Personalized Adaptive Cruise Control via Gaussian Process Regression

Yanbing Wang*† Ziran Wang† Kyungtae Han† Prashant Tiwari† Daniel B. Work*

***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 components, as a means to make the current technologies more acceptable and trustworthy for users, has only recently been gaining momentum. In this work we develop an algorithm for learning personalized longitudinal driving behaviors via a Gaussian Process (GP) model. The proposed method learns from the individual driver's naturalistic car-following behaviors, and outputs a desired acceleration that suits the user's preference. The learned model can be used as a personalized adaptive cruise control (GP-PACC). The proposed GP-PACC is evaluated both with synthetic car-following data as well as driving simulation data obtained from the Unity game engine. Results show that the GP-PACC can accurately reproduce the acceleration and space gap trajectories even with reasonable measurement noises, and can capture the driving styles of a human driver up to 80% more accurately than baseline models such as an optimal velocity model and an intelligent driver model.**

I. Introduction

A. Motivation

Personalization has gained an increasing attention in the automotive industry. However, the industry-level personalized features are 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 ADAS [1].

As one of the most common ADAS functionalities, adaptive cruise control (ACC) automatically adjusts the longitudinal speed to maintain a safe distance from the vehicle ahead. ACC has been shown to enhance driving comfort, reduce fuel consumption, and increase safety [2]–[5]. However, the limited headway settings prevent the drivers to preserve their own driving 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 depending on external factors such as the road condition and weather, as well as internal factors such as their mood. Therefore, personalization in ACC has the potential to capture the adapting preference of the driver, and adjust the settings to suit users' needs.

Corresponding author: Yanbing Wang, yanbing.wang@vanderbilt.edu

*Department of Civil and Environmental Engineering, and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37240.

†InfoTech Labs, Toyota Motor North America R&D, Mountain View, CA 94043.

B. Related Work

The most common longitudinal control laws are derived from optimal control (e.g., optimal velocity relative velocity as a proportional-derivative (PD) controller, based on constant time gap [6]). These physics-based control laws 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, such as the optimal velocity model (OVM) [7], the intelligent driver model (IDM) [8], the Gipps model [9], and the Gazis-Herman-Rothery (GHR) model [10]. The ODE-based car-following models for describing driving behaviors allow them to work in traffic microsimulation, as well as providing provable and interpretable properties such as rational driving, stability [11] and identifiability [12]. However, human-like 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 styles 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) [13], [14] 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 [15]. 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. [16], [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 the individual's driving behaviors. Instead of the explicit personalization (i.e., offering drivers to choose from a number of predefined system settings), we focus on implicit personalization (estimating the drivers' preferences based on their past behaviors) [27]. GP regression can be utilized to identify the relationship between input (driver's perceived information) and output (desired acceleration), which allows it to provide personalized guidance towards driving.

C. Contribution

Compared to the existing literature that studied the personalized ACC design, we make the following contributions:

- We design a GP-based personalized adaptive cruise controller (GP-PACC) that allows learning the implicit longitudinal human driving styles without categorizing it based on predefined rules. This approach is purely data-driven, and allows each user to have a unique hyperparameter set that characterize his/her driving profile.
- We validate the proposed GP-PACC on both the synthetic car-following data and the naturalistic human-driving data collected from a Unity game engine. Results show that GP-PACC can almost exactly recovers the synthetic car-following data even under reasonable measurement noises, and can capture the driving styles of a real human driver up to 80% more accurately than calibrated baseline car-following models.

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-driving experiments on a game engine 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 is 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 using the GP model.

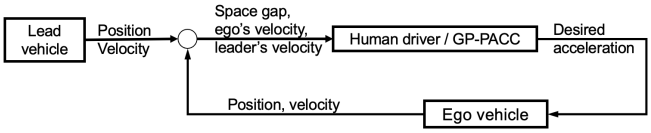

Fig. 1: Block diagram of the proposed GP-PACC system.

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 driver's naturalistic car-following styles.

