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.

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]. Analysis on the impact of traffic dynamics also follows [12]–[16]. Other modeling efforts such as [17], [18] take into account of vehicle dynamics as well. 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) [19], [20] where a control policy is learned by maximizing a reward function that describes the system evolution. Extensions of RL include inverse reinforcement learning (IRL), which learns a reward

Manuscript received 31 August 2021; revised 11 March 2022; accepted 4 May 2022. Date of publication 13 May 2022; date of current version 7 November 2022. This work was supported in part by the National Science Foundation under Grant CNS-1837652; and in part by InfoTech Labs, Toyota Motor North America, under Digital Twin Project. The Associate Editor for this article was N. Bekiaris-Liberis. *(Corresponding author: Yanbing Wang.)*

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

Ziran Wang, Kyungtae Han, and Prashant Tiwari are with InfoTech Labs, Toyota Motor North America, Mountain View, CA 94043 USA (e-mail: ryanwang11@hotmail.com; kyungtae.han@toyota.com; prashant.tiwari@toyota.com).

Digital Object Identifier 10.1109/TITS.2022.3174042

function through expert demonstration [21], [22]. 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. [23]–[25]. A notion of underactuation in control has recently been studied to design controllers that take advantage of the natural dynamics of the system [26]. Other data-driven system identification tools such as SINDy [27], Gaussian Process (GP) [28] and Neuro-fuzzy methods [29] 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 [30], [31], as well as formal verification tools to facilitate assured autonomy from learning [32]–[34].

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 [35]. These advantages allow GP to model both (a) intra-driver stochasticity (variation of driving behaviors within the same driver, through variance modeling), as well as (b) inter-driver stochasticity (through individual training). 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) [36]. 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 [37] 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 designing a personalized, high-speed controller for the combined driver-vehicle system, and the model output is the predicted acceleration of the vehicle, implicitly considering the delay of driver reaction and the lag of vehicle actuation. The driver reaction and the low-level vehicle dynamics can be separately modeled if commanded acceleration is available. In this study we consider a simplistic approach to combine the two systems. Therefore, 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. 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 [35], [38], [39]. 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.

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 $f_{\mathrm{GP}}(\mathbf{z}) : \mathbb{R}^{\dim(\mathbf{z})} \to \mathbb{R}$, such as (1). The mapping between the input vector $\mathbf{z}$ and the function value $f_{\mathrm{GP}}(\mathbf{z})$ is accomplished by the assumption that $f_{\mathrm{GP}}(\mathbf{z})$ is a random variable and is jointly Gaussian distributed with $\mathbf{z}$, which is also assumed to be a random variable [38].

1) 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 $\mathbf{z}_j$:

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

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise, and $P = \mathrm{diag}[l_1^2, \ldots, l_{\dim(\mathbf{z})}^2]$ contains the characteristic length scale for each dimension of the input vector. The hyperparameters of the covariance function $\theta = [\sigma_f, \sigma_n, l_1, \ldots, l_{\dim(\mathbf{z})}]^T$ 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.

2) 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:

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]^T \tag{4}$$

$$\mathbf{y} = [y_1, y_2, \ldots, y_N]^T, \tag{5}$$

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:

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

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

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

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

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

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

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.,

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

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

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

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 $\mathbf{z}_*$:

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

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 regression vectors in GP can be augmented to include lagged terms, for an explicit model of delays. For example, $\hat{y}_k = f_{GP}(\hat{\mathbf{z}}_k, \hat{\mathbf{z}}_{k-1}, \ldots, \hat{\mathbf{z}}_{k-m}, \theta)$, where m is the lagged timesteps. However, this formulation comes with a complexity trade-off, as the run time for GP regression is $\mathcal{O}(N^3)$ due to the matrix inversion. The naive GP regression does not lead to closed-loop stability of the car-following system [40]. 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.

