{dr}. We set the time discretization as $\Delta t = 1/\ell$ s so that $z(t_k)$ includes the input variables over the previous time step and the steering torque output at the previous $\Delta t, 2\Delta t, \ldots, \ell\Delta t (= 1)$ s. The objective is to estimate the following nonlinear function $f : \mathbb{R}^d \to \mathbb{R}$

$$T_{\text{dr}}(t_k) = f(z(t_k)). \tag{2}$$

2.2. Piecewise linear models

To benchmark the performance of the machine-learning methods we consider in this work, we quickly review two simple models, namely, Piecewise Constant Model (PCM) and Piecewise Linear Model (PLM). Both models disregard much of the information in $z(t_k)$. We thus expect that the other machine-learning methods outperform these two naïve approaches. Nonetheless, it is imperative to consider these two models in order to find the minimum performance requirements for the machine-learning-based methods.

The PCM assumes that the current driver torque is the same as the driver's torque at the previous time step, namely,

$$T_{\text{dr}}^{\text{PCM}}(t_k) = T_{\text{dr}}(t_{k-1}). \tag{3}$$

This model does not incorporate the other entries in $z(t_k)$ so we expect a good prediction performance only for smooth steering commands and sufficiently small Δt.

The PLM considers the time derivative of T_{dr} as follows

$$T_{\text{dr}}^{\text{PLM}}(t_k) = T_{\text{dr}}(t_{k-1}) + \left.\frac{\mathrm{d}T_{\text{dr}}}{\mathrm{d}t}\right|_{t_{k-1}} \Delta t, \tag{4}$$

where we approximate the time derivative of T_{dr} at $t = t_{k-1}$ as

$$\left.\frac{\mathrm{d}T_{\text{dr}}}{\mathrm{d}t}\right|_{t_{k-1}} \approx \frac{T_{\text{dr}}(t_{k-1}) - T_{\text{dr}}(t_{k-2})}{\Delta t}. \tag{5}$$

With the estimated value of the time derivative of T_{dr} we expect that this model exhibits a more accurate prediction than PCM, as long as the steering command is smooth and Δt is sufficiently small. Models based on higher-order derivatives of T_{dr}, e.g., the second derivative of T_{dr} are also possible. However, since our research interest lies in the evaluation of more sophisticated machine-learning-based algorithms, the previous two PCM and PLM models are sufficient for our purposes.

2.3. Gaussian process regression

In this section we briefly summarize Gaussian Process (GP) regression and how it is used for our problem. For a more detailed discussion, we refer the reader to [20]. A GP is a non-parametric kernel-based method represented as a collection of random variables, any finite number of which have a joint Gaussian distribution.

The training of a GP regression assumes an independent, identically distributed (i.i.d.) Gaussian observation noise ϵ

$$T_{\mathrm{dr}} = f(z) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{6}$$

We let $\mathcal{D} = \{z(t_k), T_{\mathrm{dr}}(t_k)\}_{k=1}^{N_D}$ denote the training data set. For simplicity, this section omits the argument t_k, and we let z be $z(t_k)$ and z' be $z(t_{k'})$, where $k, k' \in \{1, \ldots, N_D\}$ is the index for the training data set.

The mean $m(z)$ and covariance functions $\bar{k}(z, z')$ specifies a GP as follows

$$m(z) = \mathbb{E}[f(z)], \tag{7}$$

$$\bar{k}(z, z') = \mathbb{E}[(f(z) - m(z))(f(z') - m(z'))], \tag{8}$$

and hence a GP is often written as $f(z) \sim \mathcal{GP}\left(m(z), \bar{k}(z, z')\right)$. Here, for simplicity, we describe the training and prediction of zero-mean GP that is, $\mathcal{GP}(0, \bar{k}(z, z'))$. The kernel we employ is the exponential kernel covariance function

$$\bar{k}(z, z'|\sigma_\ell, \sigma_f) = \sigma_f^2 \exp\left(-\frac{r(z, z')}{\sigma_\ell}\right), \tag{9}$$

where σ_ℓ is the characteristic length scale, σ_f is the signal standard deviation, and $r(z, z')$ is the Euclidean distance between z and z'. The prior covariance matrix of the observations with noise is

$$K_{T_{\mathrm{dr}}}(Z, Z) = K(Z, Z) + \sigma_n^2 I, \tag{10}$$

where $Z = [z(t_1), \ldots, z(t_{N_D})]$, and $[K(Z, Z)]_{k,k'} = \bar{k}(z, z')$. The hyper-parameters σ_ℓ, σ_f, and σ_n are tuned during training.

The GP regression estimates the response $T_{\mathrm{dr}*}$, given a new input data set Z_*, where we assume a normal distribution $\Pr(T_{\mathrm{dr}*}|Z_*, \mathcal{D})$. Because the joint distribution of the observed target and function values at test locations under the above prior is

$$\begin{bmatrix} T_{\mathrm{dr}} \\ T_{\mathrm{dr}*} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{T_{\mathrm{dr}}}(Z, Z) & K(Z, Z_*) \\ K(Z_*, Z) & K(Z_*, Z_*) \end{bmatrix}\right), \tag{11}$$

the following conditional probability distribution of the function values at the test locations hold

$$\Pr(T_{\mathrm{dr}*}|Z_*, \mathcal{D}) \sim \mathcal{N}(\mathrm{mean}(T_{\mathrm{dr}*}), \mathrm{cov}(T_{\mathrm{dr}*})), \tag{12}$$

where

$$\mathrm{mean}(T_{\mathrm{dr}*}) = \mathbb{E}[T_{\mathrm{dr}*}|Z, T_{\mathrm{dr}}, Z_*] \tag{13}$$

$$= K(Z_*, Z)K_{T_{\mathrm{dr}}}^{-1}(Z, Z)Y, \tag{14}$$

$$\mathrm{cov}(T_{\mathrm{dr}*}) = K(Z_*, Z_*) - K(Z_*, Z)K_{T_{\mathrm{dr}}}^{-1}(Z, Z)K(Z, Z_*), \tag{15}$$

and $Y = [T_{\mathrm{dr}}(t_1), \ldots, T_{\mathrm{dr}}(t_{N_D})]^\top$. Thus, given a new feature vector $z(t_k)$, the GP regression predicts the driver torque as

$$T_{\mathrm{dr}}^{\mathrm{GP}}(t_k) = K(z(t_k), Z)K_{T_{\mathrm{dr}}}^{-1}(Z, Z)Y. \tag{16}$$

Letting $\theta = (\sigma_\ell, \sigma_f, \sigma_n) \in \Theta$, the parameter vector θ is tuned during the training process of the GP regression model, such that

$$\theta^* = \underset{\theta \in \Theta}{\mathrm{argmax}} \Pr(Y|Z, \theta), \tag{17}$$

where

$$\Pr(Y|Z, \theta) = \mathcal{N}(0, K_{T_{\mathrm{dr}}}(Z, Z)). \tag{18}$$

In order to compute θ^*, we take the log of the conditional distribution in Eq. (18). Thus,

$$\log \Pr(Y|Z, \theta) = -\frac{1}{2}Y^\top K_{T_{\mathrm{dr}}}^{-1}(Z, Z)Y - \frac{1}{2}\log|K_{T_{\mathrm{dr}}}(Z, Z)| - \frac{N_D}{2}\log 2\pi, \tag{19}$$

and we maximize this function with respect to θ.

2.4. Gaussian mixture regression

Gaussian Mixture Regression (GMR) is a multivariate nonlinear-function-regression method. To simplify the notation, we again omit t_k in the following discussion. GMR assumes that the joint distribution $\Pr(z, T_{\mathrm{dr}})$ is represented as a GMM with N_G Gaussian functions:

$$\Pr(z, T_{\mathrm{dr}}) = \sum_{i=1}^{N_G} \pi_i \mathcal{N}(z, T_{\mathrm{dr}}; \mu_i, \Sigma_i), \tag{20}$$

where π_i is the initial probability for (z, T_{dr}) to lie in the ith Gaussian, and μ_i and Σ_i are the mean and the covariance matrix of the ith Gaussian, defined as

$$\mu_i = \begin{bmatrix} \mu_i^z \\ \mu_i^{T_{\mathrm{dr}}} \end{bmatrix}, \quad \Sigma_i = \begin{bmatrix} \Sigma_i^{zz} & \Sigma_i^{zT_{\mathrm{dr}}} \\ \Sigma_i^{T_{\mathrm{dr}}z} & \Sigma_i^{T_{\mathrm{dr}}T_{\mathrm{dr}}} \end{bmatrix}. \tag{21}$$

The number of Gaussians, N_G, is tuned based on the Akaike and Bayesian information criteria (AIC [47] and BIC [48], respectively). We then compute the conditional probability $\Pr(T_{\mathrm{dr}}|z, i)$ as

$$\Pr(T_{\mathrm{dr}}|z, i) = \mu_i^{T_{\mathrm{dr}}} + \Sigma_i^{T_{\mathrm{dr}}z}(\Sigma_i^{zz})^{-1}(z - \mu_i^{zz}). \tag{22}$$

Thus, a GMR estimates the driver's torque $T_{\mathrm{dr}} = \sum_{i=1}^{N_G} h_i(z) \Pr(T_{\mathrm{dr}}|z, i)$ from

$$T_{\mathrm{dr}}^{\mathrm{GMR}} = \sum_{i=1}^{N_G} h_i(z)\left[\mu_i^{T_{\mathrm{dr}}} + \Sigma_i^{T_{\mathrm{dr}}z}(\Sigma_i^{zz})^{-1}(z - \mu_i^{zz})\right], \tag{23}$$

where $h_i(z) \in [0, 1]$ is the probability of z to belong to the ith Gaussian $h_i(z) = \mathcal{N}(z; \mu_i^z, \Sigma_i^{zz})$ and normalized such that $\sum_{i=1}^{N_G} h_i(z) = 1$.

2.5. Hidden Markov model Gaussian mixture regression

The HMM-GMR [26] combines a GMR with a hidden Markov model (HMM) and considers both the spatial and sequential information of $z(t_k)$ by incorporating transitions between Gaussians from the recursive computation of the weights in Eq. (23) as

$$h_i(z(t_k)) = \frac{\left(\sum_{j=1}^{N_G} h_j(z(t_{k-1}))a_{ji}\right)\hat{h}_i(z(t_k))}{\sum_{l=1}^{N_G}\left[\left(\sum_{j=1}^{N_G} h_j(z(t_{k-1}))a_{jl}\right)\hat{h}_l(z(t_k))\right]}, \tag{24}$$

where $\hat{h}_i(z(t_k)) = \mathcal{N}(z(t_k); \mu_i^z, \Sigma_i^z)$, and a_{ji} is the transition probability from the jth Gaussian to the ith Gaussian. In order to train the transition probability a_{ji}, we use the Baum–Welch algorithm [49], an expectation maximization algorithm. The prediction of HMM-GMR is

$$T_{\mathrm{dr}}^{\text{HMM-GMR}}(t_k) = \sum_{i=1}^{N_G} h_i(z)\left[\mu_i^{T_{\mathrm{dr}}} + \Sigma_i^{T_{\mathrm{dr}}z}(\Sigma_i^{zz})^{-1}(z - \mu_i^{zz})\right], \tag{25}$$

where z, as before, denotes $z(t_k)$.

