Gaussian Process-Based Personalized Adaptive Cruise Control

Yanbing Wang, Ziran Wang, *Member, IEEE,* Kyungtae Han, *Senior Member, IEEE,* Prashant Tiwari, and Daniel B. Work, *Member, IEEE*

***Abstract*—Advanced driver-assistance systems (ADAS) have matured over the past few decades with the dedication to enhance user experience and gain a wider market penetration. However, personalization features, as an approach to make the current technologies more acceptable and trustworthy for users, have been gaining momentum only very recently. In this work, we aim to learn personalized longitudinal driving behaviors via a Gaussian Process (GP) model. The proposed method learns from individual driver's naturalistic car-following behavior, and outputs a desired acceleration profile that suits the driver's preference. The learned model, together with a predictive safety filter that prevents rear-end collision, is used as a personalized adaptive cruise control (PACC) system. Numerical experiments show that GP-based PACC (GP-PACC) can almost exactly reproduce the driving styles of an intelligent driver model. Additionally, GP-PACC is further validated by human-in-the-loop experiments on the Unity game engine-based driving simulator. Trips driven by GP-PACC and two other baseline ACC algorithms with driver override rates are recorded and compared. Results show that on average, GP-PACC reduces the human override duration by 60% and 85% as compared to two widely-used ACC models, respectively, which shows the great potential of GP-PACC in improving driving comfort and overall user experience.**

***Index Terms*—Gaussian Process, adaptive cruise control, car following, personalization, driving behavior**

I. Introduction

A. Motivation

PERSONALIZATION has gained increased attention in the automotive industry. However, the industry-level personalized features are very limited and mostly at a complimentary level, such as seat position, radio station tuning, etc. Personalization on vehicle maneuvers such as path tracking, steering and car-following is less developed, yet implicit driving preference significantly impacts driver's acceptance and trust towards the existing advanced driver-assistance systems (ADAS) [1].

As one of the most common ADAS functionalities, adaptive cruise control (ACC) automatically adjusts the longitudinal speed of the ego vehicle to maintain a safe distance from the vehicle ahead. ACC has been shown to increase safety, enhance driving comfort, and reduce fuel consumption [2]–[5]. However, the limited settings of ACC prevent the drivers to preserve their own car-following styles, resulting in lack of trust and usage of that technology. In addition, a variety of usage conditions and the changing of the drivers' expectations persist in real-world driving. Drivers differ in their preferences and skills, and their styles may change over time. Therefore, personalization in ACC has the potential to capture the adapting preference of drivers, and adjusts the settings to suit their needs.

Y. Wang and D. B. Work are with Department of Civil and Environmental Engineering, and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212 (e-mails: yanbing.wang@vanderbilt.edu; dan.work@vanderbilt.edu).

Z. Wang, K. Han, and P. Tiwari are with Toyota Motor North America R&D, InfoTech Labs, 465 Bernardo Avenue, Mountain View, CA 94043 (e-mails: ryanwang11@hotmail.com; kyungtae.han@toyota.com; prashant.tiwari@toyota.com).

B. Related Work

Some of the most common longitudinal control laws are derived from physics-based control policies, which can be expressed as an ordinary differential equation (ODE) $\dot{v} = f(s, v, u)$ that describes the car-following behaviors given the space gap s, ego vehicle speed v and leader's speed u. Examples include the optimal velocity model (OVM) [6], the intelligent driver model (IDM) [7], the Gipps model [8], and the Gazis-Herman-Rothery (GHR) model [9]. The ODE-based car-following models can work in traffic microsimulation, and provide provable and interpretable properties such as rational driving, stability [10] and identifiability [11]. However, human driving does not strictly follow these pre-defined rules, and contains subtleties that cannot be fully captured by the analytical expressions. To this end, learning-based approaches are becoming a popular modeling paradigm.

The question of learning individual driving style naturally motivates us to adopt a supervised-learning approach, where a certain control action is learned from demonstration. A popular approach to manipulate a system towards terminal goal is through reinforcement learning (RL) [12], [13] where a control policy is learned by maximizing a reward function that describes the system evolution. Extensions of RL include inverse reinforcement learning, which learns a reward function through expert demonstration [14]. In more complicated robotics where the system evolution is unknown, the control tasks are often coupled with system identification to achieve various goals such as navigation, path finding, disturbance rejection, and etc. [15]–[17]. A notion of underactuation in control has recently been studied to design controllers that take advantage of the natural dynamics of the system [18]. Other data-driven system identification tools such as SINDy [19], Gaussian Process (GP) [20] and Neuro-fuzzy methods [21] are becoming popular to identify unknown and complex systems. These tools are often coupled with existing controllers such as Model Predictive Control (MPC) to enhance control performance and to achieve robust behaviors.

The learning-based control design benefits from the exploratory data-driven tools, as opposed to the model-based system identification and control which are often based on a fixed model structure. However, challenges still persist such as the verification of safety, stability and rationality. Recent developments such as the control Lyapunov and control barrier functions have been applied to provide safe and stable controlled systems [22], [23], as well as formal verification tools to facilitate assured autonomy from learning [24]–[26].

Amongst all the control design approaches, we draw particular attention to GP regression, to design an ACC system that mimics personalized driving behavior and increase drivers' acceptance and trust on the system. There are several rationales for using GP models in system identification: (a) Physics-based model structure can be simplified due to the data-driven nature of GP. (b) Bayesian treatment relies on marginal likelihood, which reduces the risk of overfitting. (c) Limited amount of data relative to the number of regressors makes GP model suitable to deal with data inadequacy and measurement noise [27]. Instead of the explicit personalization (i.e., offering drivers to choose from a number of predefined system settings), we focus on the implicit personalization (estimating the drivers' preferences based on their past behaviors) [28]. GP regression can be utilized to identify the relationship between input (driver's perceived information) and output (desired acceleration), and hence provides personalized guidance towards driving.

C. Contributions

Compared to our preliminary work [29] where a GP-PACC is designed to learn personalized implicit car-following styles without categorizing based on predefined rules, in this paper we enhance the results with the following additions:

1) A predictive safety filter is developed at the downstream of GP-PACC, which guarantees that the acceleration command will lead to the safe state. This step makes as little modification to the GP output as possible, preserving the personalized features while guaranteeing safety.
2) The proposed GP-PACC is validated on both the synthetic car-following data and the naturalistic data collected from human-in-the-loop experiments. Results show that GP-PACC can recover both the synthetic and naturalistic car-following data even under reasonable measurement noises, and it outperforms established car-following models by reducing the human takeover rate up to 85%.

The remainder of this work is organized as follows. Section II introduces the problem formulation of this study. Section III outlines the fundamentals of GP regression used to model car-following behaviors, and describes the training and validation method for the GP model. In Section IV we conduct numerical experiments and human-in-the-loop experiments to test the validity of the model. Finally, the study is concluded with some future directions in Section V.

II. Problem formulation

A. Notation

The state of the controlled car-following system at time k is $x_k = [s_k, v_k]^T$, which is composed of the space gap s_k and ego vehicle's speed v_k. We denote u_k as the lead vehicle's speed at time k, and y_k as the ego vehicle acceleration at time k. The uniform sampling timestep Δt is used to discretize the system, which runs in N timesteps in total. The nonlinear mapping $f_{\text{CF}} : \mathbb{R}^3 \to \mathbb{R}^1$ represents the car-following dynamics, and will be learned by the GP model.