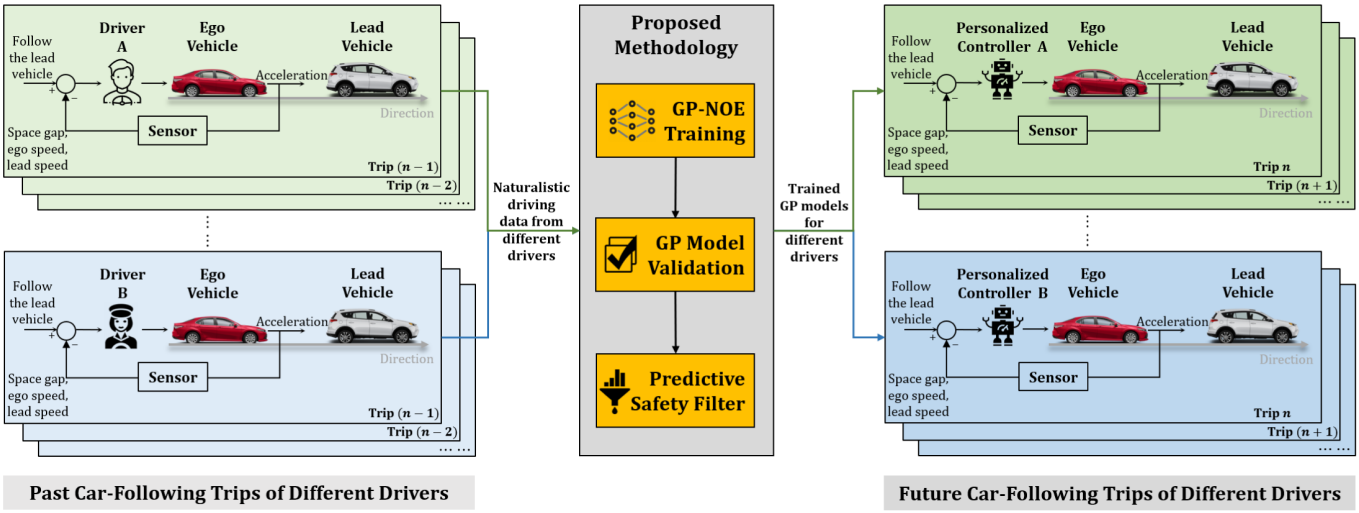


Fig. 1. Block diagram of the proposed GP-PACC system.

B. GP-NOE Training

We adopt a training process similar to calibrating an ODE-based car-following model [41]–[45]. 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:

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

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 $\mathbf{z}_k = [s_k, v_k, u_k]$.

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 $\hat{\mathbf{Z}}_{1:N} = [\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_N]^T$. 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 [46].

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 to train a reasonable GP-NOE model with 60 sec of 10Hz data, it takes 15 seconds on a 2.7GHz Quad-Core laptop. Note that the GP output is a Gaussian distribution on the acceleration prediction. The prediction mean is used as the actual acceleration, and the uncertainty is purely for understanding the prediction confidence. Although the loss function for GP-NOE is less smooth than that of the naive GP regression and non-convex (see a visual illustration in [40]), which does not guarantee the global optimal θ to be found, a good initial guess on θ can achieve 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., [41]–[45], [47]).

Two performance metrics are measured: the mean squared error (MSE) and the log predictive-density error (LPD) [35], [48] between the GP simulated acceleration and the recorded

acceleration of a validation data set:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$$
$$LPD = \frac{1}{2}\ln(2\pi) + \frac{1}{2N}\sum_{k=1}^{N}\left(\ln(\sigma_k^2) + \frac{(y_k - \hat{y}_k)^2}{\sigma_k^2}\right) \quad (15)$$

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 driving safety into account. Throughout the literature, we found learning-based control achieves "safe-by-design" with verified control envelopes [49], fixed-point computations of the set-valued mappings [50], and safety filtering [51], as common approaches. In this paper we adopt a predictive safety filtering approach similar to [51], which finds a safe acceleration profile that is closest to the GP-predicted acceleration and achieves collision avoidance.

Consider the following notations and assumptions:

Let $\mathcal{I}_{\geq k}$ denotes a set of integers in the interval $[k, \infty) \in \mathbb{R}$. Let a_k denote the acceleration of the leader vehicle at time k, which we assume can be measured. y_k stands for the acceleration for the follower at k. $a_{\min}$ denotes the hardest braking deceleration for the follower, which the follower vehicle can actuate instantaneously. $s_{\min}$ is the minimum space gap.

We develop a safety filter on commanded accelerations to achieve collision avoidance. The safety filter seeks to ensure two properties are met at all times. The first property is collision avoidance in the form $s_k \geq s_{\min}$. The second property is bounded deceleration in the form $y_k \geq a_{\min}$. To choose accelerations that achieve these two properties we form at each time-step k a set of allowable states into which the vehicle can move to time-step $k+1$, which we denote as $\mathcal{S}$:

$$(s_k, v_k, u_k, a_k) \in \mathcal{S} \Rightarrow y_{k'} \geq a_{\min}, \quad s_{k'} \geq s_{\min}, \quad \forall k' \in \mathcal{I}_{\geq k} \quad (16)$$

according to the following discrete-time dynamics:

$$\begin{bmatrix} s \\ v \\ u \\ a \end{bmatrix}_{k+1} = g(s_k, v_k, u_k, a_k, y_k) = \begin{bmatrix} s \\ v \\ u \\ a \end{bmatrix}_k + \begin{bmatrix} u - v \\ y \\ a \\ 0 \end{bmatrix}_k \Delta t. \quad (17)$$