2.6. Artificial neural network

While currently many complicated and deep artificial neural networks (ANN) are being proposed such as [46], because of the size of our data set, and to prevent overfitting, this work employs a simple ANN that has one hidden layer with N_h nodes. At each hidden unit $i \in \{1, \dots, N_h\}$, the input vector $z(t_k)$ is multiplied by a weight vector $W_h^i \in \mathbb{R}^d$ such that $h^i = W_h^{i\top} z(t_k) + b_h^i$, where $b_h^i \in \mathbb{R}$ is a bias-term vector. Then, the signal h^i becomes an input of a nonlinear transfer function $\phi : \mathbb{R} \to [-1, 1]$. This work uses a hyperbolic tangent function: $\phi(h^i) = 2/(1+\exp(-2h^i)) - 1$. Then, the output of the hidden nodes $\boldsymbol{\phi}(h) = [\phi(h^1), \dots, \phi(h^{N_h})]^\top \in \mathbb{R}^{N_h}$, becomes the input of the last layer, the output of which is the predicted T_{dr} by this ANN

$$T_{\mathrm{dr}}^{\mathrm{ANN}}(t_k) = W_o^\top \boldsymbol{\phi}(h) + b_o, \tag{26}$$

where $W_o \in \mathbb{R}^{N_h}$ and $b_o \in \mathbb{R}$. The training of the ANN deals with finding the best parameters W_h, W_o, b_h, and b_o, such that the output in Eq. (26) best fits the training and validation data sets.

3. Steering torque input and parameter estimation

In Section 2, we briefly introduced the driver lateral control prediction methods to be experimentally compared in this work. However, in order to evaluate the performance of each method, we need to have the values of the steering torque, namely, $\{T_{\mathrm{dr}}(t_k)\}_{k=1}^{N_D}$. Often this is not directly measurable, unless the steering wheel is equipped with a torque sensor, which conventional steering wheels for driving simulators are not equipped with. To this end, this section formulates the unknown steering torque input estimation problem and introduces a method to estimate the steering-wheel system parameters and the unknown steering-torque input.

3.1. Steering torque input estimation

The steering column dynamics can be modeled as follows:

$$J_s \frac{\mathrm{d}^2}{\mathrm{d}t^2}\delta_s(t) + b_s \frac{\mathrm{d}}{\mathrm{d}t}\delta_s(t) = T_{\mathrm{dr}}(t) + T_{\mathrm{aln}}(t), \tag{27}$$

where J_s is the steering column moment of inertia, b_s is the friction coefficient of the steering column, δ_s is the steering wheel angle, T_{dr} is the *unknown* steering torque from a human driver, and T_{aln} is the alignment torque from the front tires. Note that in a simulation environment, T_{aln} is available. Therefore, our problem setup assumes that the value of $T_{\mathrm{aln}}(t)$ is known. Consequently, we estimate the system state $\boldsymbol{x}(t)$, the system parameters J_s and b_s, and the unknown input $T_{\mathrm{dr}}(t)$.

We rewrite the steering column system (27) as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -b_s/J_s \end{bmatrix}\boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1/J_s \end{bmatrix} T_{\mathrm{aln}}(t) + \begin{bmatrix} 0 \\ 1/J_s \end{bmatrix} T_{\mathrm{dr}}(t), \tag{28}$$

$$y(t) = \begin{bmatrix} 180/\pi & 0 \end{bmatrix}\boldsymbol{x}(t), \tag{29}$$

where $\boldsymbol{x}(t) = [\delta_s(t), \omega_s(t)]^\top$, with ω_s being the angular velocity of the steering wheel.

Our approach consists of the following two steps. The first step employs the force-feedback functionality of the steering wheel to collect data to estimate the system parameters. Note that, at this step, we steer the steering wheel only by the force-feedback functionality and do not employ any human torque. Thus, we do not have any unknown inputs in this data set, i.e., $T_{\mathrm{dr}} = 0$. In the second step, the estimated J_s and b_s are employed to estimate the unknown input T_{dr}. To this end, we employ a PI observer.

Our approach is based on the full-order observer design [50]. Suppose we have a linear time-invariant (LTI) system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = A\boldsymbol{x}(t) + B_u\boldsymbol{u}(t) + B_v\boldsymbol{v}(t), \tag{30a}$$

$$\boldsymbol{y}(t) = C\boldsymbol{x}(t), \tag{30b}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the system state, $\boldsymbol{y} \in \mathbb{R}^{n_y}$ is the output $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the *known* input, and $\boldsymbol{v} \in \mathbb{R}^{n_v}$ is the *unknown* input. Here, based on the discussion in [51], we assume that

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v}(t) = 0, \tag{31}$$

and rewrite the system as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}(t)\\ \boldsymbol{v}(t)\end{bmatrix} = \begin{bmatrix} A & B_v \\ 0 & 0 \end{bmatrix}\begin{bmatrix}\boldsymbol{x}(t)\\ \boldsymbol{v}(t)\end{bmatrix} + \begin{bmatrix} B_u \\ 0 \end{bmatrix}\boldsymbol{u}(t). \tag{32}$$

Then, we design a PI observer as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}(t) = F\boldsymbol{z}(t) + (L_1 + L_2)\boldsymbol{y}(t) + J\boldsymbol{u}(t) + T_1 B_v \hat{\boldsymbol{v}}(t), \tag{33a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{v}}(t) = L_3(\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)), \tag{33b}$$

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{z}(t) + T_2\boldsymbol{y}(t), \tag{33c}$$

$$\hat{\boldsymbol{y}}(t) = C\hat{\boldsymbol{x}}(t), \tag{33d}$$

where $\boldsymbol{z} \in \mathbb{R}^{n_x}$, $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x}$ is the estimated state, and $\hat{\boldsymbol{v}} \in \mathbb{R}^{n_v}$ is the estimated unknown input. Our objective here is to design the matrices $F, L_1, L_2, L_3, J, T_1, T_2$ such that both $\boldsymbol{e}(t) \triangleq \boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)$ and $\boldsymbol{e}_v(t) \triangleq \boldsymbol{v}(t) - \hat{\boldsymbol{v}}(t)$ are exponentially stable at the origin. The following theorem summarizes our approach.

Theorem 1. *Given an LTI system (30) and the assumption that the pair*

$$\left(\begin{bmatrix} A & B_v \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} C & 0 \end{bmatrix}\right) \tag{34}$$

is detectable, one can design a PI observer that estimates the unknown input and achieves the exponential stability of $\boldsymbol{e}(t)$ and $\boldsymbol{e}_v(t)$ at the origin using the following process:

1. *Design T_1 and T_2 such that*

$$\begin{bmatrix} T_1 & T_2 \end{bmatrix} = \begin{bmatrix} I_{n_x} \\ C \end{bmatrix}^+, \tag{35}$$

where, since $\operatorname{rank}[I_{n_x}\ C^\top]^\top = n_x$,

$$\begin{bmatrix} T_1 & T_2 \end{bmatrix}\begin{bmatrix} I_{n_x} \\ C \end{bmatrix} = I_{n_x}. \tag{36}$$

2. *Determine F, L_1, and J from*

$$F = T_1 A - L_2 C, \tag{37a}$$

$$L_1 = FT_2, \tag{37b}$$

$$J = T_1 B_u. \tag{37c}$$

3. *Design matrices $V_1 \succeq 0$ and $V_2 \succ 0$. Then L_2 and L_3 are determined from*

$$\begin{bmatrix} L_2 \\ L_3 \end{bmatrix} = Q\tilde{C}^\top V_2^{-1}, \tag{38}$$

where Q is the solution of the following observer Riccati equation:

$$0 = \tilde{A}Q + Q\tilde{A}^\top + V_1 - Q\tilde{C}^\top V_2^{-1}\tilde{C}Q, \tag{39}$$

where

$$\tilde{A} = \begin{bmatrix} T_1 A & T_1 B_v \\ 0 & 0 \end{bmatrix}, \quad \tilde{C} = \begin{bmatrix} C & 0 \end{bmatrix} \tag{40}$$

Proof. It follows from (36) that

$$\boldsymbol{e}(t) = T_1\boldsymbol{x}(t) - \boldsymbol{z}(t). \tag{41}$$

Accordingly, it follows from (33) that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{e}(t)\\ \boldsymbol{e}_v(t)\end{bmatrix} &= \begin{bmatrix}F & T_1B_v\\ -L_3C & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t)\\ \boldsymbol{e}_v(t)\end{bmatrix}\\ &\quad+\begin{bmatrix}T_1A - FT_1 - (L_1+L_2)C\\ 0\end{bmatrix}\boldsymbol{x}(t)\\ &\quad+\begin{bmatrix}T_1B_u - J\\ 0\end{bmatrix}\boldsymbol{u}(t).\end{aligned} \tag{42}$$

Thus, using (37)

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{e}(t)\\ \boldsymbol{e}_v(t)\end{bmatrix} = \begin{bmatrix}T_1A - L_2C & T_1B_v\\ -L_3C & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t)\\ \boldsymbol{e}_v(t)\end{bmatrix}. \tag{43}$$

Thus, if the matrix

$$A_{\mathrm{obs}} \triangleq \begin{bmatrix}T_1A - L_2C & T_1B_v\\ -L_3C & 0\end{bmatrix} \tag{44}$$

is Hurwitz, then $\boldsymbol{e}(t)$ and $\boldsymbol{e}_v(t)$ are exponentially stable at the origin. Since

$$A_{\mathrm{obs}} = \tilde{A} - \begin{bmatrix}L_2\\ L_3\end{bmatrix}\tilde{C}, \tag{45}$$

the L_2 and L_3 determined from (38) makes A_{obs} Hurwitz. Note that (39) has a solution because the pair $(\tilde{A}, \tilde{C})$ is detectable from the following PBH test. Since the pair in (34) is detectable, it follows that

$$\mathrm{rank}\left(\begin{bmatrix}sI_{n_x} - A & -B_v\\ 0 & sI_{n_v}\\ C & 0\end{bmatrix}\right) = n_x + n_v \qquad \forall s \in \mathbb{C}^+. \tag{46}$$

We check the detectability condition of the pair $(\tilde{A}, \tilde{C})$ as follows

$$\mathrm{rank}\left(\begin{bmatrix}sI_{n_x+n_v} - \tilde{A}\\ \tilde{C}\end{bmatrix}\right) = \mathrm{rank}\left(\begin{bmatrix}sI_{n_x} - T_1A & -T_1B_v\\ 0 & sI_{n_v}\\ C & 0\end{bmatrix}\right) \tag{47}$$

$$= \mathrm{rank}\left(\begin{bmatrix}s(T_1+T_2C) - T_1A & -T_1B_v\\ 0 & sI_{n_v}\\ C & 0\end{bmatrix}\right) \quad (\text{Eq.}(36)) \tag{48}$$

$$= \mathrm{rank}\left(\begin{bmatrix}T_1 & 0 & sT_2\\ 0 & I_{n_v} & 0\\ 0 & 0 & I_{n_y}\end{bmatrix}\begin{bmatrix}sI_{n_x} - A & -B_v\\ 0 & sI_{n_v}\\ C & 0\end{bmatrix}\right). \tag{49}$$

The first matrix of the right hand side of (49) is invertible. Thus,

$$\begin{aligned}\mathrm{rank}\left(\begin{bmatrix}sI_{n_x+n_v} - \tilde{A}\\ \tilde{C}\end{bmatrix}\right) &= \mathrm{rank}\left(\begin{bmatrix}sI_{n_x} - A & -B_v\\ 0 & sI_{n_v}\\ C & 0\end{bmatrix}\right)\\ &= n_x + n_v \qquad \forall s \in \mathbb{C}^+.\end{aligned} \tag{50}$$

Thus, the pair $(\tilde{A}, \tilde{C})$ is also detectable. ■

Remark 1. In [50], the authors employ a pole placement method assuming the pair $(\tilde{A}, \tilde{C})$ is observable, which is more strict than our detectability assumption.

Remark 2. Note that, in our case with (28) and (29), the pair in (34) is detectable. Thus, it follows from Theorem 1 that one can estimate the unknown input $\boldsymbol{v} = T_{\mathrm{dr}}$.