B. Assumptions and Specifications

In this paper we focus on personalized ACC design, i.e., the longitudinal control of a vehicle based on the driver's car-following preference. The scope of this work is focused on the upper-level controller, where the output is the command acceleration that triggers the movement of the vehicle. We assume that the low-level vehicle dynamics has no actuator lags or delays for simplicity, and the personalized control design problem is formulated as a system identification problem. We design a data-driven GP-PACC such that the controlled car-following dynamic matches the individual driver's naturalistic car-following style. Furthermore, we consider only the car-following mode for all the ACCs discussed in this paper, i.e., the leader is always present.

C. The Car-Following Dynamics

The driver's longitudinal acceleration depends on the vehicle state in relation to the lead vehicle, characterized by a car-following model:

$$\dot{v}(t) = f_{\text{CF}}(s(t), v(t), u(t)). \tag{1}$$

The ego vehicle's dynamics will be updated in discrete time:

$$x_{k+1} = \begin{bmatrix} s \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} s_k + (u_k - v_k)\Delta t \\ v_k + f_{\text{CF}}(s_k, v_k, u_k)\Delta t \end{bmatrix} \tag{2}$$

where $f_{\text{CF}} : \mathbb{R}^3 \to \mathbb{R}^1$ will be trained with a GP model. GP-PACC is trained to achieve personalized car-following behavior by minimizing the difference between the predicted acceleration and the recorded naturalistic driving acceleration.

The block diagram of the proposed GP-PACC system is shown in Fig. 1. We consider the ACC algorithm as the high-level controller, which takes the input of the ego vehicle speed, lead vehicle speed, and space gap information, and outputs an acceleration command. The low-level vehicle dynamics will then output the corresponding speed and space gap.

III. Methodology

In this section, we briefly introduce GP regression both as a modeling tool and as a controller that will be used to model personalized car-following behavior. GP regression has been discussed in many standard textbooks such as [27], [30], [31]. Here we only outline it briefly in section III-A. Next we describe the specific training and validation procedures of GP-PACC in section III-B and III-C, respectively.

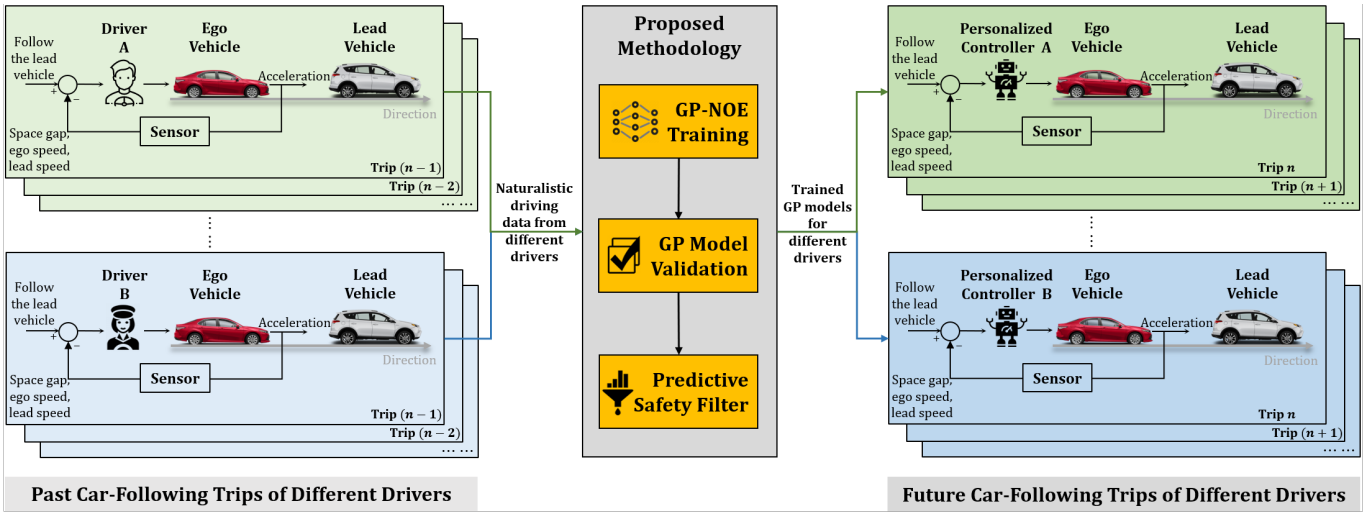


Fig. 1. Block diagram of the proposed GP-PACC system.

A. GP Regression to Model Car-Following Dynamics

In this paper, we use GP regression to model the personalized longitudinal acceleration, namely the mapping from each driver's perceived states (e.g., space gap and the relative speed) to the actual acceleration of the vehicle.

Gaussian processes extend multivariate Gaussian distributions to infinite dimensionality. They are a form of supervised learning and the training result represents a nonlinear mapping $f_{\text{GP}}(\mathbf{z}) : \mathbb{R}^{\dim(\mathbf{z})} \to \mathbb{R}$, such as (1). The mapping between the input vector $\mathbf{z}$ and the function value $f_{\text{GP}}(\mathbf{z})$ is accomplished by the assumption that $f_{\text{GP}}(\mathbf{z})$ is a random variable and is jointly Gaussian distributed with $\mathbf{z}$, which is also assumed to be a random variable [30].

a) Setup: The GP model setup includes selecting the model regressors, the mean function and the covariance function. In the following discussion, we focus on the commonly used zero-mean and the squared-exponential covariance function that relates two sample input vectors $\mathbf{z}_i$ and $\mathbf{z}_j$:

$$c(\mathbf{z}_i, \mathbf{z}_j) = \sigma_f^2 \text{exp}(-\frac{1}{2}(\mathbf{z}_i - \mathbf{z}_j)^T P^{-1}(\mathbf{z}_i - \mathbf{z}_j)) + \sigma_n^2 \delta_{ij}, \quad (3)$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise, and $P = \text{diag}[l_1^2, ..., l_{\dim(\mathbf{z})}^2]$ contains the characteristic length scale for each dimension of the input vector. The hyperparameters of the covariance function $\theta = [\sigma_f, \sigma_n, l_1, ..., l_{\dim(\mathbf{z})}]^T$ include the measurement noise σ_n, the process standard deviation σ_f, and the characteristic length scales, which are learned by maximizing the likelihood of the observation.

b) Bayesian model inference: The inference of a Bayesian model is a process where the prior knowledge of the hyperparameter vector θ is updated to a posterior distribution through the identification (training) data.

We specify the training input $\mathbf{Z}$ and target $\mathbf{y}$ for a total of N samples:

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N]^T \quad (4)$$

$$\mathbf{y} = [y_1, y_2, ..., y_N]^T, \quad (5)$$

where the subscript denotes the sample index.