$\mathcal{S}$ is derived using a standard stopping time condition under constant acceleration from the leading vehicle (see appendix). By choosing y_k such that $(s_k, v_k, u_k, a_k) \in \mathcal{S}$ the filter ensures that either both collision avoidance and bounded acceleration will be met in all following time-steps, or $\mathcal{S} = \emptyset$ meaning that a collisions cannot be avoided. In the case that multiple such commanded accelerations exist, we choose the y_k that is closest to that prescribed by the GP $\hat{y}_k$. This can be stated in the following form:

$$\begin{aligned} &\underset{y_k}{\text{minimize}} \ (\hat{y}_k - y_k)^2 \\ &\text{s.t. } \hat{y}_k = f_{GP}(s_k, v_k, u_k, \theta) \\ &\quad [s_{k+1}, v_{k+1}, u_{k+1}, a_{k+1}]^T = g(s_k, v_k, u_k, a_k, y_k) \\ &\quad (s_{k+1}, v_{k+1}, u_{k+1}, a_{k+1}) \in \mathcal{S}. \end{aligned} \quad (18)$$

Scenarios in which the set $\mathcal{S} = \emptyset$ can be trivially triggered through simulating large lead vehicle decelerations that exceed $a_{\min}$ in magnitude. This is consistent with game theoretic results that describe collisions between systems with equal, and unequal, dynamics and input ranges. Additional study of the feasibility of these safety regions through data and analysis of naturalistic and controlled scenarios is reserved for future work. One possible alternative formulation is through, for example, a *control barrier function* [30].

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 additive noises 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 simulated car-following data serves as "ground truth", from which the GP-PACC prediction errors can be computed, and the prediction variance can be compared with the known noise. The model is expressed as:

$$\begin{aligned} \dot{v}(t) &= f_{\text{CF}}(s(t), v(t), u(t)) \\ &= a\left[1 - \left(\frac{v(t)}{v_f}\right)^{\delta} - \left(\frac{s^*(v(t), u(t))}{s(t)}\right)^2\right] \end{aligned} \quad (19)$$

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

$$s^*(v(t), u(t)) = s_j + v(t)T + \frac{v(t)(v(t) - u(t))}{2\sqrt{ab}}. \quad (20)$$

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].

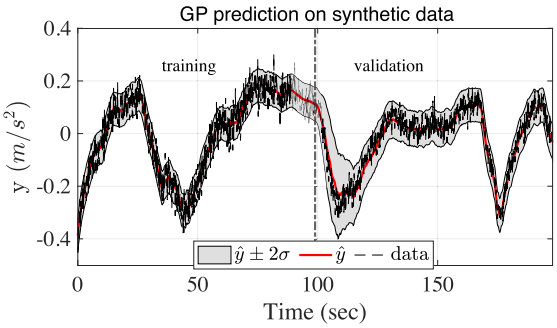


Fig. 2. Compare GP predicted acceleration (red solid line) with data (black dotted line). The first half is training result and the second half is validation result.

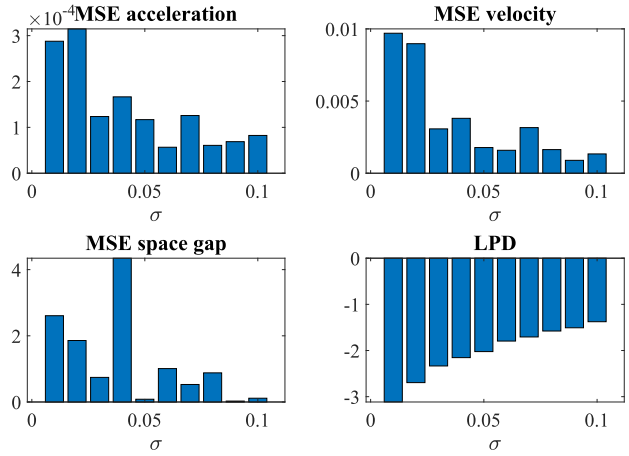


Fig. 3. Performance of GP-PACC compared with synthetic data.

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. 2). This composition is shown to reproduce the car-following styles for various drivers consistently well in our later experiments.

Fig. 2 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. 3 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. 3 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. 3) on the acceleration prediction indicates that the new observations (from the validating 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.

B. Human-in-the-Loop Experiments on the Unity Game Engine

1) 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 [52], with studies conducted for driver behavior modeling [53], connected vehicle systems prototyping [54], [55], and autonomous driving simulation [56], [57].

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 [58]. 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. 4.

2) 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-bus data of a pre-recorded trip by a human driver [59]. 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 $\mathbf{y} = \{y_k\}_{k=1}^N$.

3) 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

Fig. 4. Naturalistic driving in a car-following scenario with a gaming laptop, a Logitech racing wheel, and the Unity game engine.

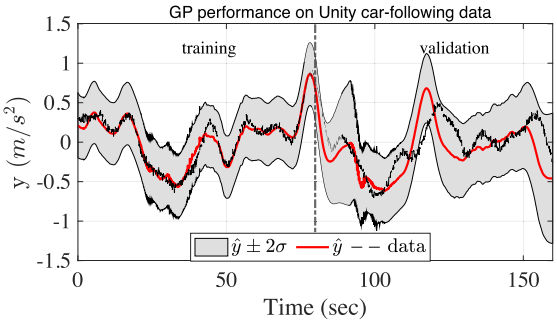


Fig. 5. Compare GP-PACC guided acceleration (red) with the actual acceleration recorded by one of the human-in-the-loop experiments (dotted black). The first half is training result and the second half is validation result.

l_1, l_2, l_3 correspond to the characteristic length scales of s, v, u, respectively.