Remark 3. Besides the PI observer [50,52,53], which we employ in this paper, there are two other main methods to estimate unknown inputs. One method is the unknown input observer (UIO) [54,55]. UIO assumes the UI decoupled condition

$$\mathrm{rank}(CB_v) = \mathrm{rank}(B_v), \tag{51}$$

where C is the observation matrix, and B_v is the input matrix corresponding to the unknown input. Note that this UI decoupled condition does not hold for our setting because, in our case, $C = \begin{bmatrix}180/\pi & 0\end{bmatrix}$ and $B_v = \begin{bmatrix}0 & 1/J_s\end{bmatrix}^\top$. The second approach for unknown input estimation is to employ EKF or UKF [56,57]. However, as discussed in [56], to estimate unknown inputs, the number of measurements needs to be larger than the total number of unknown inputs. Since our problem setup assumes single output (δ_s) and single unknown input (T_{dr}), we cannot use this approach, either.

3.2. Steering wheel parameter estimation

In order to estimate the state and the unknown parameters of the system we use a joint EKF estimator. We assume that we have a nonlinear discrete system with unknown parameters:

$$x_s(k+1) = \bar{f}(x_s(k), u(k), x_p(k)) + w_s(k), \tag{52}$$

$$x_p(k+1) = x_p(k) + w_p(k), \tag{53}$$

$$y(k) = \bar{h}(x_s(k), x_p(k)) + v(k), \tag{54}$$

where x_s, x_p, y are the system state, system parameter, and observation, respectively. Furthermore, w_s, w_p, and v are state process, parameter process, and observation noise, respectively, which we assume to be zero-mean Gaussian. The joint EKF concatenates the system state and system parameters into a single, joint state vector: $z(k) = [x_s(k)^\top, x_p(k)^\top]^\top$. Thus, by estimating the value of $z(k)$, we simultaneously estimate the system state and parameters. The system can be written as

$$z(k+1) = f(z(k), u(k)) + w(k), \tag{55}$$

$$y(k) = h(z(k)) + v(k), \tag{56}$$

where $w \sim \mathcal{N}(0, Q_z)$ and $v \sim \mathcal{N}(0, R)$. The first step is the prediction step. With $\hat{z}$ and Φ being the estimations of z and the joint state covariance matrix, respectively, the joint state and covariance matrix are predicted as

$$\hat{z}^-(k) = f(\hat{z}(k-1), u(k-1)), \tag{57}$$

$$\Phi^-(k) = J_z(k)\Phi(k-1)J_z^\top(k) + Q_z, \tag{58}$$

where $\hat{z}^-$ and Φ^- are the predictions of z and Φ, respectively. The measurement update is conducted as

$$K(k) = \Phi^-(k)H^\top(k)\left[R + H(k)\Phi^-(k)H^\top(k)\right]^{-1}, \tag{59}$$

$$\hat{z}(k) = \hat{z}^-(k) + K(k)\left[y(k) - h(\hat{z}^-(k))\right], \tag{60}$$

$$\Phi(k) = [I - K(k)H]\,\hat{\Phi}^-(k), \tag{61}$$

where

$$J_z(k) = \frac{\partial f}{\partial z}|_{\hat{z}(k-1)}, \quad H(k) = \frac{\partial h}{\partial z}|_{\hat{z}^-(k-1)}. \tag{62}$$

In addition, and similarly to the joint EKF, a joint UKF can also be used to estimate the unknown system parameters by concatenating the system state and parameters into a single, joint state vector $z(k) = [x_s(k)^\top, x_p(k)^\top]^\top$. The first step of the joint UKF is to create a matrix that consists of sigma points $\mathcal{X} \in \mathbb{R}^{n\times(2n+1)}$ and weight vectors $\mathcal{W}_c^p$, $\mathcal{W}_m^p \in \mathbb{R}^{1\times(2n+1)}$, where n is the dimensionality of the concatenated state z.

$$\begin{aligned}\mathcal{X}(k-1) = \Big[&\hat{z}(k-1), \hat{z}(k-1) + \sqrt{\lambda+n}\sqrt{\Phi(k-1)}, \hat{z}(k-1)\\ &-\sqrt{\lambda+n}\sqrt{\Phi(k-1)}\Big],\end{aligned} \tag{63}$$

$$J_z(k) = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 - b_s^-(k)\Delta t/J_s^-(k) & \left(b_s^-(k)\omega_s^-(k) - u(k)\right)\Delta t/J_s^-(k)^2 & -\omega_s^-(k)\Delta t/J_s^-(k) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{79}$$

$$H(k) = [180/\pi \quad 0 \quad 0 \quad 0], \tag{80}$$

Box I.

$$\mathcal{W}_c = \left[\frac{\lambda}{\lambda+n}, \frac{1}{2(\lambda+n)}, \ldots, \frac{1}{2(\lambda+n)}\right], \tag{64}$$

$$\mathcal{W}_m = \left[\frac{\lambda}{\lambda+n} + 1 - \alpha^2 + \beta, \frac{1}{2(\lambda+n)}, \ldots, \frac{1}{2(\lambda+n)}\right]. \tag{65}$$

Also, $\lambda = \alpha^2(n_x+\kappa) - n_x$ is a scaling parameter. The parameters α, β, and κ are the tuning parameters for UKF. Then, we perform the noise-free state propagation of the sigma points

$$\mathcal{X}^*(k)[i] = f(\mathcal{X}(k-1)[i], u(k-1)) \qquad i = 1, \ldots, 2n+1, \tag{66}$$

where $\mathcal{X}(k-1)[i]$ represents the ith column of $\mathcal{X}(k-1)$. Using the updated sigma points and the corresponding weights, we predict the joint state and covariance matrix as follows:

$$\hat{z}^-(k) = \sum_{i=1}^{2n+1} \mathcal{W}_m[i]\mathcal{X}^*(k)[i], \tag{67}$$

$$\Phi^-(k) = Q + \sum_{i=1}^{2n+1} \mathcal{W}_c[i]\left(\mathcal{X}^*(k)[i] - \hat{z}^-(k)\right)\left(\mathcal{X}^*(k)[i] - \hat{z}^-(k)\right)^\top, \tag{68}$$

where $\mathcal{W}_m[i]$ and $\mathcal{W}_c[i]$ represent the ith component of $\mathcal{W}_m$ and $\mathcal{W}_c$. We perform the measurement update first by updating the sigma points around $\hat{z}^-(k)$

$$\bar{\mathcal{X}}(k) = \left[\hat{z}^-(k), \hat{z}^-(k) + \sqrt{\lambda+n}\sqrt{\Phi^-(k)}, \hat{z}^-(k) - \sqrt{\lambda+n}\sqrt{\Phi^-(k)}\right], \tag{69}$$

and second by obtaining the predicted observation of sigma points, $\mathcal{Y} \in \mathbb{R}^{n_y \times (2n+1)}$

$$\mathcal{Y}(k)[i] = h(\bar{\mathcal{X}}(k)[i]) \qquad i = 1, \ldots, 2n+1, \tag{70}$$

where n_y is the dimensionality of the observation vector y in Eq. (56). The measurement update is given as follows:

$$\hat{y}(k) = \sum_{i=1}^{2n+1} \mathcal{W}_m[i]\mathcal{Y}(k)[i], \tag{71}$$

$$S(k) = R + \sum_{i=1}^{2n+1} \mathcal{W}_c[i]\left(\mathcal{Y}(k)[i] - \hat{y}(k)\right)\left(\mathcal{Y}(k)[i] - \hat{y}(k)\right)^\top, \tag{72}$$

$$\Phi^{zy}(k) = \sum_{i=1}^{2n+1} \mathcal{W}_c[i]\left(\bar{\mathcal{X}}(k)[i] - \hat{z}^-(k)\right)\left(\mathcal{Y}(k)[i] - \hat{y}(k)\right)^\top, \tag{73}$$

$$K(k) = \Phi^{zy}(k)S(k)^{-1}, \tag{74}$$

$$\hat{z}(k) = \hat{z}^-(k) + K(k)(y(k) - \hat{y}(k)), \tag{75}$$

$$\Phi(k) = \Phi^-(k) - K(k)S(k)K(k)^\top. \tag{76}$$

3.3. Details of overall proposed approach

So far, we have introduced the methods we utilize in the proposed approach. Next, we explain how we used these methods for estimating the steering input torque. We first conducted a system identification experiment, which is summarized in Algorithm 1. After the system parameters were estimated, we conducted the unknown input estimation using a PI observer as described in Algorithm 2.

First, we identified the system parameters J_s and b_s using a joint E/UKF. In our joint E/UKF setting, we define the joint state vector $z(k) = [\delta_s(k), \omega_s(k), J_s(k), b_s(k)]^\top$. The system dynamics of $z(k)$ is

$$z(k+1) = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 - b_s\Delta t/J_s & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ \Delta t/J_s \\ 0 \\ 0 \end{bmatrix} u(k) + w(k), \tag{77}$$

$$y(k) = \begin{bmatrix} 180/\pi & 0 & 0 & 0 \end{bmatrix} z(k) + v(k), \tag{78}$$

where Δt is the sampling time interval. Note that $u(k) = T_{\text{aln}}(k)$ for Algorithm 1 and $u(k) = T_{\text{aln}}(k) + T_{\text{dr}}(k)$ for Algorithm 2. The Jacobians needed for the joint EKF are given in Box I where $\hat{z}^-(k) = [\delta_s^-(k), \omega_s^-(k), J_s^-(k), b_s^-(k)]^\top$.

Algorithm 1 Steering Wheel System Identification.

Input: $\mathcal{D}_{\text{Synthetic}} = \{\delta_s(1), \ldots, \delta_s(T), T_{\text{aln}}(1), \ldots, T_{\text{aln}}(T)\}$
Output: J_s, b_s
1: Specify initial guess: x_0, Φ_0, J_{s0}, b_{s0}
2: Specify process and observation covariance matrices: Q, R
3: Make joint state $z_0 = [x_0, J_{s0}, b_{s0}]^\top$.
4: **for** k=1:T **do**
5: Joint state prediction via Eqs. (57)–(58) or Eqs. (67)–(68).
6: Joint state measurement update via Eqs. (60)–(61) or Eqs. (75)–(76).
7: **end for**
8: **if** Covnergence **then**
9: Return $J_s(T)$, $b_s(T)$
10: **else**
11: Change initial guess or covariance matrices and restart from line 3.
12: **end if**

Algorithm 2 Unknown Steering Torque T_{dr} Estimation.

Input: $\mathcal{D} = \{\delta_s(1), \ldots, \delta_s(T), T_{\text{aln}}(1), \ldots, T_{\text{aln}}(T), J_s, b_s\}$
Output: $T_{\text{dr}}(1), \ldots, T_{\text{dr}}(T)$
1: Specify V_1 and V_2 and compute the design matrices using Theorem 1
2: **for** k=1:T **do**
3: Run the PI observer (33).
4: **end for**
5: $T_{\text{dr}} \leftarrow \hat{\boldsymbol{v}}$.
6: Return T_{dr}.

Remark 4. While a dual EKF can be applied to estimate unknown parameters [58,59], the dual EKF cannot estimate the parameters of the steering wheel because, in the case of the dual EKF, the Jacobian of h with respect to the parameters for our case is

$$H_p = C\frac{\partial}{\partial x_p} f(x_s, x_p, u) = \begin{bmatrix} 180/\pi & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 \\ b_s\omega_s/J_s - u/J_s^2 & -\omega_s/J_s \end{bmatrix} = 0. \tag{81}$$

Thus, the measurement of the output cannot update the estimation of the system parameters. Similarly, the dual UKF cannot update the system parameters using the measurements. On the contrary, as we have shown, the joint E/UKF can update the parameter estimate.

4. Validation of torque input estimation algorithm

In this section we test the proposed algorithms using numerical simulations followed by experiments.