C. The Car-Following Dynamics

The driver's longitudinal acceleration depends on the vehicle state in relation to the preceding 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. For this work we assume that there is no actuator delay or reaction delay. These assumptions can be relaxed if command acceleration data and full vehicle dynamics model are available. The 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 Figure 1. We consider the ACC algorithm as the high-level controller, which takes the input of the ego vehicle speed, preceding 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 the GP regression both as a modeling tool and as a controller that will be used to model the driving behaviors. The GP regression has been discussed in many standard textbooks such as [28]–[30]. 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 the GP regression to model the personalized longitudinal acceleration. We focus on the high-level driving behavior modeling without considering the low-level vehicle dynamics, i.e., the GP models the mapping from the driver's perceived states (e.g., space gap and the relative speed) to the actual acceleration of the vehicle.

GPs 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 $f_{\text{GP}}(\mathbf{z})'s$ are random variables and they are jointly Gaussian distributed with $\mathbf{z}'s$, which are also assumed to be random variables [28].

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 and only if $i = j$ and 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:

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

If assuming unknown prior $p(\theta)$ (uniform distributed $p(\theta)$), 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 is equivalent to minimizing the negative log likelihood:

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

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:

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

Since regressing on the acceleration data tends to lead to higher error in speed and space gap, we adopt a nonlinear output-error (NOE) approach to improve the training accuracy. The training process is described next.

B. GP-NOE Training

We adopt a training process similar to calibrating an ODE-based car-following model [31]–[35]. 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) will be used as part of the pseudo training input of the GP-NOE model. The simulated state can be obtained via:

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

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 simulation also requires having an initial guess of the hyperparameters 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 $\mathbf{y}_{1:N}$.

The algorithm for GP-NOE training is the following: 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:

Algorithm 1 GP-NOE training

1: **set** input data, target data, covariance function, initial hyperparameters
2: **while** $l(\theta)$ (11) differs from the previous evaluation **do**
3: **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).
4: **update** θ by minimizing the negative log likelihood $l(\theta)$.
5: **end while**

The implementation is based on the GP-Model-based System-Identification Toolbox for Matlab [36].

C. GP Model Validation

Training a GP-NOE model is very much 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 experimental data. 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., [31]–[35], [37]).

Two performance metrics are measured, the mean squared error (MSE) and the log predictive-density error (LPD) [30], [38] between the GP simulated acceleration and the recorded acceleration of a validation data set:

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

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.

IV. Experiments and Results

In this section, we provide the validation of GP-PACC on two 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 the human-driving simulation data generated using the Unity game engine driving platform for a more naturalistic driving scenario.

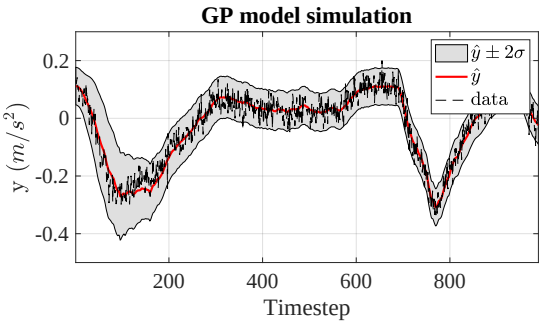

Fig. 2: Compare GP acceleration prediction to the synthetic data.

A. Numerical Experiments

A set of car-following data is synthetically generated using an IDM [8]. It has been used throughout the literature to model a realistic driver behavior, such as asymmetric accelerations and decelerations. The acceleration is expressed as:

$$\begin{aligned}\dot{v}(t) &= f_{\text{CF}}(s(t), v(t), u(t)) \\ &= a\left[1 - \left(\frac{v(t)}{v_f}\right)^{\delta} - \left(\frac{s^*(v(t), u(t))}{s(t)}\right)^2\right]\end{aligned} \tag{16}$$

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

$$s^*(v(t), u(t)) = s_j + v(t)T + \frac{v(t)(v(t) - u(t))}{2\sqrt{ab}}. \tag{17}$$

The parameters of the model are the acceleration exponent δ, freeflow speed v_f, the desired time gap T, the jam distance s_j, the maximum acceleration a and the desired deceleration b. We fixed the parameters to simulate human-driving data using $\theta = [s_j, v_f, T, a, b, \delta] = [2, 33.3, 1.6, 0.73, 1.67, 4]$ based on empirical investigations [8].