To visualize the training result, Fig. 5 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 its ability to reconstruct human-driving profiles with that of two ODE-based car-following models. The first model is the *constant-time headway relative-velocity* (CTH-RV) model used to characterize adaptive cruise control driving behaviors [45], and the second one is IDM [7], which is used to describe human-driving behaviors. Since the GP-PACC design problem is formulated as a system identification problem, where the goal is to minimize the discrepancy between the predicted driving profiles and the measured ones, it shares the same objective with calibrating car-following models. GP-PACC, CTH-RV and IDM are trained (calibrated) with the same training data, and validated with the same testing data produced from the same driver shown in Fig. 5. We use acceleration as the prediction target in order to be consistent with the GP model training, which also has the target of minimizing the error on predicted acceleration. A general form of calibrating any car-following model is written as minimizing a sum-of-squared cost function:

$$\begin{aligned} &\underset{\theta}{\text{minimize}}: \ \sum_{k=1}^{N} (y_k - \hat{y}_k)^2 \\ &\text{subject to: } \hat{y}_k = f_{\text{CF}}(s_k, v_k, u_k, \theta), \quad k = 1, 2, .., N \end{aligned} \tag{21}$$

with possible additional constraints on the initial conditions, and bounds on the parameters.

The training and testing errors are measured by MSE on the acceleration and space gap. As shown in Table I, GP can perform on par with, or even outperform some established analytical car-following models in terms of reproducing acceleration, velocity and space gap trajectories. Notably, GP outperforms both other models with the lowest space-gap MSE, which tends to accumulate from inaccurate acceleration prediction.

TABLE I

MODEL TRAINING RESULTS: ALL TRAINED ON THE SAME TRAINING SET AND VALIDATED ON THE SAME VALIDATION SET SHOWN IN FIG. 5

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

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. More driving datasets are desired to test the ability of our model on recovering the longitudinal car-following behavior. It will be interesting to compare not only with other car-following models, but also across datasets (e.g., naturalistic driving datasets [60] and field experimental data for ACC vehicles [12], [16]) for future work. Nevertheless, GP modeling of human-in-the-loop experiments shows promising results, even with no explicit 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. 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.

1) 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

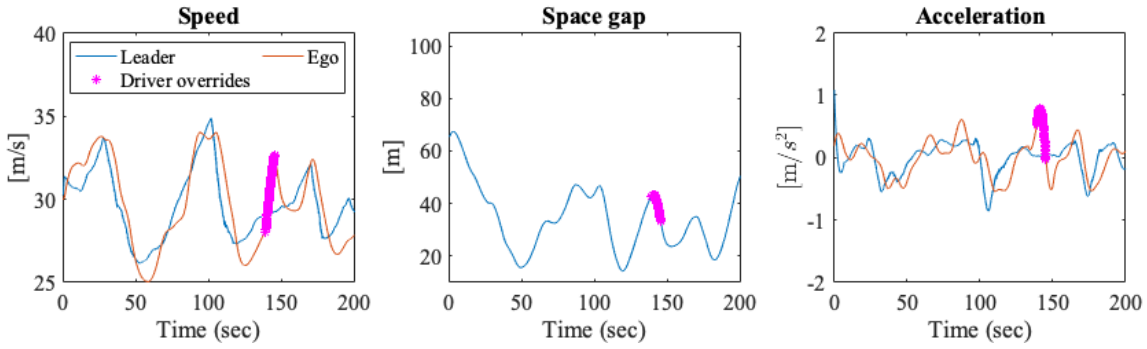


Fig. 6. A trip driven by GP controller with driver B behind the wheel.

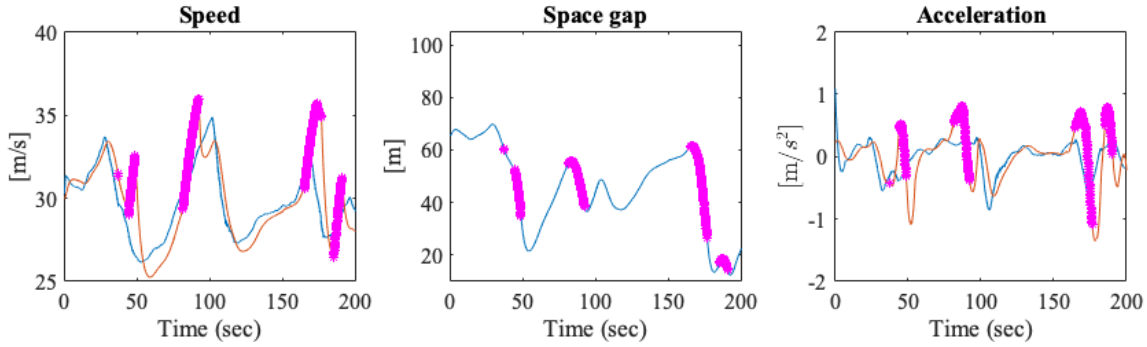


Fig. 7. A trip driven by ACC#1 with driver B behind the wheel.