4.1. Numerical simulation

First, we verify the algorithm using synthetic data. We use a steering wheel model and simulated unknown input. As the steering wheel parameters in Eqs. (28)–(29), we use

$$J_s = 0.0672 \text{ kg m}^2, \qquad b_s = 0.3562 \text{ N m/rad/s}. \tag{82}$$

The known input signal is sinusoidal:

$$u(t) = 2\ \sin(4t) \text{ N m}. \tag{83}$$

The initial condition is chosen as $[\delta_s(0), \omega_s(0)]^\top = [1, 0]^\top$. The length of the data is 15 s with sampling time interval $\Delta t = 0.01$ s. As the process and measurement noise, we use

$$w_s \sim \mathcal{N}\left(0, \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}\right), \qquad v \sim \mathcal{N}(0, 0.01). \tag{84}$$

System parameter identification. In order to estimate the values of J_s and b_s we employ and compare the joint E/UKF having the following initial state, process noise, and observation noise covariance matrices

$$\begin{aligned} &\Phi(0) = \text{diag}(1,\ 1,\ 8\times10^{-3},\ 0.1), \\ &Q_z = \text{diag}(0.01,\ 0.01,\ 5\times10^{-8},\ 5\times10^{-8}), \qquad R = 0.01. \end{aligned} \tag{85}$$

As the tuning parameters for UKF, we employ the most general values, i.e.,

$$\alpha = 1\times10^{-3}, \qquad \beta = 2, \qquad \kappa = 0. \tag{86}$$

Figs. 2 and 3 depict the system parameter identification results using the joint EKF and joint UKF, respectively. As an initial guess, we used $[\hat\delta_s(0), \hat\omega_s(0)]^\top = [0, 0]^\top$, $\hat J_s(0) = 0.5J_s$, and $\hat b_s(0) = 0.5b_s$. Subfigures (a) and (b) in Figs. 2 and 3 depict each state. Subfigures (c) and (d) in Figs. 2 and 3 depict the estimation history of J_s, and subfigures (e) and (f) in Figs. 2 and 3 depict the estimation history of b_s. While subfigures (c) and (e) in Figs. 2 and 3 depict the 1σ error bar to better see the convergence of the covariance matrix, subfigures (d) and (f) in Figs. 2 and 3 illustrate only the estimated and true values of J_s and b_s to better see that the estimation converges to its true value. The final estimated values of J_s and b_s for each method are listed in Table 1.

Thanks to its second order accuracy, the joint UKF exhibited faster convergence than the joint EKF, which has first order accuracy. However, we observed that the joint UKF was more sensitive to the parameter settings. We conjecture that this is due to the nonlinearity caused by the J_s in the denominator of the Eqs. (28)–(29). In contrast, the joint EKF was more robust to the parameter settings.

Table 1
Estimated values and error from the true values of J_s and b_s.

	True value	Joint EKF	Joint UKF
J_s (kg m^2)	0.0672	0.0654 (-1.8×10^{-3})	0.0681 ($+0.9\times10^{-3}$)
b_s (N m/rad/s)	0.3562	0.3498 (-6.4×10^{-3})	0.3572 ($+1.0\times10^{-3}$)

Steering torque estimation. After the system parameters are identified, we proceed to the second step, namely to estimate the unknown steering torque input. As synthetic inputs to the system, we employ

$$T_{\text{dr}}(t) = 3\sin(3t) \text{ N m}, \tag{87}$$

and

$$T_{\text{aln}}(t) = \sin(4t - \frac{\pi}{5}) \text{ N m}. \tag{88}$$

The task here is to estimate T_{dr} at each time step given the values of J_s and b_s and given the observations of the steering angle and T_{aln}. The process and observation noise are the same as (84).

Fig. 4 depicts the estimation results with $V_1 = \texttt{diag}(1,\ 1,\ 1.0\times10^{11})$ and $V_2 = 1$. Note that we use the true values of J_s and b_s in order to isolate the unknown-input estimation error from the joint E/UKF error. In order to compute the integral in the computation of the PI observer, the fourth-order Runge–Kutta method was employed. The root mean squared error, defined as $\text{RMSE}(\hat T_{dr}, T_{\text{dr}}) = \sqrt{1/N\sum_{k=1}^{N}(\hat T_{\text{dr}}(k) - T_{\text{dr}}(k))^2}$, where N is the number of observations, of this estimation was 0.39 N m.

4.2. Experimental results

Section 4.1 employed synthetic data to verify the proposed algorithm. Here, we use real data collected using a driving simulator we built in house.

First, we preset the experimental results from the steering-wheel force-feedback unit identification. Since the unit of the force-feedback torque command of our steering wheel is unknown, before conducting the experiments, we start with identifying the relationship between the input signal values in the system and the torque values in the real world. To this end, we hung weights on the steering wheel, and computed the torque to make the steering wheel angle zero by searching over the steering command torque T_{com} such that $T_{\text{com}} = mgr$, where m is the weight, $g = 9.81$ m/s^2 is the gravitational acceleration, and $r = 0.14$ m is the outer radius of the steering wheel. We then designed a PI controller and measured the steering torque command T_{com}. We performed the experiment multiple times with different values of m. From this experiment, we found that the conversion from T_{com} to T_{aln} is

$$T_{\text{aln}}(t) = 1.8615\ T_{\text{com}}(t) + 0.2197 \text{ N m}. \tag{89}$$

In addition to this unit identification, we resampled the data so that the sampling interval became 0.02 s using nearest neighbor interpolation before applying the methods in Sections 2 and 3.

Similarly to the case with the synthetic data in Section 4.1, the first step is to identify the steering wheel parameters J_s and b_s using joint E/UKF. First, we generated a chirp signal and activated the force-feedback functionality of our steering wheel. Note that, in this scenario, $T_{\text{dr}} = 0$. As the initial guess we set $[\hat\delta_s(0), \hat\omega_s(0)]^\top = [0, 0]^\top$, $\hat J_s(0) = 0.0672$ kg m^2 and $\hat b_s(0) = 0.3562$ N m/rad/s. Figs. 5 and 6 illustrate the results from the joint E/UKF estimations, respectively. The final estimated values of J_s and b_s using joint EKF are

$$J_s^{\text{EKF}} = 0.0298 \text{ kg m}^2, \qquad b_s^{\text{EKF}} = 0.3515 \text{ N m/rad/s}, \tag{90}$$

and the values by joint UKF are

$$J_s^{\text{UKF}} = 0.0296 \text{ kg m}^2, \qquad b_s^{\text{UKF}} = 0.3514 \text{ N m/rad/s}. \tag{91}$$

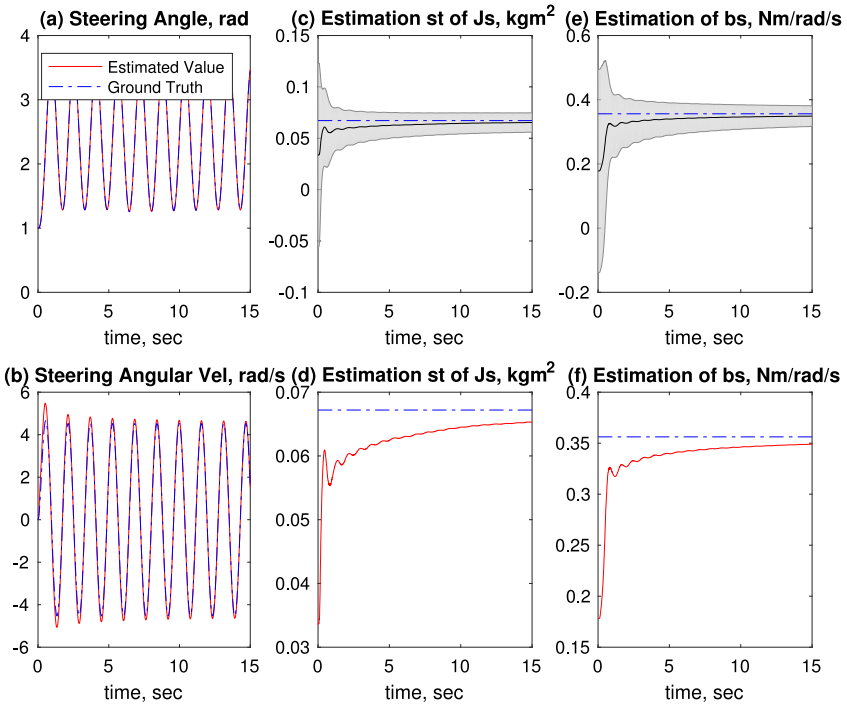


Fig. 2. System identification using joint EKF.

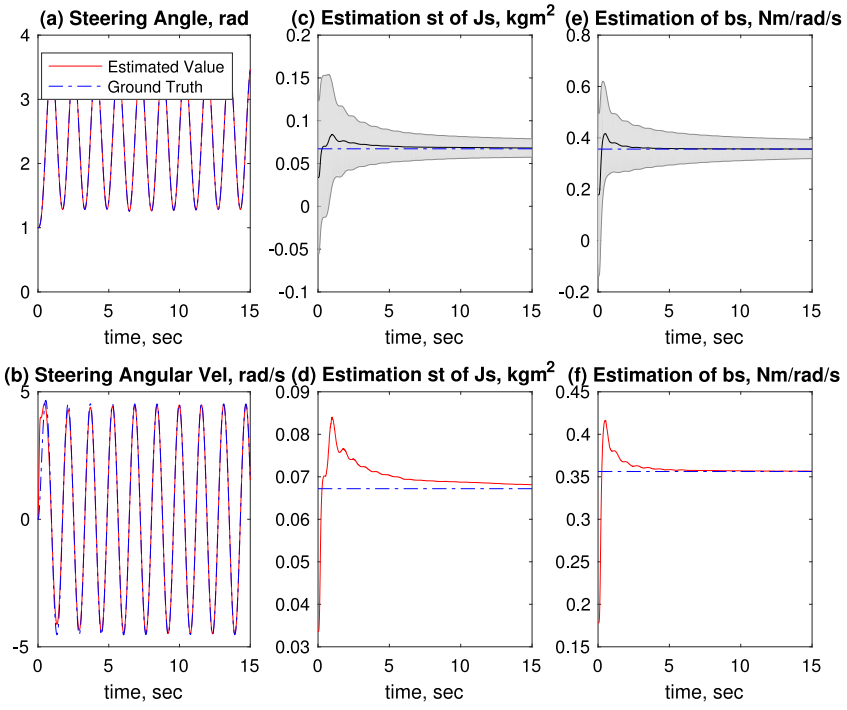


Fig. 3. System identification using joint UKF.

We employ these values to estimate the steering torque input from human drivers. In order to collect data, we used the Georgia Tech Driving Simulator (GTDS) shown in Fig. 7a. The human driver subject has eight-year driving experience, and the geometry of the road is depicted in Fig. 7b.