We generate 200 seconds of data at 10Hz given a pre-recorded, freeway high-speed preceding vehicle speed profile ranging between 25m/s to 35m/s. The simulated data is also manually polluted with added 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.

Figure 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 an IDM and manually polluted with $0.03m/s^2$ standard deviation of Gaussian white noise. 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.

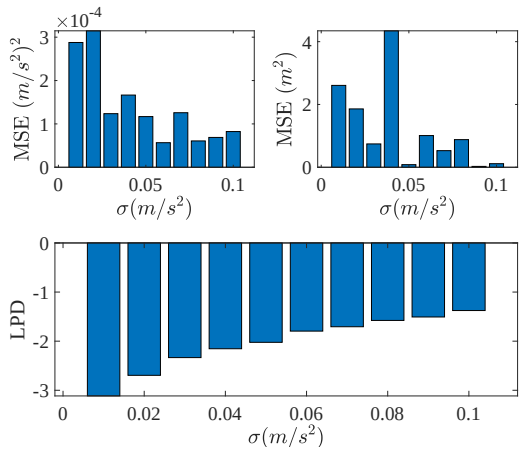

Fig. 3: Performance of GP-PACC compared with synethetic data. Upper left: MSE on the acceleration prediction; upper right: MSE on the space gap prediction; bottom: LPD on the acceleration prediction.

More quantitatively, Figure 3 shows the MSE of the GP simulation on the acceleration 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 space gap prediction MSE (under $4.5m^2$). It indicates that the GP model can very accurately reproduce the driving profile and is robust under noisy measurements.

A counter intuitive discovery from Figure 3 is that, as the standard deviation of the added noise increases (emulating the real-life acceleration measurement has higher noise), the MSE's for both the acceleration and the space gap prediction appear to be lower. This could be due to two reasons: (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, and (b) the 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 Figure 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. These results further indicate that the proposed GP model can capture the implicit driving styles without any pre-defined rules such as the time-headway, following distances, or acceleration/breaking dynamics. In other words, the 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. 4: Naturalistic driving in a car-following scenario with Unity game engine and Logitech racing wheel.

In this section we demonstrate the possibility of the GP to learn the IDM dynamic as a numerical test. Next we learn real driving behavior by using the Unity game platform to generate the true data rather than the IDM.

B. Human-Driving Experiments on 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 [39], with studies conducted for driver behavior modeling [40], connected vehicle systems prototyping [41], [42], and autonomous driving simulation [43], [44].

In this study, the realistic human-driving 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. 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 Figure 4.

b) Data preparation: The experiment trip resembles a freeway high-speed scenario, and has a total period of 200 seconds, where the first 80 seconds is designed as a warm-up phase for drivers to familiarize with the engine and get to a comfortable car-following setting. The next 60 seconds is for training and the 60 seconds after is used for validation. The preceding vehicle drives at a time-varying speed within the range 25-35m/s that captures a naturalistic freeway acceleration and deceleration scenario. The data is recorded in 1Hz. The training input and target are organized

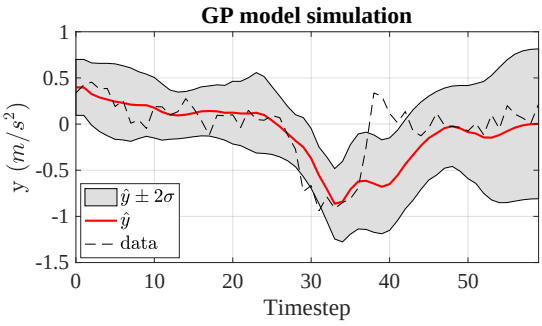

Fig. 5: Compare the GP-PACC guided acceleration (red) with the actual acceleration recorded by Unity (dotted).

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, Figure 5 shows the GP simulated acceleration and Unity recorded acceleration. The mean prediction (red line) generally aligns well with the recorded data (dashed line), and the uncertainty captures the variation of the recorded data, with a few exceptions at around timestep 40.