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.

2) 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 and acceleration profiles are shown as the blue lines in Fig. 6. 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 (18)), 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

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

the form:

$$a_k = 0.0131(s_k - 1.6881\ v_k - 7.57) + 0.2692(u_k - v_k), \tag{22}$$

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

$$a_k = 0.73\left[1 - \left(\frac{v_k}{30}\right)^4 - \left(\frac{s^*(v_k, u_k)}{s_k}\right)^2\right], \tag{23}$$

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

$$s^*(v_k, u_k) = 2 + 1.5v_k + \frac{v_k(v_k - u_k)}{2.21}. \tag{24}$$

The parameters for both baseline ACCs are chosen as suggested in [7], [45].

3) Results: All four drivers override the operating ACC models to different extents. From Table III, 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 4.43% (8.7 sec total) of the 200-sec trip when GP-PACC is on board.

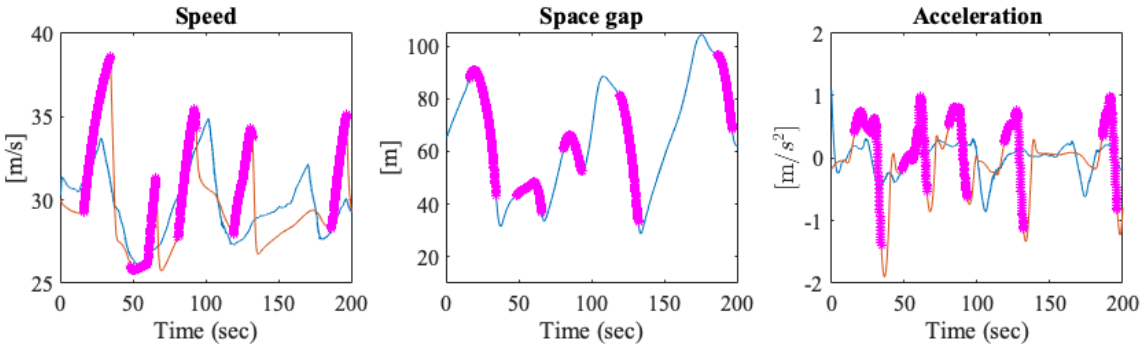


Fig. 8. A trip driven by ACC#2 with driver B behind the wheel.

TABLE III
HUMAN-IN-THE-LOOP EXPERIMENTS RESULTS: DRIVERS GAS AND BRAKE TAKEOVER PERCENTAGE DURING A 200-sec TRIP

Driver	GP-PACC	ACC#1(CTH-RV)	ACC#2(IDM)
	Gas / Brake	Gas / Brake	Gas / Brake
A (F)	**0% / 0.60%**	4.2% / 0.65%	11.9% / 0.6%
B (M)	**3.1% / 0.15%**	14.6% / 1.8%	26.7% / 1.7%
C (F)	**8.6%/ 2.7%**	18.8% / 4.6%	58.8% / 8.25%
D (M)	**2.3% / 0.25%**	4.4% / 0.35%	22.4% / 0.9%
Avg.	**3.5% / 0.93%**	10.5% / 1.85%	30.0% / 2.86%

As an illustration, the recorded trips from one of the drivers (driver B) can be visualized in Fig. 6-8. The top row (Fig. 6) shows the trajectories when GP-PACC is the selected controller. The middle row (Fig. 7) corresponds to ACC#1 (CTH-RV controller) being in operation and the bottom row (Fig. 8) corresponds to ACC#2 (IDM controller). The recorded trajectories include the speeds for leader and follower (left-most column), space gap (middle column) and accelerations for leader adn follower (right column) with respect to time. The magenta highlights indicate the timestamps when the driver overrides ACC (either by pressing gas or brake pedal) that is in operation.

Driver B indicates that he overrides when he feels "falling behind from the lead vehicle, and the neighboring vehicles on the right lane will cut into the gap between the ego vehicle and the lead vehicle". Fig. 6 shows that the driver feels comfortable when GP-PACC is in control, i.e., the driver only overtook the controller for about 1 second during the entire trip. On the other hand, the driver pressed the gas pedal several times when ACC#1 is in operation (Fig. 7), and even more so with ACC#2 engaged (Fig. 8). 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 two 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 and a wider range of speed. 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. As more contextual information (e.g., weather, road geometry) becomes available, the training features can be augmented to expand the functionality of personalization.

Furthermore, safety critical scenarios were not triggered for simulation ranges utilized within this paper, even with the conservative assumption (e.g., constant acceleration dynamics). More work that relies on reachability analysis with controller-based methods is reserved for future work.