The corresponding GP model can be used for predicting the function value y_* given a new input $\mathbf{z}_*$ based on a set of past observations $\mathcal{D} = \{\mathbf{Z}, \mathbf{y}\}$. The key assumption is that the data can be represented as a sample from a multivariate Gaussian distribution:

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right), \quad (6)$$

where $\mathbf{0} \in \mathbb{R}^N$ is a vector of zeros, and K is the covariance matrix

$$K = \begin{bmatrix} c(\mathbf{z}_1, \mathbf{z}_1), c(\mathbf{z}_1, \mathbf{z}_2)...c(\mathbf{z}_1, \mathbf{z}_N) \\ c(\mathbf{z}_2, \mathbf{z}_1), c(\mathbf{z}_2, \mathbf{z}_2)...c(\mathbf{z}_2, \mathbf{z}_N) \\ ..., ... \\ c(\mathbf{z}_N, \mathbf{z}_1), c(\mathbf{z}_N, \mathbf{z}_2)...c(\mathbf{z}_N, \mathbf{z}_N) \end{bmatrix} \quad (7)$$

$$K_* = [c(\mathbf{z}_*, \mathbf{z}_1), c(\mathbf{z}_*, \mathbf{z}_2)...c(\mathbf{z}_*, \mathbf{z}_N)] \quad K_{**} = c(\mathbf{z}_*, \mathbf{z}_*). \quad (8)$$

We want to infer θ by computing the posterior distribution of the hyperparameters:

$$p(\theta|\mathbf{Z}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{Z}, \theta)p(\theta)}{p(\mathbf{y}|\mathbf{Z})}. \quad (9)$$

For unknown knowledge of θ, it is reasonable to specify a uniform distribution $p(\theta)$, and as a result, the posterior distribution is proportional to the marginal likelihood, i.e.,

$$p(\theta|\mathbf{Z}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{Z}, \theta). \quad (10)$$

Maximizing the posterior distribution is equivalent to minimizing the negative log likelihood $l(\theta)$:

$$l(\theta) := \ln\, p(\mathbf{y}|\mathbf{Z}, \theta) = -\frac{1}{2}\ln|K| - \frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{N}{2}\ln(2\pi). \quad (11)$$

Once the best-fit θ is obtained, we can compute the covariance matrix (7) and the output distribution y_* (in terms of the prediction mean and variance) given a new input vector $\mathbf{z}_*$:

$$\begin{aligned} \hat{y}_* &= K_* K^{-1}\mathbf{y} \\ \text{var}(y_*) &= K_{**} - K_* K^{-1} K_*^T. \end{aligned} \quad (12)$$

For the simplicity of notation, we denote the output prediction as:

$$y_* = f_{GP}(\mathbf{z}_*, \theta) + \mathcal{N}(0, \sigma_n^2). \tag{13}$$

The run time for GP regression is $\mathcal{O}(N^3)$ due to the matrix inversion. However, the naive GP regression does not lead to closed-loop stability of the car-following system [32]. The training samples are assumed to be independent in time, i.e., the covariance function $c(\mathbf{z}_i, \mathbf{z}_j)$ considers the similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$ only in terms of value, not in terms of time. When using the naive GP model as a dynamical model for simulation, the error will accumulate. Therefore, it is important to consider dynamics (2) in model training as the previous prediction y_k will affect the future states x_{k+1}. We adopt a nonlinear output-error (NOE) approach to improve the training accuracy, where the training process is described in the next subsection.

B. GP-NOE Training

We adopt a training process similar to calibrating an ODE-based car-following model [33]–[37]. The process is to find the parameters of which the simulated output is closest to the recorded measurement. The simulated state $\{\hat{x}_k = [\hat{s}, \hat{v}]_k\}_{k=1}^N$ given the initial state $x_0 = [s_0, v_0]$, the external input signal $u_{0:N-1}$ and a GP model (13) are used as part of the pseudo training input of the GP-NOE model. This way, the dynamics (2) can be inherently included in the training when the simulated states are fed back as the regressors. The simulated state can be obtained via:

$$\begin{aligned} \hat{x}_{k+1} &= \begin{bmatrix} \hat{s} \\ \hat{v} \end{bmatrix}_{k+1} = \begin{bmatrix} \hat{s}_k + (u_k - \hat{v}_k)\Delta t \\ \hat{v}_k + f_{GP}(\hat{\mathbf{z}}_k, \theta)\Delta t \end{bmatrix} \\ \hat{x}_0 &= x_0 = [s_0, v_0], k = 0 : N-1 \end{aligned} \tag{14}$$

where $\hat{\mathbf{z}}_k = [\hat{s}_k, \hat{v}_k, u_k]$ is the k^{th} sample of the pseudo training input, which contains the simulated state and the measured external input at time k, as opposed to the recorded data $\mathbf{z}_k = [s_k, v_k, u_k]$. We rewrite (14) as the following to denote the simulated trajectories:

$$\hat{x}_{k+1} = g(\hat{x}_k, u_k, \theta, \Delta t), k = 0 : N-1. \tag{15}$$

The simulation also requires an initial guess of the hyper-parameters θ. The mean prediction is stated as $f_{GP}(\hat{\mathbf{z}}_k, \theta)$ according to (12). The training target is the acceleration data at the same timestep $\mathbf{y}_{1:N}$.

Let us denote $\hat{\mathbf{Z}}_{1:N} = [\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, ..., \hat{\mathbf{z}}_N]^T$. The training of the GP model with NOE structure is an iterative process shown in **Algorithm 1**. The implementation is based on the GP-Model-based System-Identification Toolbox for Matlab [38].

Remarks: Dynamical models trained using GP-NOE are empirically shown to produce closed-loop stability, which the naive GP regression usually fails to achieve due to the lack of time-dependency. In our experiments we found that it is sufficient to use as few as 600 continuous data samples (60 sec of 10Hz data) to train a reasonable GP-PACC. Note that the GP output is a Gaussian distribution on the acceleration prediction. The prediction mean is used as the actual command acceleration, and the uncertainty is purely for understanding the prediction confidence. Although the loss function for GP-NOE is much more complicated than that of the naive GP regression (see a visual illustration in [32]) and often the globally optimal θ cannot be guaranteed to be found, a good initial guess on θ can help to reach faster convergence and more accurate data-fitting results. This feature makes GP-NOE suitable for adaptive training on small batches of data. In other words, the model can be improved overtime when initialized with previously trained hyperparameters. In addition, the adaptive training process can incorporate the changes of driving styles due to, for example, external conditions (e.g., road, traffic, weather) and internal conditions (e.g., moods, skills).

Algorithm 1: GP-NOE training

Data: Training input $\mathbf{Z}$, training target $\mathbf{Y}$, covariance function $c(\cdot, \cdot)$, initial hyperparameters θ, initial condition $x_0 = [s_0, v_0]$

while $l(\theta)$ (11) *is not minimal* **do**
  obtain the simulated (pseudo) regression vectors $\hat{\mathbf{Z}}_{1:N}$ with the initial state $x_0 = [s_0, v_0]$ and the current hyperparameters θ, according to (14);
  update θ by minimizing the negative log likelihood $l(\theta)$.
end

C. GP Model Validation

Training a GP-NOE model is similar to calibrating a car-following model, which is conducted by finding the model parameters that minimize the error between the *simulated* vehicle trajectories and the benchmark. We validate the GP model in simulation, i.e., obtaining a closed-loop simulated trajectory according to (14), and compare the acceleration and space-gap trajectories with the recorded data, similar to evaluating a car-following model from calibration (e.g., [33]–[37], [39]).

Two performance metrics are measured: the mean squared error (MSE) and the log predictive-density error (LPD) [27], [40] between the GP simulated acceleration and the recorded acceleration of a validation data set:

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 \\ LPD &= \frac{1}{2}\ln(2\pi) + \frac{1}{2N}\sum_{k=1}^{N}\left(\ln(\sigma_k^2) + \frac{(y_k - \hat{y}_k)^2}{\sigma_k^2}\right) \end{aligned} \tag{16}$$

where y_k is the acceleration data at timestep k, $\hat{y}_k$ is the mean prediction of GP at timestep k, and σ_k^2 is the prediction variance. MSE measures the error only on the mean predicted acceleration, whereas LPD takes into account the entire distribution of the prediction by penalizing the overconfident prediction (smaller variance) more than the acknowledged bad predicting values (higher variance). In addition, the MSE on the space gap (MSE-s) will also be calculated, since small and biased acceleration prediction might lead to a larger space gap error. The simulated space gap can be obtained from the GP output using (14). The lower these measures the better GP

model performs in terms of recovering the original driving data.