Fig. 7c illustrates the interaction between a human driver and the components of the simulator. Each component is connected via the Robot Operating System (ROS) [60], while CarSim® [61] computes the vehicle dynamics. The high-fidelity vehicle model of CarSim® reproduces realistic vehicle behaviors in a simulation environment. In this experiment, the vehicle speed is fixed, and the control output from the human driver is only the steering torque T_{dr}, which then becomes the input to the steering wheel. The output of the steering wheel is the steering wheel angle δ_s. The simulator uses a high-end gaming steering wheel with force-feedback functionality, and the driver can feel the alignment torque from the steering wheel, calculated as

$$T_{\mathrm{aln}}(t) = \frac{K_{\mathrm{aln}}}{g_s}\left(\beta(t) + \frac{\ell_F r(t)}{V_x(t)} - \delta(t)\right), \tag{92}$$

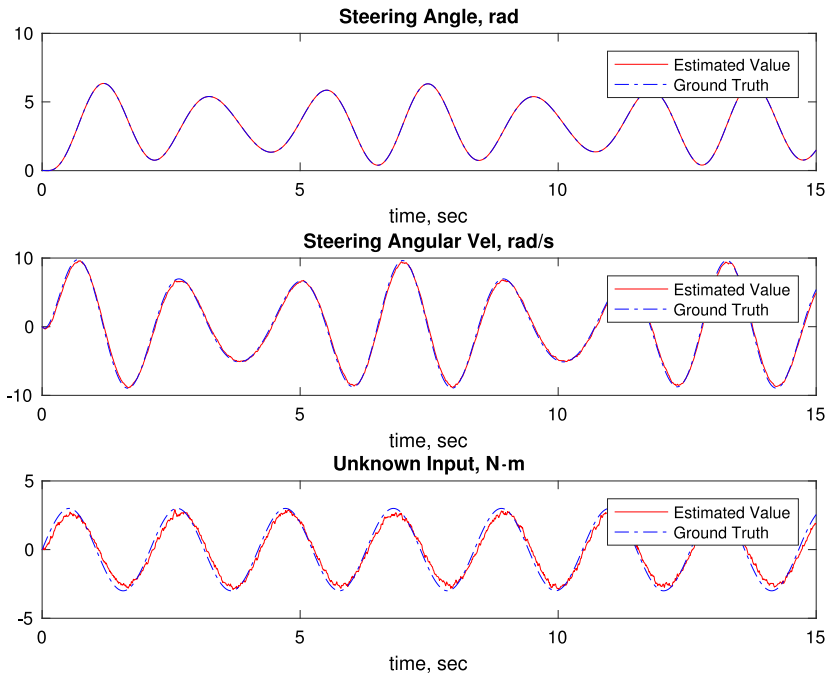


Fig. 4. Results of unknown input estimation.

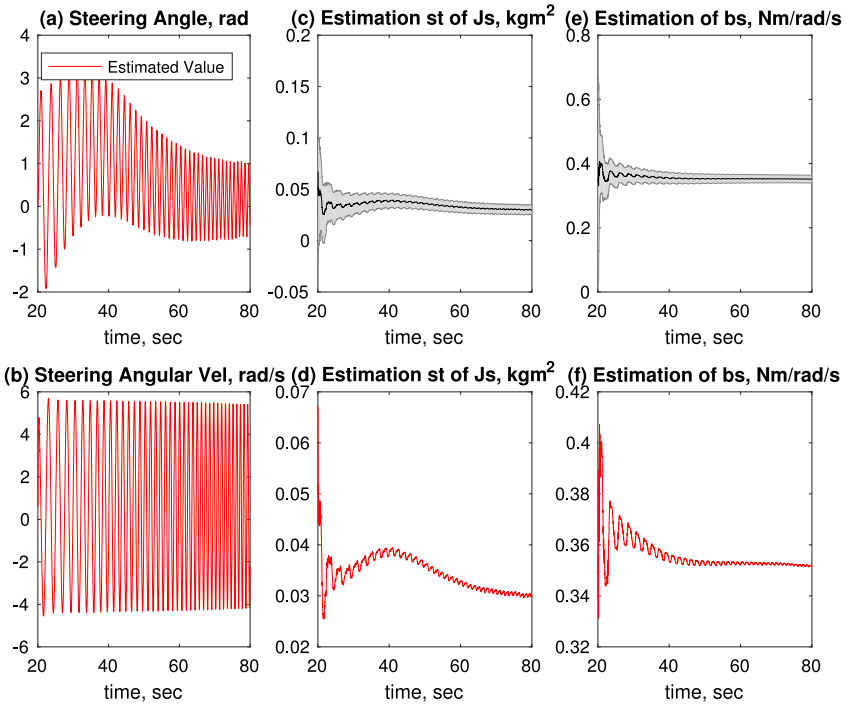


Fig. 5. System identification of the GTDS steering wheel using joint EKF.

where $K_{\text{aln}} = -K_p C_f n_t$, and where $K_p > 0$ is the manual steering column coefficient, $C_f > 0$ is the front tire cornering stiffness, which we assume to be constant, n_t is the tire length contact, g_s is the gear ratio, ℓ_F is the distance of the vehicle mass center from the front axle, V_x is longitudinal speed of the vehicle, and δ is the front-wheel-steering angle. This force-feedback makes the simulation environment more realistic. Note that all the variables to compute T_{aln} are available in our simulation environment. In order to visualize the information computed by CarSim®, we employed Unity3D [62], and projected the simulated driver view on a 8×6 ft^2 screen. Based on the view of the screen, the human driver steers the vehicle (see Fig. 7a).

Using the GTDS driving simulator, we collected data from a human driver. The course employed was one of the pre-installed CarSim® road circuits. The vehicle speed was fixed at 50 km/h, which was high for the radii of the corners of the given road circuit, so that we could investigate both the comfort and non-comfort zones of the driver. Fig. 8 depicts the collected data. We set the matrices for the PI observer as $V_1 = \texttt{diag}(1.0, 1.0, 3.1 \times 10^9)$ and $V_2 = 1.0$.

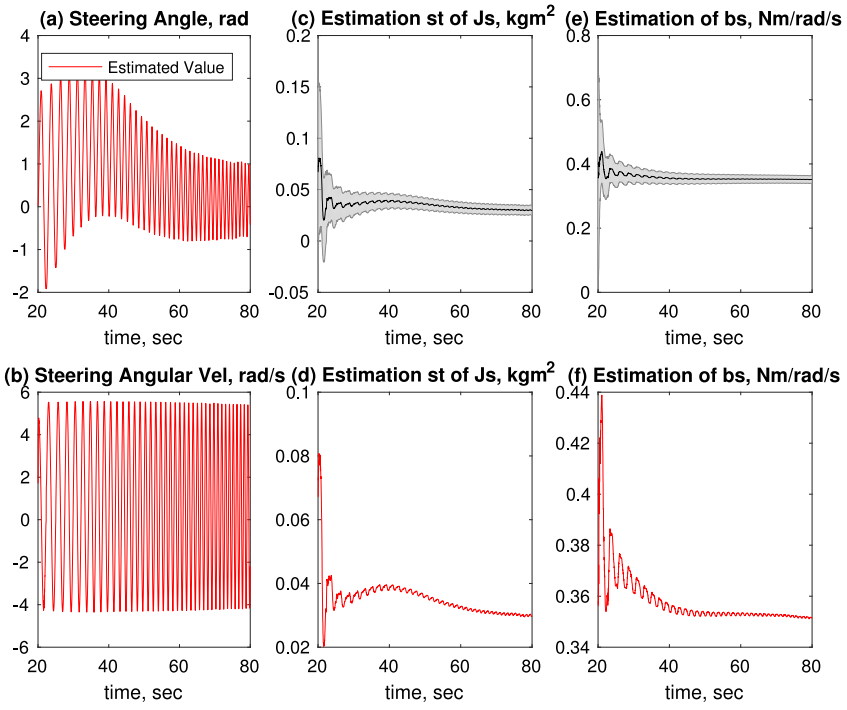
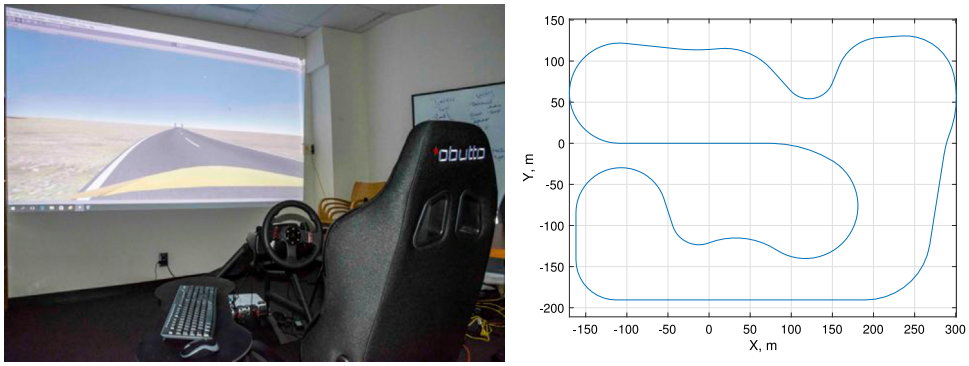
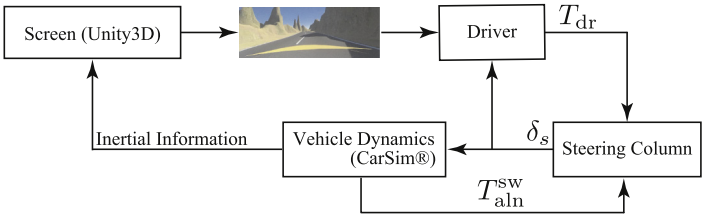


Fig. 6. System identification of the GTDS steering wheel using joint UKF.



(a) GTDS screen, steering wheel, and driver's seat.

(b) The road geometry.



(c) Structure of the GTDS simulator.

Fig. 7. Georgia Tech Driving Simulator (GTDS).

Figs. 9a and 9b show the results when we employed the estimation results by joint EKF (90) and joint UKF (91), respectively. In order to evaluate the estimated T_{dr}, we compared the measured steering wheel angle and the output of the identified system ((28) and (29) with (90)) given the estimated T_{dr}. The results are illustrated in Figs. 10a and 10b for joint EKF and joint UKF, respectively. The performance difference between the parameters estimated using the joint EKF and joint UKF is difficult to discern from Figs. 10a and 10b, but the root mean squared error of the steering wheel angle of each method is $\mathrm{RMSE}(\hat{\delta}_s, \delta_s | J_s^{\mathrm{EKF}}, b_s^{\mathrm{EKF}}) = 24.93$ deg and $\mathrm{RMSE}(\hat{\delta}_s, \delta_s | J_s^{\mathrm{UKF}}, b_s^{\mathrm{UKF}}) = 24.90$ deg, respectively. Thus, we observed a slightly better performance if we employ the joint UKF. However, the estimation from the joint UKF was more sensitive to the parameter settings than the joint EKF. Thus, considering minor performance difference, the joint EKF seems to be a better solution in practice.

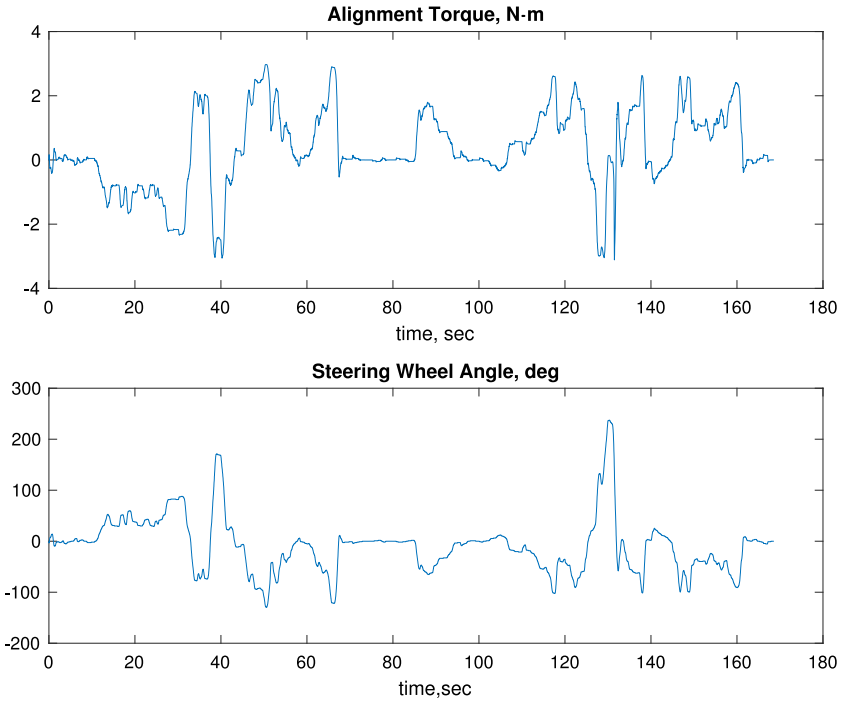


Fig. 8. Data collected using the Georgia Tech Driving Simulator (GTDS).