In addition, we train three analytical car-following models with the same training data, and calculated the MSE on the acceleration and space gap using the same validation data. First model is the *constant-time headway relative-velocity* (CTH-RV) model [35], [45]–[49], the second one is the OVM [7], [50], and the last is IDM [8]. As shown in Table I, GP significantly outperforms other three car-following models with low MSE errors on both acceleration and space gap (up to 80% improvement on acceleration prediction). CTH-RV, OVM and IDM fail to accurately reproduce the space-gap trajectories. 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 have a cleaner relationship between the input variable (s, v, u) and the output (acceleration), that can be captured by the squared-exponential covariance function (3); On the other hand, naturalistic human driving data contains more randomness and inconsistent patterns even during the same trip. GP modeling of human-driving shows promising results, even with no assumptions on the individual's driving styles.

V. Conclusion and Future Work

In this article we propose GP-PACC that mimics individual's car-following behavior. The learning is achieved using a Gaussian Process regression on the car-following data. We explore this purely data-driven controller design to capture individual's driving styles, which sometimes cannot be captured by an explicit car-following model. The GP regression learns the driving behavior without making assumptions such as time-headway or preferred following distance.

Model	GP	CTH-RV	OVM	IDM
MSE - acceleration	0.0553	0.2831	0.0933	0.1171
MSE - space gap	81.6	5216	365.3	1220
LPD - acceleration	-0.0023	N/A	N/A	N/A

TABLE I: GP vs. other analytical car-following-model-based controllers

The training result shows that GP has the potential to provide realistic acceleration guidance that closely resembles individual's acceleration profile. Specifically, GP almost exactly recovers the car-following profiles of an IDM driver (data generated using an IDM), and reproduces the naturalistic human-driving data up to 80% more accurately than some well-known car-following models. More experiments need to be conducted with a diverse collection of drivers, road conditions and weather conditions. With integrating the proposed GP-PACC into the Unity simulation platform, human-in-the-loop testing on the acceptance of the proposed controller can also be achieved.

Acknowledgment

This material 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.

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, K. Han, B. Kim, G. Wu, and M. J. Barth, "Lookup table-based consensus algorithm for real-time longitudinal motion control of connected and automated vehicles," in *2019 American Control Conference (ACC)*, pp. 5298–5303, 2019.

[6] G. Burnham, Jinbom Seo, and G. Bekey, "Identification of human driver models in car following," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 911–915, 1974.

[7] 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.

[8] 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.

[9] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[10] 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.

[11] 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.

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

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

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

[15] 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)*, pp. 2641–2646, 2015.

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

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

[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*, pp. 161–186, Springer, 2003.

[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)*, pp. 3420–3431, IEEE, 2019.

[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)*, pp. 7130–7135, IEEE, 2018.

[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*, pp. 1424–1431, IEEE, 2014.

[27] 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.

[28] C. E. Rasmussen, *Gaussian Processes in Machine Learning*, pp. 63–71. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[29] M. P. Deisenroth, *Efficient Reinforcement Learning using Gaussian Processes*. PhD thesis, 2010.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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)*, pp. 3049–3054, 2019.

[36] M. Stepančič and J. Kocijan, "Gaussian process model-based system identification toolbox for matlab," 2017.

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

[38] A. Girard, *Approximate methods for propagation of uncertainty with Gaussian process models*. PhD thesis, Citeseer, 2004.

[39] 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), pp. 111–119, Springer International Publishing, 2020.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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)*, pp. 1–6, IEEE, 2020.

[45] V. Milanés and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 285–300, 2014.

[46] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, 2014.

[47] 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.

[48] Z. Bareket, P. S. Fancher, Huei Peng, Kangwon Lee, and C. A. Assaf, "Methodology for assessing adaptive cruise control behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 123–131, 2003.

[49] C.-Y. Liang and H. Peng, "Optimal adaptive cruise control with guaranteed string stability," *Vehicle System Dynamics*, vol. 32, no. 4-5, pp. 313–330, 1999.

[50] 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.