D. Predictive Safety Filter

A purely data-driven control approach such as GP does not explicitly take the driving safety into account. Throughout literature, we found learning-based control achieves "safe-by-design" with verified control envelopes [41], fixed-point computations of certain set-valued mappings [42], safety filtering [43], etc. In this paper we adopt a predictive safety filtering approach based on [43], which is an MPC-like method that solves an optimization problem in a receding horizon fashion: given the GP-PACC model hyperparameters θ, the initial states and safety constraints, minimize the modification of the m-step GP-predicted acceleration. We consider the following notion of safety:

Definition 1: The state of a car-following system is considered *safe* if there exists an admissible deceleration profile such that the ego vehicle can avoid rear-end collision with the leader for all time, with the assumptions that 1) the leader vehicle travels at a constant speed u and 2) the set of admissible deceleration cannot be lower than the minimum allowable deceleration $a_{\min}$.

This safety can be guaranteed if there exists a non-empty safe set $\mathcal{S}_{k^*}$ composed of the space gap s_{k^*} and the relative velocity $\Delta v_{k^*} = u - v_{k^*}$ at time k^* such that rear-end collision can be avoided for all future time $T \in [k^*, \infty)$. The derivation of safe set under above assumptions can be found in the Appendix.

The safety filter formulation is as follows:

$$\begin{aligned}
\text{minimize} \sum_{i=0}^{m} & \|\hat{y}_{k,i} - y_{k,i}\|^2 \\
\text{s.t. } & \hat{s}_{k,0} := s_k^* \\
& \hat{v}_{k,0} := v_k^* \\
& u_{k,0} := u_k^* \\
& \forall i = 0,1,...,m: \\
& \hat{y}_{k,i} = f_{GP}(\hat{\mathbf{z}}_{k,i}, \theta) \\
& \hat{x}_{k,i+1} = [\hat{s}_{k,i+1}, \hat{v}_{k,i+1}]^T = g(\hat{x}_{k,i}, u_{k,i}, \theta, \Delta t) \\
& u_{k,i+1} = u_{k,i} \\
& \Delta \hat{v}_{k,i} = u_{k,i} - \hat{v}_{k,i} \\
& (\hat{s}_{k,i}, \Delta \hat{v}_{k,i}) \in \mathcal{S}_{k,i}
\end{aligned} \tag{17}$$

A feasible solution of (17) for each GP output can guarantee that the states for the next m time steps fall into the safe set. ►YW: end modified◄

IV. Experiments and Results

In this section, the validation of GP-PACC is conducted on two different data sets. The first one is the synthetically generated data from a car-following model, with various additive noise levels on the acceleration to emulate realistic sensor errors. The second one is human-driving car-following data generated by the Unity game engine-based driving simulator, which introduces a more naturalistic driving scenario.

A. Numerical Experiments

A set of car-following data is synthetically generated using IDM [7], which has been used throughout the literature to model a realistic driver behavior, such as asymmetric accelerations and decelerations. The acceleration is expressed as:

$$\begin{aligned}
\dot{v}(t) &= f_{\text{CF}}(s(t), v(t), u(t)) \\
&= a\left[1 - \left(\frac{v(t)}{v_f}\right)^{\delta} - \left(\frac{s^*(v(t), u(t))}{s(t)}\right)^2\right]
\end{aligned} \tag{18}$$

where the desired space gap s^* is defined as:

$$s^*(v(t), u(t)) = s_j + v(t)T + \frac{v(t)(v(t) - u(t))}{2\sqrt{ab}}. \tag{19}$$

The parameters of the model are the acceleration exponent δ, free-flow speed v_f, the desired time headway T, the jam distance s_j, the maximum acceleration a and the desired deceleration b. In this experiment, the synthetic data is obtained from an IDM with parameters $\theta = [s_j, v_f, T, a, b, \delta] = [2, 33.3, 1.6, 0.73, 1.67, 4]$ based on empirical investigations [7].

We generate 200 seconds of data at 10Hz given a pre-recorded, freeway high-speed lead vehicle speed profile ranging between 25m/s to 35m/s. The simulated data is also manually polluted with Gaussian white noise ranging from 0.01 to 0.1 standard deviation onto the acceleration signal, in order to emulate the realistic sensor errors. We train the GP model on the first 100 seconds and use the second half as the validation set (see Fig. 3). This composition is shown to reproduce the car-following styles for various drivers consistently well in our later experiments.

Fig. 3 visualizes the GP simulated acceleration (red solid line) and the benchmark data (black dashed line), as well as the prediction uncertainty (grey area). The data is synthetically generated using IDM. One can see that the uncertainty band well captures the deviation of the data set, and the mean prediction traces the mean of the data accurately.

More quantitatively, Fig. 4 shows the MSE of the GP simulation on the acceleration, velocity and the space gap, as well as the LPD on the acceleration, respectively. When various levels of sensor noises are present, the GP results show that the MSE of acceleration prediction is overall very low (under 3.5×10^{-4}), and so does the corresponding velocity (under $0.01(m/s)^2$) and space gap MSE (under $4.5m^2$). It indicates that the GP model can very accurately reproduce the driving profile and is robust under noisy measurements.

Note in Fig. 4 that, as the standard deviation of the added noise increases (emulating a higher noise of real-world acceleration measurement), the MSE values for both the acceleration and the space gap prediction are lower. There are two reasons for this: (a) Inverting the covariance matrix K during the parameter inference step (11) suffers from numerical issues when the variance of $\mathbf{y}$ is too low; (b) Training may not converge to a global minimal due to the non-convex and non-smooth objective function (11), albeit the warm start.

Lastly, the LPD (bottom of Fig. 4) on the acceleration prediction indicates that the new observations (from the validating

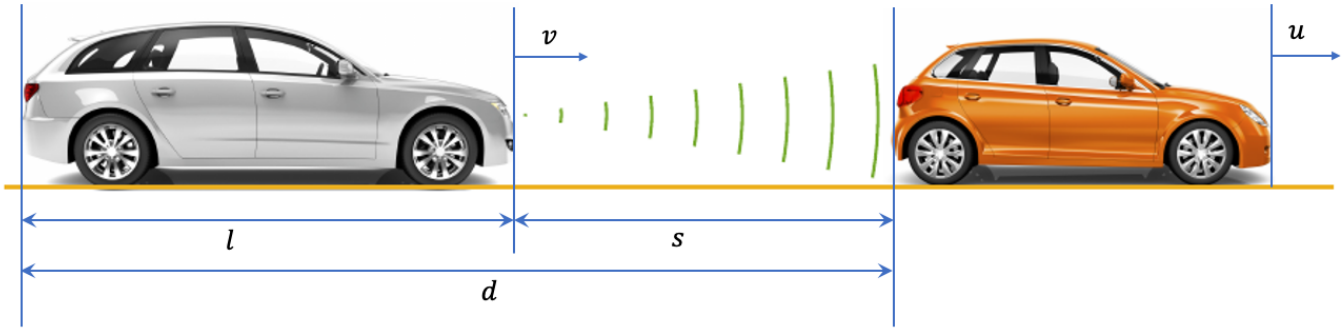


Fig. 2. A car-following system.