APPENDIX

Derivation of Safe Set: For a car-following system shown in Figure 9, we derive the safe set $\mathcal{S}$ given the state (s_0, v_0, u_0, a_0) and $a_{\min}$, which stand for the initial space gap,

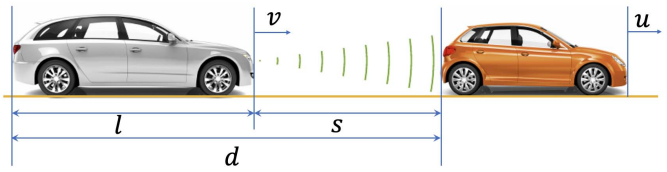


Fig. 9. A car-following system.

follower velocity, leader velocity, leader acceleration and the hardest braking deceleration for the follower. The safe condition gives a requirement for (s_0, v_0, u_0, a_0) such that future collision can be prevented if the follower vehicle executes $a_{\min}$, $a_{\min} < 0$. Let the safety space-gap margin be $s_{\min}$, $s_{\min} > 0$, and assume the given state is safe, i.e., $s_0 \geq s_{\min}$, and $v_0 \geq 0, u_0 \geq 0$.

Consider two scenarios:

1) $a_0 < 0$ and
2) $a_0 \geq 0$.

Scenario 1): Denote the leader and follower position as $p_l(t)$ and $p_f(t)$, respectively. Consider a non-decreasing position for the leader vehicle when decelerating:

$$p_l(t) = \begin{cases} p_l(0) + u_0 t + \frac{1}{2} a_0 t^2, & 0 < t < -\frac{u_0}{a_0} \\ p_l(0) - \frac{u_0^2}{2a_0}, & t \geq -\frac{u_0}{a_0}, \end{cases}$$

Similarly, the non-decreasing follower's position during hardest braking can be denoted as

$$p_f(t) = \begin{cases} p_f(0) + v_0 t + \frac{1}{2} a_{\min} t^2, & 0 < t < -\frac{v_0}{a_{\min}} \\ p_f(0) - \frac{v_0^2}{2a_{\min}}, & t \geq -\frac{v_0}{a_{\min}}. \end{cases}$$

Denote the stopping time for leader and follower as

$$T_l^s = -\frac{u_0}{a_0}, \quad T_f^s = -\frac{v_0}{a_{\min}}.$$

Safety requires that the space gap between the two vehicles is above the safety margin when both vehicles are at a stop, i.e.,

$$p_l(T_l^s) - p_f(T_f^s) - l > s_{\min}, \text{ or}$$

$$p_l(0) - \frac{u_0^2}{2a_0} - \left(p_f(0) - \frac{v_0^2}{2a_{\min}} \right) - l > s_{\min}$$

$$s_0 - \frac{u_0^2}{2a_0} + \frac{v_0^2}{2a_{\min}} > s_{\min}.$$

Consequently, the condition for safe state when $a_0 < 0$ is

$$C_1 := s_0 - \frac{u_0^2}{2a_0} + \frac{v_0^2}{2a_{\min}} > s_{\min}.$$

Scenario 2): The position for the leader becomes

$$p_l(t) = p_l(0) + u_0 t + \frac{1}{2} a_0 t^2.$$

The same safety criterion can be derived by setting

$$s(t) = p_l(t) - p_f(t) - l = \frac{1}{2}(a_0 - a_{\min})t^2 + (u_0 - v_0)t + s_0 > s_{\min} \quad \forall t > 0.$$

Note that $s(t)$ is a convex quadratic function. It can be observed that $s(t) > 0$ $\forall t$ if $s(t)$ has no real roots, i.e., $(u_0 - v_0)^2 - 2(a_0 - a_{\min})(s_0 - s_{\min}) < 0$, or the larger of the real roots < 0, i.e., $-(u_0 - v_0) - \sqrt{(u_0 - v_0)^2 - 2(a_0 - a_{\min})(s_0 - s_{\min})} < 0$. The corresponding safe condition becomes

$$C_2 := s_0 - (u_0 - v_0)^2 - 2(a_0 - a_{\min})(s_0 - s_{\min}) < 0 \cup (u_0 - v_0) + \sqrt{(u_0 - v_0)^2 - 2(a_0 - a_{\min})(s_0 - s_{\min})} > 0$$

The overall set for the safe state is

$$\mathcal{S} = \{s_0 \geq s_{\min}, v_0, u_0 \geq 0 | ((a_0 < 0) \cap C_1) \cup ((a_0 \geq 0) \cap C_2)\}.$$

ACKNOWLEDGMENT

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

REFERENCES

[1] M. Hasenjager, M. Heckmann, and H. Wersing, "A survey of personalization for advanced driver assistance systems," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 2, pp. 335–344, Jun. 2020.

[2] S. E. Shladover *et al.*, "Automated vehicle control developments in the PATH program," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 114–130, Feb. 1991.

[3] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2011.

[4] R. E. Stern *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 205–221, Apr. 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 Intell. Transp. Syst. Mag.*, vol. 12, no. 1, pp. 4–24, Dec. 2019.