5. Comparative study of data-driven driver lateral control models

In Section 2 we introduced the data-driven human driver control models we wish to compare in this work. In order to evaluate the performance of these algorithms, we need the measurement of the steering torque, which is not measurable for a conventional steering wheel of a driving simulator. Thus, in Section 3, we developed an algorithm to estimate the steering input torque using joint E/UKF and a PI observer, so that we can evaluate the prediction performance of the data-driven driver lateral control models. We conducted two experiments to apply the proposed approach. First, we used a driver control model to generate a synthetic data set, applied the algorithms in Section 2, and evaluated their performance. The purpose of the first experiment was to evaluate the performance without the effect of inconsistency inherent in human driving skills, which always exist with real data, especially from non-professional human drivers. By employing a mathematical human driver control model, we can eliminate this inconsistency of driving behaviors and isolate and evaluate the regression performance of each algorithm. In the second experiment, we employed a real human-driving data set that is collected using the GTDS driving simulator in Fig. 7a. From this experiment, we compared and evaluated the robustness to driving-behavior variations. The results of the experiments are discussed in Section 6.

5.1. Performance measure

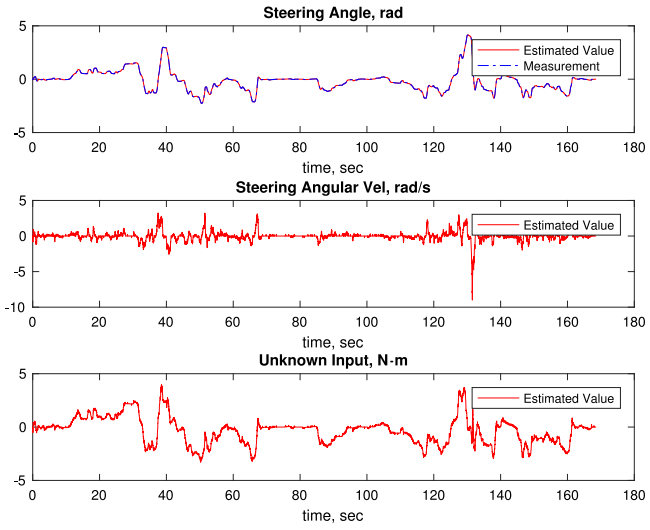
We compared the performance only over short-term prediction, i.e., prediction of the driver's torque at the current time step. Long-term predictions involve modeled dynamics of other subsystems in Fig. 1 and perform an iterative computation of the future state and the driver's steering torque as depicted in Fig. 11. The function f in (2), which the algorithms try to identify, corresponds to the perpendicular arrows from z to T_{dr}. The horizontal arrows between z correspond to the vehicle dynamics, steering column, and road geometry models. Also, the slant arrows from T_{dr} to z shows that T_{dr} at the previous time step is incorporated into an entry of z at the next time step (see (1)). To predict the future driving torque at $t = t_{k+1}$, we need to first estimate $T_{\mathrm{dr}}(t_k)$ from $z(t_k)$ using the methods outlined in Section 2. We then propagate the information in $z(t_k)$ based on a vehicle dynamics model and road geometry, and obtain the estimated value of $z(t_{k+1})$. Having obtained the estimation of $z(t_{k+1})$, we estimate $T_{\mathrm{dr}}(t_{k+1})$ using the same method as the one we employed to estimate $T_{\mathrm{dr}}(t_k)$ from $z(t_k)$. Apparently, the accuracy of the horizontal arrows (e.g., vehicle-dynamics models) affects the performance for long-term driver torque prediction. The interest of this work is, however, the comparison of human-driver control models. Comparing the models of other subsystems is beyond the scope of this paper. Therefore, we do not compare the long-term prediction performance in this work. Note that, as our steering wheel is not equipped with a torque sensor, we estimate T_{dr} via the algorithm introduced in Section 3.3.

5.2. Synthetic data generated with a human-driver control model

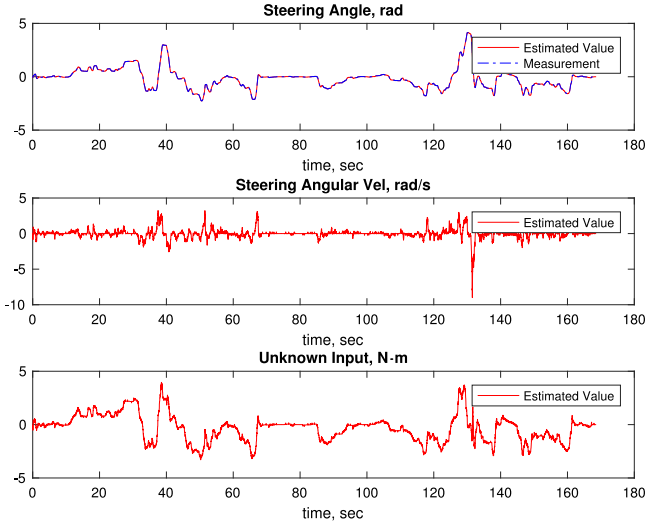
The first experiment employs data generated using CarSim® [61] with a hybrid sensorimotor two-point visual control model [6] (a two-point visual control model [4] and the expansion of the two-point visual sensorimotor model proposed in [5]). The model we employ accounts for the anticipatory control of human drivers with a model predictive controller. By employing the methods introduced in Section 2, we aim to reproduce the actions of this "human control".

Before applying the algorithms from Section 2, all inputs were normalized to achieve mean zero and standard deviation one. Then, and in order to reduce noise, we applied a first-order lowpass filter with a cutting frequency at $\omega_s = 2.5$ rad/s and a local weighted linear least squares to a second degree polynomial.

To evaluate the performance of the methods, we divided the data set into four subsets A,B,C, and D, and performed a four-fold cross validation. For instance, in sub-data set A, we employ the first 3/4 of data as the training data set and the last quarter as the test data set. In addition, we resampled the data set with $\Delta t = 0.2$ s, setting $\ell = 5$ and $d = 29$ in (1).

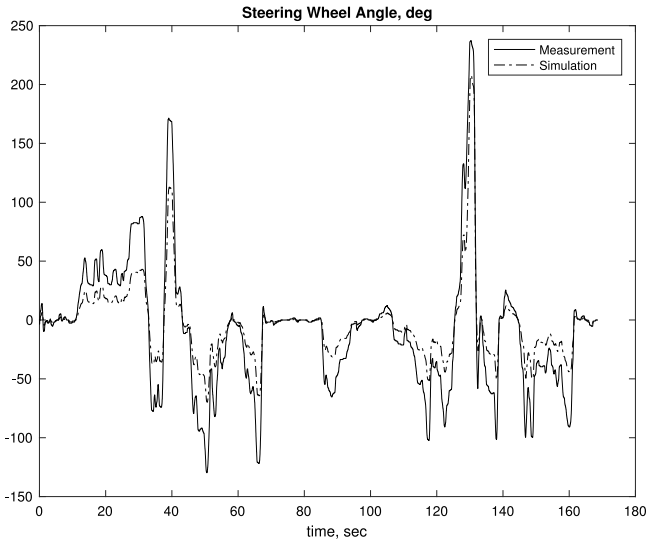


(a) With J_s and b_s estimated by joint EKF.

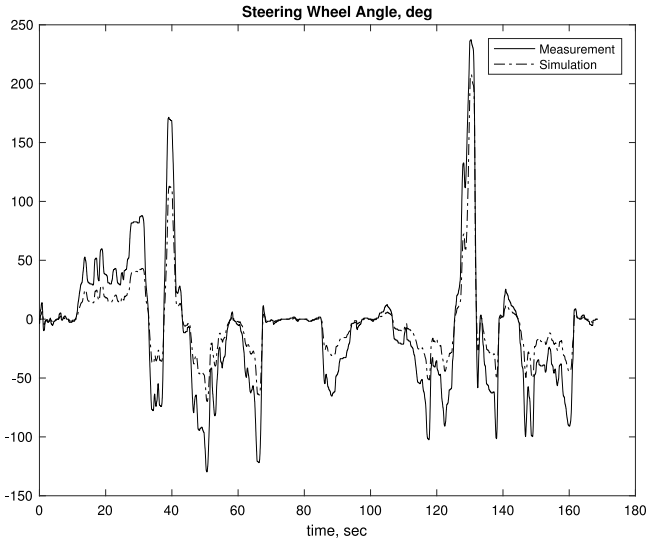


(b) With J_s and b_s estimated by joint UKF.

Fig. 9. Unknown input estimation using a PI observer with estimated J_s and b_s.



(a) Reproduction with the steering angle with joint EKF.



(b) Reproduction with the steering angle with joint UKF.

Fig. 10. Steering wheel angles of measurement and reproduction.

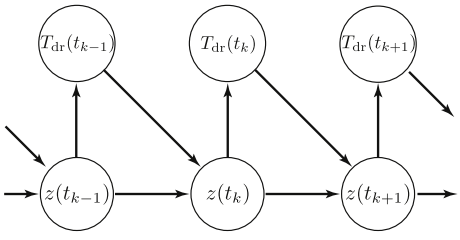


Fig. 11. The graphical model of driver behavior prediction.

Fig. 12 illustrates the RMSE of each method. We used the GMR and HMM-GMR with $N_G = 21$ and the ANN with $N_h = 2$. The largest RMSE in all the sub-data sets is exhibited by the PWC, while PWL demonstrates slightly smaller values. The GP regression exhibited almost one third of the RMSE value of the PWC and PWL. Also, GMR is competitive with GP regression for this scenario. Since the HMM-GMR considers both the spatial and sequential information of z, not surprisingly, it outperforms the GMR. Finally, the ANN is comparable with the HMM-GMR.

5.3. Human driving data

In order to verify the performance of each method against real human-driving data, we employed actual human-driver data collected using the GTDS driving simulator and compared the performance of all the methods. We pre-process the data in the same way as the synthetic data set.

Using the GTDS driving simulator, we duplicated exactly the same road circuit, the geometry of which is in Fig. 7b, employed for the previous synthetic data set in Section 5.2. Thus, we can directly compare the numerical simulations and the actual human driving experiments by eliminating the difference between simulation and experiment road geometry.

In this experiment, and similarly to the first experiment, we fix the vehicle speed at 50 km/h. Thus, the control output from the human driver is only the steering torque T_{dr}, which is then fed into

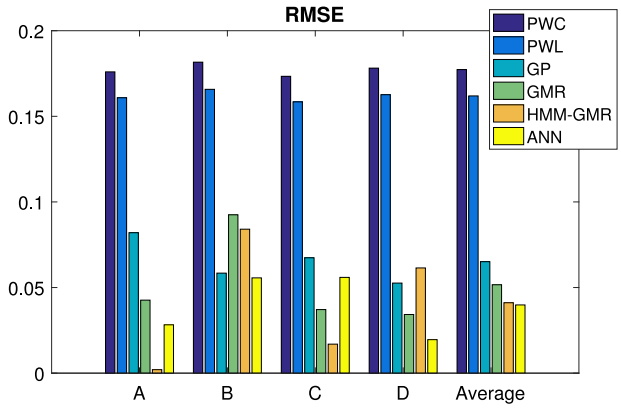


Fig. 12. The RMSE of each method for each simulated human driver data set.

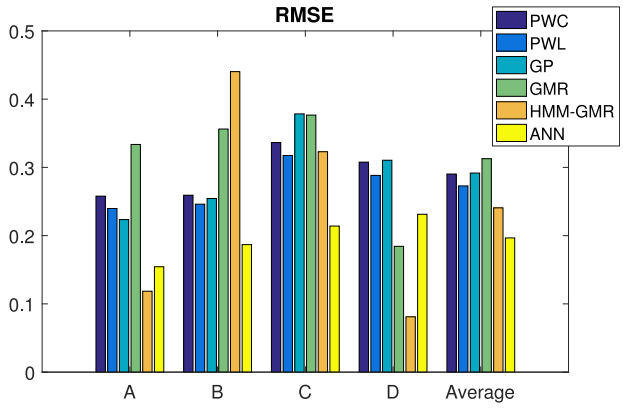


Fig. 14. The RMSE of each method for each real human driver data set.