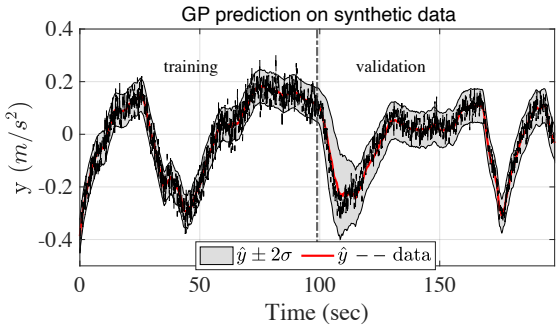


Fig. 3. Compare GP predicted acceleration (red solid line) with data (black dotted line). The first half is training result and the second half is validation result.

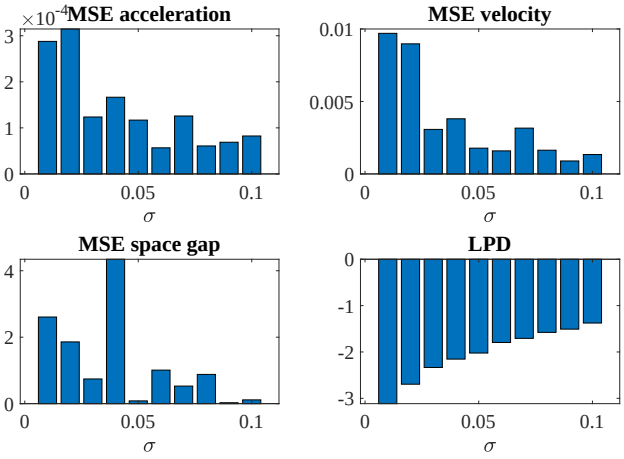


Fig. 4. Performance of GP-PACC compared with synthetic data.

set) are well-accounted by the posterior predictive distribution, even with higher sensor errors.

Overall, the numerical experiments suggest that GP can accurately reproduce the driving data even with reasonable measurement noise. The posterior distribution can also accurately characterise the uncertainty of the data set. The results show that GP-PACC almost exactly mimics the driver in a purely data-driven way, and hence improves the personalization in ADAS by adapting the longitudinal driving assistance to the driver's preferences and needs.

Fig. 5. Naturalistic driving in a car-following scenario with a gaming laptop, a Logitech racing wheel, and the Unity game engine.

B. Human-In-The-Loop Experiments on the Unity Game Engine

a) Modeling and simulation environment in Unity game engine: Game engines are conceptually the core software necessary for a game program to properly run. They generally consist of a rendering engine for graphics, a physics engine for collision detection and response, and a scene graph for the management of elements like models, sound, scripting, threading, etc. Along with the rapid development of game engines in recent years, they become popular options in the development of intelligent vehicle technology [44], with studies conducted for driver behavior modeling [45], connected vehicle systems prototyping [46], [47], and autonomous driving simulation [48], [49].

In this paper, human-in-the-loop experiments are conducted on a customized driving simulator platform, which is built with a Windows gaming laptop (processor Intel Core i7-9750 @2.60 GHz, 32.0 GB memory, NVIDIA Quadro RTX 5000 Max-Q graphics card), a Logitech G29 Driving Force racing wheel, and Unity game engine 2019.2.11f1 [50]. A three-lane highway scene is built in the simulation environment, where human drivers are able to manually drive the ego vehicle to follow the target vehicle, shown as Fig. 5.

b) Data acquisition: The experiment trip resembles a freeway high-speed scenario, and has a total period of 200 seconds. The lead vehicle's trajectory comes from the CAN-

TABLE I
MODEL TRAINING RESULTS: ALL TRAINED ON THE SAME TRAINING SET AND VALIDATED ON THE SAME VALIDATION SET SHOWN IN FIG. 6.

Model	GP	CTH-RV	OVM	IDM
MSE - acceleration	0.0909	**0.0742**	0.0914	0.0927
MSE - velocity	0.851	**0.779**	1.635	6.044
MSE - space gap	**24.8**	41.6	151.1	1101
LPD - acceleration	-0.0023	N/A	N/A	N/A

bus data of a pre-recorded trip by a human driver [51]. The trajectory contains a time-varying speed profile within the range 25-35m/s that captures a naturalistic freeway acceleration and deceleration scenario. The data is recorded in 10Hz. The training input and target are organized according to (4) and (5), where $\mathbf{Z} = \{\mathbf{z}_k = [s, v, u]_k\}_{k=1}^{N}$, and $\mathbf{y} = \{y_k\}_{k=1}^{N}$.

c) Training result: The parameter inference takes about 10 seconds to complete, with the best-estimated parameters $\theta = [l_1, l_2, l_3, \sigma_f, \sigma_n] = [14.4, 1.4, 5.9, 0.56, 0.11]$, where l_1, l_2, l_3 correspond to the characteristic length scales of s, v, u, respectively.

To visualize the training result, Fig. 6 compares the GP simulated acceleration and Unity recorded acceleration. The mean prediction (red line) aligns well with the recorded data (dotted black line) both in the training and validation sets. The uncertainty captures the variation of the recorded data in the training set, and accurately acknowledges the uncertain prediction in the validation set (with a wider prediction variance), with a few exceptions at around 120 sec.

To further validate that the GP model captures the driving dynamics, we compare with three analytical car-following models, all calibrated with the same training data, and calculate the MSE on the acceleration and space gap using the same validation data as in Fig. 6. The first model is the *constant-time headway relative-velocity* (CTH-RV) model used to characterize adaptive cruise control driving behaviors [37], the second one is the OVM [6], [52], and the last is IDM [7]. As shown in Table III, GP can perform on par, or even outperform some established analytical car-following models in terms of reproducing acceleration, velocity and space gap trajectories. Notably, GP outperforms all other models with the lowest space-gap MSE, which tends to accumulate from inaccurate acceleration prediction.

In addition, we see that the training on naturalistic driving data does not provide satisfactory results as compared to training with synthetic data. One immediate reason is that synthetic data generated using ODE-based models has a cleaner relationship between the inputs (s, v, u) and the output (acceleration), which can be captured by the squared-exponential covariance function (3); On the other hand, naturalistic driving data contains more randomness and inconsistent patterns even during the same trip. GP modeling of human-in-the-loop experiments shows promising results, even with no assumptions on the personalized driving styles.

C. Human-in-the-Loop Override Validation

In addition to the numerical analytics, the GP controller is also validated with human-in-the-loop override validation.

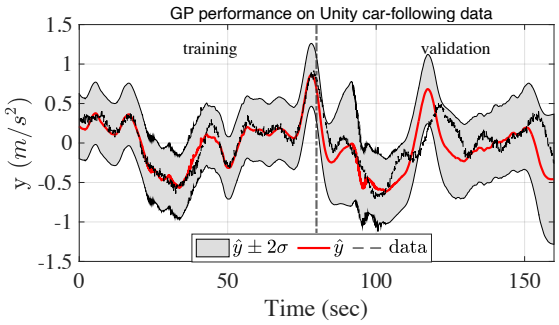


Fig. 6. Compare GP-PACC guided acceleration (red) with the actual acceleration recorded by one of the human-in-the-loop experiments (dotted black). The first half is training result and the second half is validation result.