[6] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, pp. 1035–1042, Feb. 1995, 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, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000, doi: 10.1103/PhysRevE.62.1805.

[8] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transp. Res. B, Methodol.*, vol. 15, no. 2, pp. 105–111, Apr. 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," *Oper. Res.*, vol. 6, no. 2, pp. 165–184, 1958, doi: 10.1287/opre.6.2.165.

[10] R. E. Wilson and J. A. Ward, "Car-following models: Fifty years of linear stability analysis—A mathematical perspective," *Transp. Planning Technol.*, vol. 34, no. 1, pp. 3–18, Feb. 2011, doi: 10.1080/03081060.2011.530826.

[11] Y. Wang, M. L. D. Monache, and D. B. Work, "Identifiability of car-following dynamic," *Phys. D, Nonlinear Phenomena*, vol. 430, Feb. 2021, Art. no. 133090.

[12] G. Gunter *et al.*, "Are commercially implemented adaptive cruise control systems string stable?" *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6992–7003, Nov. 2021.

[13] M. Makridis *et al.*, "Empirical study on the properties of adaptive cruise control systems and their impact on traffic flow and string stability," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2674, no. 4, pp. 471–484, Apr. 2020, doi: 10.1177/0361198120911047.

[14] B. Ciuffo, M. Makridis, T. Toledo, and G. Fontaras, "Capability of current car-following models to reproduce vehicle free-flow acceleration dynamics," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3594–3603, Nov. 2018.

[15] J. A. Laval, C. S. Toth, and Y. Zhou, "A parsimonious model for the formation of oscillations in car-following models," *Transp. Res. B, Methodol.*, vol. 70, pp. 228–238, Dec. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261514001581

[16] M. Makridis, G. Fontaras, B. Ciuffo, and K. Mattas, "MFC free-flow model: Introducing vehicle dynamics in microsimulation," *Transp. Res. Rec.*, vol. 2673, pp. 762–777, Mar. 2019, doi: 10.1177/0361198119838515.

[17] K. Fadhloun, H. Rakha, A. Loulizi, and A. Abdelkefi, "Vehicle dynamics model for estimating typical vehicle accelerations," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2491, no. 1, pp. 61–71, Jan. 2015, doi: 10.3141/2491-07.

[18] K. Fadhloun and H. Rakha, "A novel vehicle dynamics and human behavior car-following model: Model development and preliminary testing," *Int. J. Transp. Sci. Technol.*, vol. 9, no. 1, pp. 14–28, Mar. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2046043018301631

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

[20] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

[21] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2641–2646.

[22] Z. Zhao, Z. Wang, K. Han, P. Tiwari, G. Wu, and M. Barth, "Personalized car following for autonomous driving with inverse reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2022, pp. 1–7.

[23] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Hoboken, NJ, USA: Wiley, 2013.

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

[25] L. Ljung, "System identification," in *Wiley Encyclopedia of Electrical and Electronics Engineering*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999, pp. 1–19.

[26] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation course notes for MIT 6.832," *Work. Draft Ed.*, vol. 3, pp. 1–109, 2020. [Online]. Available: http://underactuated.mit.edu

[27] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 15, pp. 3932–3937, 2015.

[28] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, Feb. 2008.

[29] R. Babuška, "Neuro-fuzzy methods for modeling and identification," in *Recent Advances in Intelligent Paradigms and Applications*. Berlin, Germany: Springer, 2003, pp. 161–186.

[30] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

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

[32] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5777–5783, Mar. 2018.

[33] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 7130–7135.

[34] 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 *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 1424–1431.

[35] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with Gaussian processes," *Math. Comput. Model. Dyn. Syst.*, vol. 11, no. 4, pp. 411–424, 2005, doi: 10.1080/13873950500068567.

[36] H. Fan and M. S. Poole, "What is personalization? Perspectives on the design and implementation of personalization in information systems," *J. Organizational Comput. Electron. Commerce*, vol. 16, no. 3, pp. 179–202, 2006.

[37] Y. Wang, Z. Wang, K. Han, P. Tiwari, and D. B. Work, "Personalized adaptive cruise control via Gaussian process regression," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 1496–1502.

[38] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Berlin, Germany: Springer, 2004, pp. 63–71, doi: 10.1007/978-3-540-28650-9_4.

[39] M. P. Deisenroth, "Efficient reinforcement learning using Gaussian processes," Ph.D. dissertation, Karlsruhe Ser. Intell. Sensor-Actuator-Syst./Karlsruher Institut für Technologie, Intell. Sensor-Actuator-Systems Lab., Karlsruhe, Germany, 2010.

[40] 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, Germany: Springer, 2011, pp. 312–321.

[41] T. Ma and B. Abdulhai, "Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1800, no. 1, pp. 6–15, Jan. 2002, doi: 10.3141/1800-02.

[42] H. Wang, W. Wang, J. Chen, and M. Jing, "Using trajectory data to analyze intradriver heterogeneity in car-following," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2188, no. 1, pp. 85–95, 2010, doi: 10.3141/2188-10.