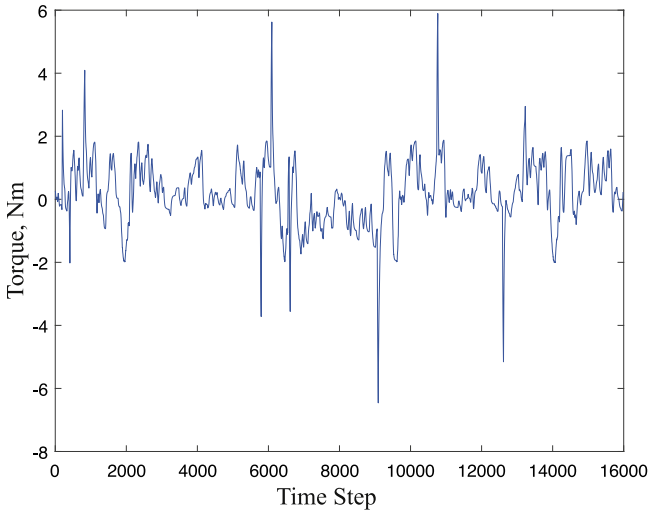


Fig. 13. Estimated driving torque to drive around the circuit clockwise (first half) and counter-clockwise (second half).

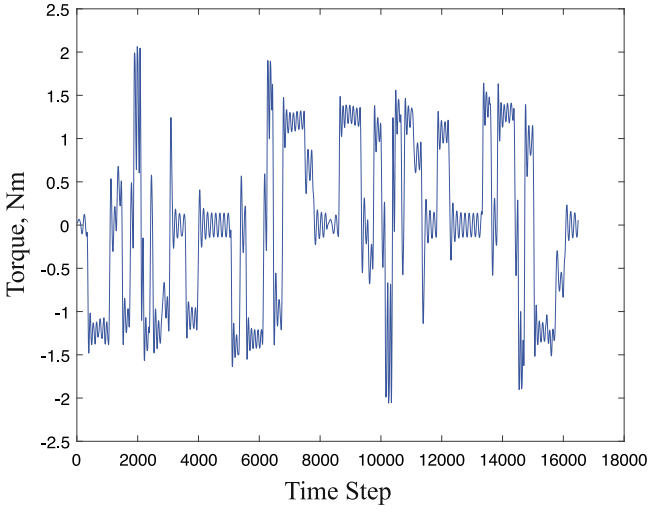


Fig. 15. The output of trained driver control model identified with dual EKF.

the steering wheel. The steering wheel outputs the steering-wheel angle δ_s. Fig. 13 depicts the estimated driving torque by the algorithm introduced in Section 3.3 while driving around clockwise and counter-clockwise.

Fig. 14 depicts the results. The GMR and HMM-GMR in this experiment had $N_G = 23$ and ANN had $N_h = 2$. The RMSE of PWC and PWL show almost the same performance as in the first experiment. In contrast, GP regression, which outperformed PWC and PWL in the first experiment, exhibits similar performance to them in this experiment. Furthermore, GMR is worse than PWC and PWL as well. The HMM-GMR error is even larger than the GMR error in sub-data set B. These results are somewhat contrary to our expectation. As these methods (i.e, GP regression, GMR, and HMM-GMR), involve the state variables (i.e., β, r, δ_s, and ρ), we expected that they will outperform PWC and PWL. By contrast, and similarly to the first experiment, the ANN outperforms the other methods. In order to investigate this counter-intuitive result, we conducted another experiment, which is described in below.

To investigate the inconsistency between the results in synthetic and real-driving data of GMR and HMM-GMR, we evaluated the performance of each method based on the output of a sensorimotor two-point visual control model, the parameters of which were identified using the real human driving data in the second experiment. We smoothed the data to eliminate nonlinearities and applied the method proposed in [59] to identify the parameters of the driver control model. Fig. 15 depicts the control input to drive around the track for the identified driver control model.

By using the identified sensorimotor two-point visual driver control model, we can evaluate the performance of the methods against the actual driving data without driving skill inconsistency. Fig. 16 depicts the regression performance of the algorithms. We observe a significant performance improvement of GMR and HMM-GMR. In addition, both GMR and HMM-GMR exhibited superior performance to the ANN model. We discuss the results in the following section.

6. Discussion

In the previous section, we observed that the HMM-GMR and ANN exhibited the least RMSE among the methods with synthetic data. With human-driver data, however, the HMM-GMR was outperformed by the ANN and even the PWC and PWL in some of the sub data set. Furthermore, with the human-driving data set, GP regression and GMR exhibited greater RMSE than PWC and PWL. Below we provide some explanation of these results.

The PWC and PWL are the simplest models. They do not use the input variables, i.e., β, r, δ_s, and ρ, and still exhibited sound performance with both synthetic and human-driving data sets. We would like to emphasize, however, that, since they ignore the information in z other than T_{dr}, their performance is expected to degrade in long-term prediction tasks.

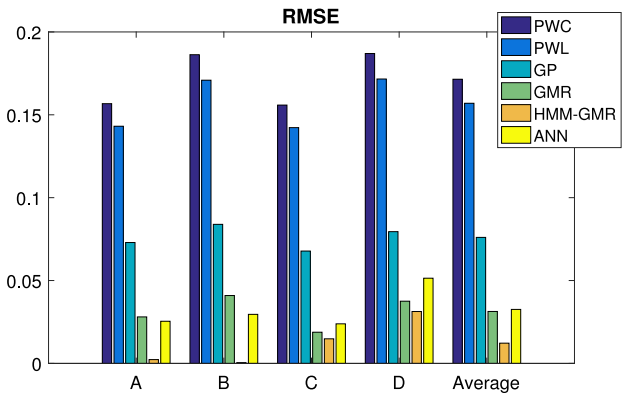


Fig. 16. The RMSE of each method for a sensorimotor two-point visual driver control model trained with a dual EKF from real human driver data set.

Among the machine-learning algorithms we evaluated, GP regression exhibited the worst performance in terms of RMSE. Furthermore, the heavy computational cost of the inverse operation in Eq. (16), the need of choosing the kernel function to use, and the number of hyper-parameters to tune, make GP a less than ideal choice to predict the driver's torque. By contrast, the HMM-GMR was competitive with the ANN in the experiment with synthetic data. The same result has also been observed in the case of human-driver longitudinal control actions [25]. Also, the HMM-GMR and ANN are competitive in terms of the implementation-complexity. In order to train GMR and HMM-GMR, we need to choose the number of Gaussians N_G. On the other hand, in order to train an ANN, we need to specify the number of hidden layers and nodes. Thus, both from the performance and implementation complexity, we regard the HMM-GMR and ANN as competitive with the synthetic data set.

In the case of the human-driving data set, we observed that ANN outperformed HMM-GMR. While the performance of HMM-GMR was worse than PWC and PWL, especially in the sub-data set B, the ANN always exhibited superior performance than those (see Fig. 14). We conjecture that this difference in the performance is because of the algorithmic robustness to the control variations of human drivers. Since human-driver control models are mathematical abstractions, when the input is the same, their response is always the same. However, human driver behavior is not consistent, especially when the situation gets outside of the comfort zone of the driver; the driver may use larger or less steering torque than needed. Our hypothesis is supported by the additional experiment, in which a sensorimotor two-point visual driver control model was first identified based on the given data, and it was used to regenerate "real" human-driving data, which does not have control variations. In this additional experiment, we observed a significant performance improvement with the GMR and HMM-GMR (see Fig. 16), which implies that GMR and HMM-GMR can exhibit accurate prediction if the human subject is a skilled driver with little control variations. The result also implies that, since ANN showed consistent performance among all the experiments, in addition to skilled human-driver subjects [27], ANN can accurately predict novice human driver behavior. We conjecture that this stable performance of our ANN is due to its simple structure, which may prevent overfitting to the training data set. From the experiments in Section 5, we may conclude that an ANN with one hidden layer and two nodes is a good model for predicting the lateral control action of non-professional normal human drivers. It should be noted that we do not claim that human driver control actions can be modeled with a simple artificial neural network. Driving consists of multiple difficult tasks such as scene understanding and decision making. In this work, we focused our attention only on the specific task of path following. Similarly to the work by Cook in [63], we conclude that for path following tasks a simple neural network exhibits an acceptable performance.

7. Summary

This paper addressed the problem of predicting lateral control actions of human drivers using six non-parameterized driver control models, namely, PWC, PWL, GP, GMR, HMM-GMR, and ANN. We briefly proposed a method to perform long-term predictions with a graphical model, and we compared the performance of the methods in terms of short-term predictions in detail. The first experiment utilized a simulated human-driving data set, and the second experiment employed a real human-driving data set collected using our GTDS driving simulator. By employing exactly the same road circuit in these two experiments, we directly compared the performance of the driver control models with both synthetic and real human driving data sets. From these experiments, we found that the ANN exhibited the most accurate predictions with both data sets, as well as the highest robustness to driving control variations. Note that, in order to perform these experiments, we needed the driver torque input, which was not measurable, because, similarly to other inexpensive driving simulators, the used GTDS driving simulator is not equipped with a steering-wheel torque sensor. Thus, this work also addressed the problem of unknown steering torque input estimation from steering-wheel-angle measurements, which is a result of independent interest. We proposed to employ joint E/UKF as steering wheel system parameter estimation algorithms and a PI observer as an unknown input estimation algorithm since the use of a dual E/UKF was not feasible. We verified the proposed algorithm both via numerical simulation and real data obtained from human driving. From both computer simulations and experiments we observed that, although the joint UKF was sensitive to parameter settings, both the joint E/UKF exhibited sound estimation performance. We also found that the PI observer accurately estimated the unknown steering torque of a human driver. The results of this paper will contribute to the development of haptic-shared-control-based ADAS using driving simulators that are not equipped with steering-torque sensors.

Future work will investigate the algorithms in more complex scenarios, such as lane change and obstacle avoidance. As the results of this work enable the use of steering torque as the output from human driver models even when using low-cost driving simulators, we also plan to investigate personalized haptic-shared-control-based ADAS for vehicle lateral control by taking advantage of the predictive power of driver control models.

Acknowledgement

This work was supported by NSF award CPS-1544814. K. Okamoto was also partially supported by Funai Foundation for Information Technology and Ito Foundation USA-FUTI Scholarship.

References

[1] R.N. Jazar, Vehicle Dynamics: Theory and Application, Springer Science & Business Media, 2013.
[2] E.A. Wan, R.V.D. Merwe, The unscented Kalman filter for nonlinear estimation, in: Adaptive Systems for Signal Processing, Communications, and Control Symposium, Alberta, Canada, 2000, pp. 153–158.
[3] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174–188.
[4] D.D. Salvucci, R. Gray, A two-point visual control model of steering, Perception 33 (10) (2004) 1233–1248.

[5] C. Sentouh, P. Chevrel, F. Mars, F. Claveau, A sensorimotor driver model for steering control, in: IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, 2009, pp. 2462–2467.

[6] K. Okamoto, P. Tsiotras, A new hybrid sensorimotor driver model with model predictive control, in: IEEE International Conference on Systems, Man, and Cybernetics, Budapest, Hungary, 2016, pp. 1866–1871.

[7] A. Pentland, A. Liu, Modeling and prediction of human behavior, Neural Comput. 11 (1) (1999) 229–242.