The purpose of the tests is to measure each driver's comfort and trust of the proposed GP-PACC as well as two other baseline models (i.e., CTH-RV and IDM). The test drivers undergo several blind tests: an unknown controller drives the ego vehicle for each trip, and the frequency and duration of which the drivers override the equipped ACC (by stepping on the acceleration/braking pedals) are recorded.

a) Experiment setup: In this validation, instead of using manual control for car-following, the ego vehicle is driven with the trained GP-PACC as well as two other baseline ACC models. Four drivers (two males and two females with diverse real-world driving experience) participate in the tests, and each is randomly provided with the individualized GP-PACC or either of the two baseline models. The operating controller for a specific trip is unknown to the driver in order to eliminate potential bias. Each driver completes the tests when all three controllers are covered.

Each trip lasts 200 sec, where each driver monitors the trip and overrides the equipped ACC when he/she feels uncomfortable. The equipped ACC resumes control immediately after the driver lets go the overrides. The timestamps of which the driver overrides the ACC are be recorded.

b) Controller specifications: GP-PACC is customized for each driver. First, a 200-sec naturalistic car-following data is collected from each driver with the same simulation setup: all drivers are told to naturally follow the same leader, whose speed profile is shown as the red line in Fig. 7. All trips are recorded on the same Unity game engine with the same Logitech G29 Driving Force racing wheel. Other simulation parameters (e.g., weather, surrounding traffic and road conditions are fixed for all trips). Next, the training for GP-PACC is conducted using Algorithm 1. The resulting GP-PACC specifications are summarized in Table II.

The GP-PACC is enhanced by a predictive safety filter (formulated in (17)), with parameters emperically chosen as a_{min}=$-3m/s^2$ and $l = 4m$.

The other two baseline ACC models are taken directly from two calibrated ACC models. Specifically, ACC#1 is the constant-time headway relative-velocity (CTH-RV) model of the form [37]:

$$a_k = 0.0131(s_k - 1.6881v_k - 7.57) + 0.2692(u_k - v_k), \quad (20)$$

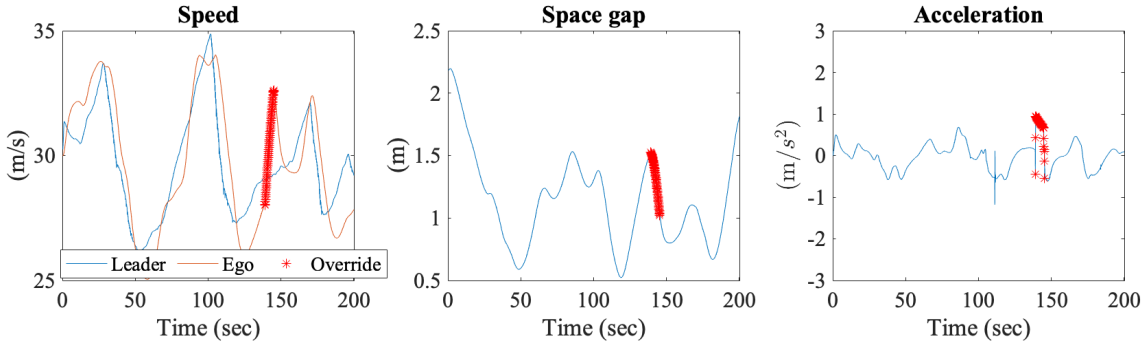


Fig. 7. A trip driven by GP controller with driver B behind the wheel.

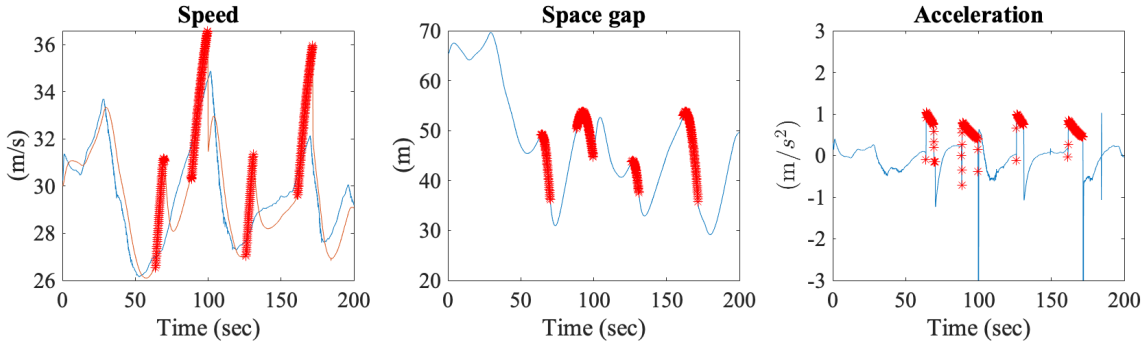


Fig. 8. A trip driven by ACC#1 with driver B behind the wheel.

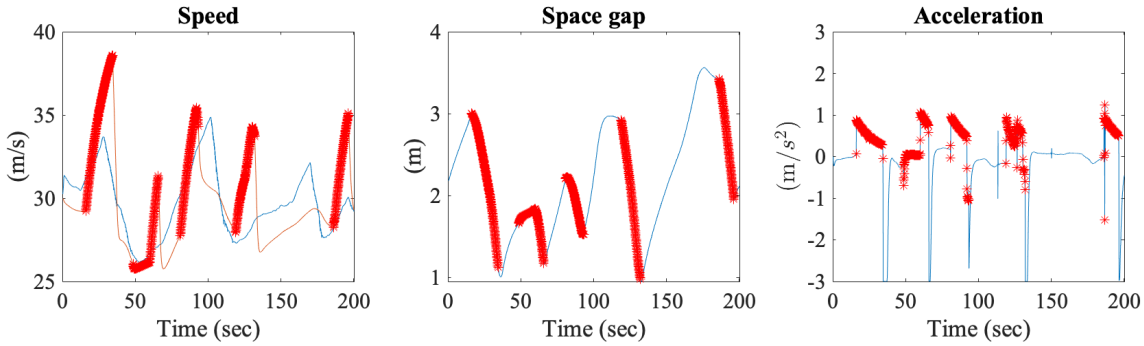


Fig. 9. A trip driven by ACC#2 with driver B behind the wheel.►YW: space gap yaxis are wrong◄

TABLE II
GP-PACC PARAMETERS FOR EACH DRIVER

Driver	l_1	l_2	l_3	σ_f	σ_n
A	14.4	1.40	5.90	0.56	0.11
B	18.4	1.72	1.11	0.43	0.20
C	5.61	0.47	1.70	0.56	0.20
D	4.33	1.25	2.40	0.17	0.19

and ACC#2 is of the form of an IDM:

$$a_k = 0.73\left[1 - \left(\frac{v_k}{30}\right)^4 - \left(\frac{s^*(v_k, u_k)}{s_k}\right)^2\right], \qquad (21)$$

where the desired space gap s^* is defined as:

$$s^*(v_k, u_k) = 2 + 1.5v_k + \frac{v_k(v_k - u_k)}{2.21}. \qquad (22)$$

c) Results: All four drivers override the operating ACC models to different extents. From Table I, in general, all drivers intervene the vehicle less when running GP-PACC as compared to running other two baseline ACC models. On average, all drivers override only 4.33% (8.7 sec total) of the 200-sec trip when GP-PACC is on board.