[43] B. Ciuffo and V. Punzo, "'No free lunch' theorems applied to the calibration of traffic simulation models," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 553–562, Apr. 2014.

[44] V. Papathanasopoulou and C. Antoniou, "Towards data-driven car-following models," *Transp. Res. C, Emerg. Technol.*, vol. 55, pp. 496–509, Jun. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X15000716

[45] G. Gunter, R. Stern, and D. B. Work, "Modeling adaptive cruise control vehicles from experimental data: Model comparison," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3049–3054.

[46] M. Stepančič and J. Kocijan. (2017). *Gaussian Process Model-Based System Identification Toolbox for MATLAB*. [Online]. Available: https://github.com/Dynamic-Systems-and-GP/GPdyn

[47] F. de Souza and R. Stern, "Calibrating microscopic car-following models for adaptive cruise control vehicles: Multiobjective approach," *J. Transp. Eng., A, Syst.*, vol. 147, no. 1, Jan. 2021, Art. no. 04020150.

[48] A. Girard, "Approximate methods for propagation of uncertainty with Gaussian process models," Ph.D. dissertation, Dept. Comput. Sci. Univ., Univ. Glasgow, Glasgow, U.K., 2004.

[49] N. Arechiga and B. Krogh, "Using verified control envelopes for safe controller design," in *Proc. Amer. Control Conf.*, Jun. 2014, pp. 2918–2923.

[50] P. Nilsson *et al.*, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 1294–1307, Jul. 2016.

[51] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, Jul. 2021, Art. no. 109597.

[52] 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, Switzerland: Springer, 2020, pp. 111–119.

[53] Z. Wang *et al.*, "Driver behavior modeling using game engine and real vehicle: A learning-based approach," *IEEE Trans. Intell. Veh.*, vol. 5, no. 4, pp. 738–749, Dec. 2020.

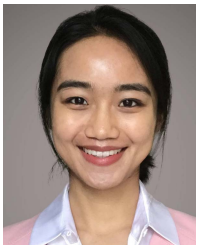
[54] Z. Wang *et al.*, "Cooperative ramp merging system: Agent-based modeling and simulation using game engine," *SAE Int. J. Connected Automated Vehicles*, vol. 2, no. 2, pp. 1–16, May 2019.

[55] 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 *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 182–187.

[56] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," 2017, *arXiv:1711.03938*.

[57] G. Rong *et al.*, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

[58] Z. Wang, K. Han, and P. Tiwari, "Digital twin simulation of connected and automated vehicles with the unity game engine," in *Proc. IEEE 1st Int. Conf. Digit. Twins Parallel Intell. (DTPI)*, Jul. 2021, pp. 1–4.
[59] Y. Wang, G. Gunter, M. Nice, M. L. D. Monache, and D. B. Work, "Online parameter estimation methods for adaptive cruise control systems," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 2, pp. 288–298, Jun. 2021.
[60] R. Jiang, M.-B. Hu, H. M. Zhang, Z. Y. Gao, and Q.-S. Wu, "On some experimental features of car-following behavior and how to model them," *Transp. Res. B, Methodol.*, vol. 80, pp. 338–354, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261515001782



Yanbing Wang received the B.S. degree from the University of Illinois at Urbana-Champaign in 2018. She is currently pursuing the Ph.D. degree in civil and environmental engineering with the Institute for Software Integrated Systems, Vanderbilt University. She was a Research Intern with InfoTech Labs of Toyota Motor North America. Her research interests include systems identification and control for autonomous vehicles. She was a recipient of the Eisenhower Graduate Fellowship from 2018 to 2021.

Ziran Wang (Member, IEEE) received the B.E. degree from the Beijing University of Posts and Telecommunications in 2015 and the Ph.D. degree from the University of California at Riverside, Riverside, in 2019. He is currently a Principal Researcher with InfoTech Labs, Toyota Motor North America, Silicon Valley, where he conducts research in the "Digital Twin" Project. His research focuses on intelligent vehicle technology, including cooperative automated driving, driver behavior modeling, and vehicular cyber-physical systems. He serves as a member for four other technical committees across IEEE and SAE, the Founding Chair for the IEEE Technical Committee on Internet of Things in Intelligent Transportation Systems (IoT in ITS), and an Associate Editor for *SAE International Journal of Connected and Automated Vehicles*.

Kyungtae (KT) Han (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin in 2006. He is currently a Senior Principal Scientist with InfoTech Labs, Toyota Motor North America. Prior to joining Toyota, he was a Research Scientist at Intel Labs and the Director of 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 the Rensselaer Polytechnic Institute in 2004 and the M.B.A. degree from The University of Chicago in 2016. He is currently the Executive Director of InfoTech Labs, Toyota Motor North America. He is highly active in Automotive Edge Computing Consortium (AECC) and SAE. Prior to joining Toyota, he held several leadership positions of increasing responsibilities at GE and UTC aerospace systems.

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