[8] N. Kuge, T. Yamamura, O. Shimoyama, A. Liu, A driver behavior recognition method based on a driver model framework, in: SAE Technical Paper, Tech. Rep., 2000.

[9] T. Kumagai, Y. Sakaguchi, M. Okuwa, M. Akamatsu, Prediction of driving behavior through probabilistic inference, in: Proceedings of the 8th International Conference on Engineering Applications of Neural Networks, Malaga, Spain, 2003, pp. 117–123.

[10] N. Oliver, A.P. Pentland, Driver behavior recognition and prediction in a smartcar, in: AeroSense 2000, International Society for Optics and Photonics, 2000, pp. 280–290.

[11] G.S. Aoude, V.R. Desaraju, L.H. Stephens, J.P. How, Driver behavior classification at intersections and validation on large naturalistic data set, IEEE Trans. Intell. Transp. Syst. 13 (2) (2012) 724–736.

[12] Q. Ji, P. Lan, C. Looney, A probabilistic framework for modeling and real-time monitoring human fatigue, EEE Trans. Syst. Man Cybern. A 36 (5) (2006) 862–875.

[13] N.P. Chandrasiri, K. Nawa, A. Ishii, Driving skill classification in curve driving scenes using machine learning, J. Mod. Transp. 24 (3) (2016) 196–206.

[14] K. Okamoto, K. Berntorp, S. Di Cairano, Similarity-based vehicle-motion prediction, in: American Control Conference, Seattle, WA, 2017, pp. 303–308.

[15] K. Okamoto, K. Berntorp, S. Di Cairano, Driver intention-based vehicle threat assessment using random forests and particle filtering, in: The 20th World Congress of the International Federation of Automatic Control, Toulouse, France, 2017.

[16] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[17] M. Enev, A. Takakuwa, K. Koscher, T. Kohno, Automobile driver fingerprinting, Proc. Priv. Enhancing Technol. 2016 (1) (2016) 34–50.

[18] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robot. Auton. Syst. 57 (5) (2009) 469–483.

[19] D. Nguyen-Tuong, J. Peters, M. Seeger, B. Schölkopf, Learning inverse dynamics: a comparison, in: European Symposium on Artificial Neural Networks, Bruges, Belgium, 2008, pp. 13–18.

[20] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[21] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, F. Itakura, Driver modeling based on driving behavior and its evaluation in driver identification, Proc. IEEE 95 (2) (2007) 427–437.

[22] T. Wakita, K. Ozawa, C. Miyajima, K. Takeda, Parametric versus nonparametric models of driving behavior signals for driver identification, in: International Conference on Audio-and Video-Based Biometric Person Authentication, Springer, Rye Brook, NY, 2005, pp. 739–747.

[23] M. Kuderer, S. Gulati, W. Burgard, Learning driving styles for autonomous vehicles from demonstration, in: IEEE International Conference on Robotics and Automation, Seattle, WA, 2015, pp. 2641–2646.

[24] B.D. Ziebart, A. Maas, J.A. Bagnell, A.K. Dey, Maximum entropy inverse reinforcement learning, in: AAAI Conference on Artificial Intelligence, Chicago, IL, 2008, pp. 1433–1438.

[25] S. Lefèvre, C. Sun, R. Bajcsy, C. Laugier, Comparison of parametric and non-parametric approaches for vehicle speed prediction, in: American Control Conference, Portland, OR, 2014, pp. 3494–3499.

[26] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, A. Billard, A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation, IEEE Robot. Autom. Mag. 17 (2) (2010) 44–54.

[27] H. Wei, W. Ross, S. Varisco, P. Krief, S. Ferrari, Modeling of human driver behavior via receding horizon and artificial neural network controllers, in: IEEE 52nd Annual Conference on Decision and Control, Florence, Italy, 2013, pp. 6778–6785.

[28] S. Di Cairano, D. Bernardini, A. Bemporad, I.V. Kolmanovsky, Stochastic MPC with learning for driver-predictive vehicle control and its application to HEV energy management, IEEE Trans. Control Syst. Technol. 22 (3) (2014) 1018–1031.

[29] S. Lefèvre, A. Carvalho, F. Borrelli, Autonomous car following: a learning-based approach, in: Intelligent Vehicles Symposium, Seoul, South Korea, 2015, pp. 920–926.

[30] C. Liu, A. Gray, C. Lee, J.K. Hedrick, J. Pan, Nonlinear stochastic predictive control with unscented transformation for semi-autonomous vehicles, in: American Control Conference, Portland, OR, 2014, pp. 5574–5579.

[31] V.A. Shia, Y. Gao, R. Vasudevan, K.D. Campbell, T. Lin, F. Borrelli, R. Bajcsy, Semiautonomous vehicular control using driver modeling, IEEE Trans. Intell. Transp. Syst. 15 (6) (2014) 2696–2709.

[32] D.A. Abbink, M. Mulder, Neuromuscular analysis as a guideline in designing shared control, in: Advances in Haptics, InTech, 2010.

[33] K. Driggs-Campbell, V. Shia, R. Bajcsy, Improved driver modeling for human-in-the-loop vehicular control, in: IEEE International Conference on Robotics and Automation, Seattle, WA, 2015, pp. 1654–1661.

[34] R. Hess, A. Modjtahedzadeh, A control theoretic model of driver steering behavior, IEEE Control Syst. Mag. 10 (5) (1990) 3–8.

[35] A.Y. Ungoren, H. Peng, An adaptive lateral preview driver model, Vehicle Syst. Dyn. 43 (4) (2005) 245–259.

[36] A.J. Pick, D.J. Cole, A mathematical model of driver steering control including neuromuscular dynamics, J. Dyn. Syst. Meas. Control 130 (3) (2008) 031004.

[37] L. Saleh, P. Chevrel, F. Claveau, J.-F. Lafay, F. Mars, Shared steering control between a driver and an automation: Stability in the presence of driver behavior uncertainty, IEEE Trans. Intell. Transp. Syst. 14 (2) (2013) 974–983.

[38] S. Zafeiropoulos, P. Tsiotras, Design of a lane-tracking driver steering assist system and its interaction with a two-point visual driver mode, in: American Control Conference, Portland, OR, 2014, pp. 3911–3917.

[39] L. Saleh, P. Chevrel, J.-F. Lafay, Optimal control with preview for lateral steering of a passenger car: Design and test on a driving simulator, in: Time Delay Systems: Methods, Applications and New Trends, Springer, 2012, pp. 173–185.

[40] D.A. Abbink, M. Mulder, E.R. Boer, Haptic shared control: smoothly shifting control authority? Cognit. Technol. & Work 14 (1) (2012) 19–28.

[41] E. Rendon-Velez, P. Van Leeuwen, R. Happee, I. Horváth, W. Van der Vegte, J. De Winter, The effects of time pressure on driver performance and physiological activity: a driving simulator study, in: Transportation Research Part F: Traffic Psychology and Behaviour, Vol. 41, 2016, pp. 150–169.

[42] D. Sadigh, S.S. Sastry, S.A. Seshia, A. Dragan, Information gathering actions over human internal state, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Korea, 2016, pp. 66–73.

[43] F. Mars, M. Deroo, J.-M. Hoc, Analysis of human-machine cooperation when driving with different degrees of haptic shared control, IEEE Trans. Haptics 7 (3) (2014) 324–333.

[44] K. Okamoto, P. Tsiotras, A comparative study of data-driven human driver lateral control models, in: American Control Conference, Milwaukee, WI, 2018, pp. 3988–3993.

[45] M. Bojarski, D.D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316, 2016.

[46] C. Chen, A. Seff, A. Kornhauser, J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in: IEEE International Conference on Computer Vision, Santiago, Chile, 2015, pp. 2722–2730.

[47] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Selected Papers of Hirotugu Akaike, Springer, 1998, pp. 199–213.

[48] G. Schwarz, et al., Estimating the dimension of a model, Ann. Statist. 6 (2) (1978) 461–464.

[49] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.

[50] D. Koenig, S. Mammar, Design of proportional-integral observer for unknown input descriptor systems, IEEE Trans. Automat. Control 47 (12) (2002) 2057–2062.

[51] D. Söffker, T.-J. Yu, P.C. Müller, State estimation of dynamical systems with nonlinearities by using proportional-integral observer, Int. J. Syst. Sci. 26 (9) (1995) 1571–1582.

[52] D. Koenig, Unknown input proportional multiple-integral observer design for linear descriptor systems: application to state and fault estimation, IEEE Trans. Automat. Control 50 (2) (2005) 212–217.

[53] K. Yamamoto, D. Koenig, O. Sename, P. Moulaire, Driver torque estimation in electric power steering system using an H_∞/H_2 proportional integral observer, in: EEE Conference on Decesion and Control, Osaka, Japan, 2015, pp. 843–848.

[54] S. Hui, S.H. Żak, Observer design for systems with unknown inputs, Int. J. Appl. Math. Comput. Sci. 15 (4) (2005) 431–446.

[55] Y. Xiong, M. Saif, Unknown disturbance inputs estimation based on a state functional observer design, Automatica 39 (8) (2003) 1389–1398.

[56] J.N. Yang, S. Pan, H. Huang, An adaptive extended Kalman filter for structural damage identifications ii: unknown inputs, Struct. Control Health Monit. 14 (3) (2007) 497–521.

[57] E. Ghahremani, I. Kamwa, Dynamic state estimation in power system by applying the extended Kalman filter with unknown inputs to phasor measurements, IEEE Trans. Power Syst. 26 (4) (2011) 2556–2566.

[58] E.A. Wan, R. van der Merwe, A.T. Nelson, Dual estimation and the unscented transformation, in: S.A. Solla, T.K. Leen, K. Müller (Eds.), Advances in Neural Information Processing Systems 12, Denver, CO: MIT Press, 1999, pp. 666–672.

[59] C. You, J. Lu, P. Tsiotras, Nonlinear driver parameter estimation and driver steering behavior analysis for ADAS using field test data, IEEE Trans. Hum.-Mach. Syst. 47 (5) (2017) 686–699.

[60] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, no. 3.2, Kobe, Japan, 2009.

[61] Mechanical Simulation Inc. "Carsim," 2009. [Online]. Available: www.carsim.com.

[62] Unity Technologies, "Unity3D." [Online]. Available:.

[63] M. Cook, It takes two neurons to ride a bicycle, in: Demonstration at NIPS, 2004.

Kazuhide Okamoto earned his bachelor's and master's degrees in aeronautics and astronautics from the University of Tokyo, Japan. Currently working on his doctoral degree in aerospace engineering at the Georgia Institute of Technology, Atlanta, Georgia, he is researching stochastic optimal control and its application to human-vehicle systems. He was also a summer research intern at Mitsubishi Electric Research Laboratories in Boston, Massachusetts, and at Honda Research Institute USA in Mountain View, California.

Panagiotis Tsiotras is the David and Andrew Lewis Chair Professor in the School of Aerospace Engineering at the Georgia Institute of Technology (Georgia Tech), and the Director of the Dynamics and Controls Systems Laboratory (DCSL) in the same school as well as an Associate Director of the Institute for Robotics and Intelligent Machines at Georgia Tech. He received his Ph.D. degree in Aeronautics and Astronautics from Purdue University in 1993 and also holds degrees in Mathematics and Mechanical Engineering. He has held visiting research appointments at MIT, JPL, INRIA Rocquencourt, and Mines ParisTech. His research interests include optimal control of nonlinear systems and ground, aerial and space vehicle autonomy.

He has served in the Editorial Boards of the Transactions on Automatic Control, the IEEE Control Systems Magazine, the AIAA Journal of Guidance, Control and Dynamics and the journal Dynamics and Control. He is the recipient of the NSF CAREER award and the Outstanding Aerospace Engineer award from Purdue. He is a Fellow of AIAA and IEEE, and a member of the Phi Kappa Phi, Tau Beta Pi, and Sigma Gamma Tau Honor Societies.