As an illustration, the recorded trips from one of the drivers (driver B) can be visualized in Fig. 7-9. The top row (Fig. 7) shows the trajectories when GP-PACC is the selected controller. The middle row (Fig. 8) corresponds to ACC#1 (CTH-RV controller) being in operation and the bottom row (Fig. 9) corresponds to ACC#2 (IDM controller). The recorded trajectories include the leader's and the ego vehicle's speeds (left-most column), space gap (middle column) and acceleration (right column) with respect to time. The red highlights indicate the timestamps when the driver overrides ACC that is operating during that trip.

Driver B indicates that he overrides when he feels "falling behind from the lead vehicle, and the neighboring vehicles on

TABLE III
HUMAN-IN-THE-LOOP EXPERIMENTS RESULTS: DRIVERS TAKEOVER PERCENTAGE DURING A 200-SEC TRIP

Driver	GP-PACC	ACC#1(CTH-RV)	ACC#2(IDM)
A (F)	**0.6%**	4.9%	12.5%
B (M)	**5.6%**	16.6%	29.7%
C (F)	**11.3%**	23.5%	67.1%
D (M)	**2.1%**	4.7%	23.6%
Avg.	**4.9%**	12.4%	33.2%

the right lane will cut into the gap between the ego vehicle and the lead vehicle". Fig. 7 shows that the driver feels comfortable when GP-PACC is in control, i.e., the driver did not override the controller during the entire trip. On the other hand, the driver pressed the gas pedal several times when ACC#1 is in operation (Fig. 8), and even more so with ACC#2 engaged (Fig. 9). The results strongly indicate that the driver favors the personalized controller (GP-PACC) in the unbiased test settings.

V. CONCLUSION AND FUTURE WORK

In this paper we propose GP-PACC that mimics personalized car-following behavior. The learning is achieved using a Gaussian Process regression with nonlinear output-error training on the car-following data. We explore this purely data-driven controller design in conjunction with a predictive safety filter to capture personalized driving styles, which sometimes cannot be captured by an explicit car-following model.

The training result shows that GP has the potential to provide safe and realistic acceleration guidance that closely resembles personalized acceleration profile. Specifically, GP almost exactly recovers the car-following profiles of an IDM driver (data generated using an IDM), and outperforms three other established analytical car-following models in terms of reproducing naturalistic car-following space gap trajectories. A series of human-in-the-loop experiments are conducted on the Unity driving simulator to test drivers' override rates when running their personalized GP-PACC versus other baseline ACC models. Results indicate that all tested drivers express comfort using GP-PACC, which reduces the human override duration 60% and 85% as compared to two other standard ACC models, respectively. This brings promising potentials of the acceptance towards the personalized controller in near real-world scenarios.

For future work, adaptive GP training can be incorporated into current routine to enhance the proposed GP-PACC. Since training a GP dynamical system requires only limited data, it is possible to adaptively train the GP model as more data is collected. This training procedure allows to capture the variations in driving behaviors across a longer period of time. Additionally, since driver override has only been adopted as a measurement to test our GP-PACC in this paper, it can also be considered as a direct feedback to the GP model, which will enhance the performance of our future GP-PACC in a more straightforward manner.

ACKNOWLEDGMENT

The authors want to sincerely thank Zhouqiao Zhao and Ziwei Zhang for their participation in the human-in-the-loop experiments. The comments and suggestions came from Rohit Gupta, Akila Ganlath, Sergei Avedisov, Xuewei Qi, and Takamasa Higuchi are also greatly appreciated.

This study is based upon work supported by the National Science Foundation under Grant No. CNS-1837652, and it is sponsored by the "Digital Twin" project of InfoTech Labs, Toyota Motor North America. The contents of this study only reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views of Toyota Motor North America.

APPENDIX

Derivation of safe set

We solve for a set at time step k^* consists of s_{k^*} and Δv_{k^*} s.t. the spacing d between the two vehicles is always greater than the length of the ego vehicle l.

Consider the leader vehicle's rear bumper position during a time horizon T given its initial position $p_1(0)$ and constant speed u:

$$p_1(T) = p_1(0) + uT, \tag{23}$$

and the ego vehicle's rear bumper position during constant maximum deceleration $a_{\min}$ given initial position $p_2(0)$ and initial velocity v_0:

$$p_2(T) = p_2(0) + v_0 T + \frac{1}{2} a_{\min} T^2. \tag{24}$$

Consequently, the spacing between the leader and the ego vehicle at T is:

$$\begin{aligned} d(T) = p_1(T) - p_2(T) &= (p_1(0) - p_2(0)) + (u - v_0)T - \frac{1}{2} a_{\min} T^2 \\ &= (s_0 + l) + \Delta v_0 T - \frac{1}{2} a_{\min} T^2, \end{aligned} \tag{25}$$

where $s_0 = d(0) - l$ is the initial space gap and $\Delta v_0 = u - v_0$ is the initial relative velocity between the two vehicles. Since it is always safe if the ego vehicle starts no faster than the leader, i.e., $\Delta v_0 \geq 0$, we only consider $\Delta v_0 < 0$ when solving for $s_0, \Delta v_0$ such that $d(T) > l, \forall T > 0$. Note that the function $d(T)$ is an upward-convex quadratic function with respect to T, and safety will be guaranteed if the minimum of $d(T)$ is greater than l. The minimum of $d(T)$ is denoted as $d(T^*)$, where T^* can be derived when $\frac{d}{dT} d(T) = 0$:

$$\begin{aligned} \frac{d}{dT} d(T^*) = \Delta v_0 - a_{\min} = 0 \Rightarrow T^* &= \frac{\Delta v_0}{a_{\min}} \\ d(T)|_{T=T^*} &= s_0 + l + \frac{\Delta v_0^2}{2 a_{\min}}. \end{aligned} \tag{26}$$

Therefore, the safe set at any time k is

$$\mathcal{S}_k = \{(s_k, \Delta v_k) \in \mathbb{R}^2 | (s_k + \frac{\Delta v_k^2}{2 a_{\min}} > l) \cup (\Delta v_k > 0)\}. \tag{27}$$

REFERENCES

[1] M. Hasenjäger, M. Heckmann, and H. Wersing, "A survey of personalization for advanced driver assistance systems," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 335–344, 2019.

[2] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the path program," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 114–130, 1991.

[3] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[4] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[5] Z. Wang, Y. Bian, S. E. Shladover, G. Wu, S. E. Li, and M. J. Barth, "A survey on cooperative longitudinal motion control of multiple connected and automated vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 1, pp. 4–24, 2019.

[6] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," *Phys. Rev. E*, vol. 51, pp. 1035–1042, Feb 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.51.1035

[7] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, pp. 1805–1824, Aug 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.62.1805

[8] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0191261581900370

[9] R. E. Chandler, R. Herman, and E. W. Montroll, "Traffic dynamics: Studies in car following," *Operations Research*, vol. 6, no. 2, pp. 165–184, 1958. [Online]. Available: https://doi.org/10.1287/opre.6.2.165

[10] R. Wilson and J. Ward, "Car-following models: fifty years of linear stability analysis – a mathematical perspective," *Transportation Planning and Technology*, vol. 34, no. 1, pp. 3–18, 2011. [Online]. Available: https://doi.org/10.1080/03081060.2011.530826

[11] Y. Wang, M. L. D. Monache, and D. B. Work, "Identifiability of car-following dynamic," 2021.

[12] M. Wiering and M. Van Otterlo, *Reinforcement Learning: State of the Art*. Springer, 2012.

[13] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.

[14] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2641–2646.

[15] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[16] G. B. Giannakis and E. Serpedin, "A bibliography on nonlinear system identification," *Signal Processing*, vol. 81, no. 3, pp. 533–580, 2001.

[17] L. Ljung, "System identification," *Wiley encyclopedia of electrical and electronics engineering*, pp. 1–19, 1999.

[18] R. Tedrake, "Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832," *Working draft edition*, vol. 3, 2009.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[20] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 283–298, 2007.

[21] R. Babuška, "Neuro-fuzzy methods for modeling and identification," in *Recent advances in intelligent paradigms and applications*. Springer, 2003, pp. 161–186.

[22] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[23] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[24] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5777–5783, 2018.

[25] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7130–7135.

[26] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1424–1431.

[27] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with gaussian processes," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005. [Online]. Available: https://doi.org/10.1080/13873950500068567

[28] H. Fan and M. S. Poole, "What is personalization? perspectives on the design and implementation of personalization in information systems," *Journal of Organizational Computing and Electronic Commerce*, vol. 16, no. 3-4, pp. 179–202, 2006.

[29] Y. Wang, Z. Wang, K. Han, P. Tiwari, and D. B. Work, "Personalized adaptive cruise control via gaussian process regression," 2021, accepted.

[30] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. [Online]. Available: https://doi.org/10.1007/978-3-540-28650-9_4

[31] M. P. Deisenroth, "Efficient reinforcement learning using gaussian processes," Ph.D. dissertation, Karlsruhe Series on Intelligent Sensor-Actuator-Systems / Karlsruher Institut für Technologie, Intelligent Sensor-Actuator-Systems Laboratory, 2010.

[32] J. Kocijan and D. Petelin, "Output-error model training for gaussian process models," in *Adaptive and Natural Computing Algorithms*, A. Dobnikar, U. Lotrič, and B. Šter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 312–321.

[33] T. Ma and B. Abdulhai, "Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters," *Transportation Research Record*, vol. 1800, no. 1, pp. 6–15, 2002. [Online]. Available: https://doi.org/10.3141/1800-02

[34] H. Wang, W. Wang, J. Chen, and M. Jing, "Using trajectory data to analyze intradriver heterogeneity in car-following," *Transportation Research Record*, vol. 2188, no. 1, pp. 85–95, 2010. [Online]. Available: https://doi.org/10.3141/2188-10

[35] B. Ciuffo and V. Punzo, ""no free lunch" theorems applied to the calibration of traffic simulation models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 553–562, 2014.

[36] V. Papathanasopoulou and C. Antoniou, "Towards data-driven car-following models," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 496 – 509, 2015, engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X15000716

[37] G. Gunter, R. Stern, and D. B. Work, "Modeling adaptive cruise control vehicles from experimental data: model comparison," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3049–3054.

[38] M. Stepančič and J. Kocijan, "Gaussian process model-based system identification toolbox for matlab," 2017. [Online]. Available: https://github.com/Dynamic-Systems-and-GP/GPdyn

[39] F. de Souza and R. Stern, "Calibrating microscopic car following models for adaptive cruise control vehicles: a multi-objective approach," 2020.

[40] A. Girard, "Approximate methods for propagation of uncertainty with gaussian process models," Ph.D. dissertation, Citeseer, 2004.

[41] N. Aréchiga and B. Krogh, "Using verified control envelopes for safe controller design," in *2014 American Control Conference*, 2014, pp. 2918–2923.

[42] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1294–1307, 2016.

[43] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," 2021.

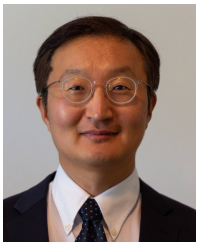
[44] J. Ma, C. Schwarz, Z. Wang, M. Elli, G. Ros, and Y. Feng, "New simulation tools for training and testing automated vehicles," in *Road Vehicle Automation 7*, G. Meyer and S. Beiker, Eds. Cham: Springer International Publishing, 2020, pp. 111–119.

[45] Z. Wang, X. Liao, C. Wang, D. Oswald, G. Wu, K. Boriboonsomsin, M. Barth, K. Han, B. Kim, and P. Tiwari, "Driver behavior modeling using game engine and real vehicle: A learning-based approach," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 738–749, 2020.

[46] Z. Wang, G. Wu, K. Boriboonsomsin, M. Barth *et al.*, "Cooperative ramp merging system: Agent-based modeling and simulation using game engine," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 2, 2019.
[47] Y. Liu, Z. Wang, K. Han, Z. Shou, P. Tiwari, and J. H. L. Hansen, "Sensor fusion of camera and cloud digital twin information for intelligent vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2020.
[48] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.
[49] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.
[50] Z. Wang, K. Han, and P. Tiwari, "Digital twin simulation of connected and automated vehicles with the unity game engine," *TechRxiv*, 2021.
[51] Y. Wang, G. Gunter, M. Nice, M. L. D. Monache, and D. B. Work, "Online parameter estimation methods for adaptive cruise control systems," *IEEE Transactions on Intelligent Vehicles*, 2020.
[52] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Structure stability of congestion in traffic dynamics," *Japan Journal of Industrial and Applied Mathematics*, vol. 11, pp. 203–223, 1994.

Yanbing Wang received the B.S. degree from the University of Illinois at Urbana-Champaign in 2018. She is currently a Ph.D. student in Civil and Environmental Engineering and the Institute for Software Integrated Systems at Vanderbilt University, and was a Research Intern at the InfoTech Labs of Toyota Motor North America. Ms. Wang is an recipient of the Eisenhower Graduate Fellowship (2018 and 2019). Her research interests include system identification and control for autonomous vehicles.

Ziran Wang (S'16-M'19) received the Ph.D. degree from The University of California, Riverside in 2019, and the B.E. degree from Beijing University of Posts and Telecommunications in 2015, respectively. He is currently a Research Scientist at Toyota Motor North America, InfoTech Labs in Silicon Valley, where he conducts research in the "Digital Twin" project. Dr. Wang serves as associate editor of SAE International Journal of Connected and Automated Vehicles, founding chair of IEEE Technical Committee on Internet of Things in Intelligent Transportation Systems (IoT in ITS), and member of four other technical committees across IEEE and SAE. His research focuses on intelligent vehicle technology, including cooperative automated driving, driver behavior modeling, and vehicular cyber-physical systems.



Kyungtae (KT) Han (M'97-SM'15) received the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin in 2006. He is currently a Principal Researcher at Toyota Motor North America, InfoTech Labs. Prior to joining Toyota, Dr. Han was a Research Scientist at Intel Labs, and a Director in Locix Inc. His research interests include cyber-physical systems, connected and automated vehicle technique, and intelligent transportation systems.

Prashant Tiwari received the Ph.D. degree in mechanical engineering from Rensselaer Polytechnic Institute in 2004, and the MBA degree from University of Chicago in 2016. He is currently a Executive Director at Toyota Motor North America, InfoTech Labs. Dr. Tiwari is highly active in Automotive Edge Computing Consortium (AECC) and SAE. Prior to joining Toyota, Dr. Tiwari held several leadership positions of increasing responsibilities at GE and UTC Aerospace Systems.

Daniel B. Work received the B.S. degree from the Ohio State University, Ohio, in 2006 and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 2007, and 2010, respectively, all in civil engineering. He is an Associate Professor with Civil and Environmental Engineering and Institute for Software Integrated Systems, Vanderbilt University. His research interests include transportation cyber physical systems. He was the recipient of the CAREER award from the National Science Foundation (2014) the Gilbreth Lectureship from the National Academy of Engineering (2018).