Personalized Car Following for Autonomous Driving with Inverse Reinforcement Learning

Zhouqiao Zhao*†, Ziran Wang*, Kyungtae Han*, Rohit Gupta*, Prashant Tiwari*,
Guoyuan Wu†, and Matthew J. Barth†

***Abstract*— Driving automation is gradually replacing human driving maneuvers in different applications such as adaptive cruise control and lane keeping. However, contemporary driving automation applications based on expert systems or predefined control strategies are not in line with individual human driver's preference. To overcome this problem, we propose a Personalized Adaptive Cruise Control (P-ACC) system that can learn the driver's car-following preferences from historical data using model-based maximum entropy Inverse Reinforcement Learning (IRL). Once activated in real-time, the P-ACC system first classifies the driver type and the weather type (at that moment). The vehicle is then controlled using the pre-trained IRL model on the cloud of the associated class. The personalized IRL model on the cloud will be updated as more human driving data is collected from various scenarios. Numerical simulation with real-world naturalistic driving data shows that, the accuracy of reproducing the real-world driving profile improves up to 30.1% in terms of speed and 36.5% in terms of distance gap, when P-ACC is compared with the Intelligent Driver Model (IDM). Game engine-based human-in-the-loop simulation demonstrates that, the takeover frequency of the driver during the usage of P-ACC decreases up to 93.4%, compared with that during the usage of IDM-based ACC.**

I. Introduction

Expert systems are commonly used to build classic driving automation. An Adaptive Cruise Control (ACC) system, for example, allows the ego vehicle to travel at a set speed and/or maintain a desirable constant time headway with its preceding vehicle based on the measurements from its ranging sensors (e.g., radar) and its own states. Currently, ACC is one of the most commonly used Advanced Driver-Assistance Systems (ADAS), which has gained its market penetration rate over the last decade [1]–[3].

However, the settings of existing ACC systems are limited with only a few options to choose from (e.g., long, medium, and short), which do not consider individual customer's driving style and sometimes turn out to be excessively aggressive/conservative for different drivers. When environmental factors such as weather conditions are taken into account, the same driver's car-following preference may also vary. Therefore, the predetermined rule-based system is not able to accommodate diverse driving styles, and the driver may eventually disregard the ACC system due its unsatisfactory user experience. Adding personalized features to the ACC system can have a significant impact on user's acceptance and trust in the system.

Physics-based control policies are one of the most prevalent longitudinal control methods, where the most common ones are using an Ordinary Differential Equation (ODE) to model car following. The ODE equation tries to explicitly depict the driver's interaction with the preceding vehicle considering the dynamics of the vehicle. Based on the given distance gap, speed of the ego vehicle, and the speed of the preceding vehicle, the acceleration of the ego vehicle can be calculated from the ODE to enable the car-following behavior. The most widely used models include IDM [4], Gipps model [5], and Newell's car-following model [6].

Another physics-based controller that models the dynamics of the car-following system in the state space, and the acceleration of the ego vehicle is the Model Predictive Controller (MPC), which optimizes the predefined objectives such as safety, comfort and fuel economy requirements [7], [8]. However, both the ODE and the objective function for MPC require prior knowledge of the system, and they are difficult to capture personalized behaviors.

The learning-based methods, on the other hand, model the car-following behaviors directly from the demonstration trajectories, so they can better learn the personalized driving styles. In general, the learning-based method can be classified into two categories. The first one is *Imitation Learning*, where the models clone the behavior of the demonstrations by learning the mapping from states to actions. For example, Deep Neural Network (DNN) and Gaussian Mixture Model (GMM) have been used to model car-following behaviors like [9], [10]. Also, because the decision-making process may depend on the sequential state inputs, the networks with memory, such as Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) have been used [11], [12]. However, as the demonstration trajectory is not able to visit all possible states, extrapolation is needed at the inference stage. As a result, the system's performance and even safety cannot be guaranteed. Toward this end, a safety filter has to be implemented as discussed in [13].

The second category of the learning-based method is *Apprenticeship Learning*, where the models first learn the

Corresponding author: Zhouqiao Zhao, zzhao084@ucr.edu
*InfoTech Labs, Toyota Motor North America R&D, Mountain View, CA 94043.
†Department of Electrical and Computer Engineering, and the Center for Environmental Research and Technology, University of California, Riverside, CA 92507.

reward/objective of the demonstrations using IRL [14]–[16] or Inverse Optimal Control (IOC) [17], and then Reinforcement Learning (RL), MPC, or other controllers can be implemented based on the recovered reward/objective. Instead of copying the movement directly, the apprenticeship learning method infers the preference of the agent before making decisions. Therefore, better performance for unseen scenarios is expected. Also, because the design of the controller is decoupled with the modeling part, both the stability and safety can be proved mathematically based on the selection of the controller.

In this work, we propose a Personalized Adaptive Cruise Control (P-ACC) system to learn from the realistic car-following behaviors of individual drivers. The proposed methodology is based on Inverse Reinforcement Learning (IRL) [15], which infers the reward function of a target agent given the demonstration trajectories. Instead of directly mimicking the behavior of the demonstrations, the recovered reward function can help explain not only the observed demonstrations, but also the representation of the preference of the agent doing the specific task. Therefore, the reward is an ideal representation of the personalized driving style of each driver for the proposed P-ACC system.

Compared to the existing literature that studied the personalization of ACC systems, we make the following contributions in this work:

- The driving preferences of a driver is modeled by the proposed IRL algorithm after classifying the data by its driver type and weather type.
- The maximum cumulative reward criterion is used for choosing the most suitable pre-trained models for new drivers, which enables the real-time implementation of the proposed system.
- The real-world naturalistic driving data is collected by a test vehicle and then used for training and validation.
- The comfort and trust levels of drivers on the P-ACC system are quantitatively measured by the takeover frequency while testing the system on the human-in-the-loop simulator.

II. Problem Formulation

A. System Architecture

The architecture of the proposed P-ACC system is shown in Fig. 1. Different from existing research that considers only the driving trajectories, driver types and weather types are also utilized by the system. A cloud server built by our previous work [18], namely "Digital Twin", is used to train and store the personalized models.

The offline training process (i.e., P-ACC off) is depicted by the blue pipeline in Fig. 1. The raw trajectories are first classified into different subcategories based on the driver type and weather type, where the details are introduced in section III-A. Then, the trajectories are used to train the associated

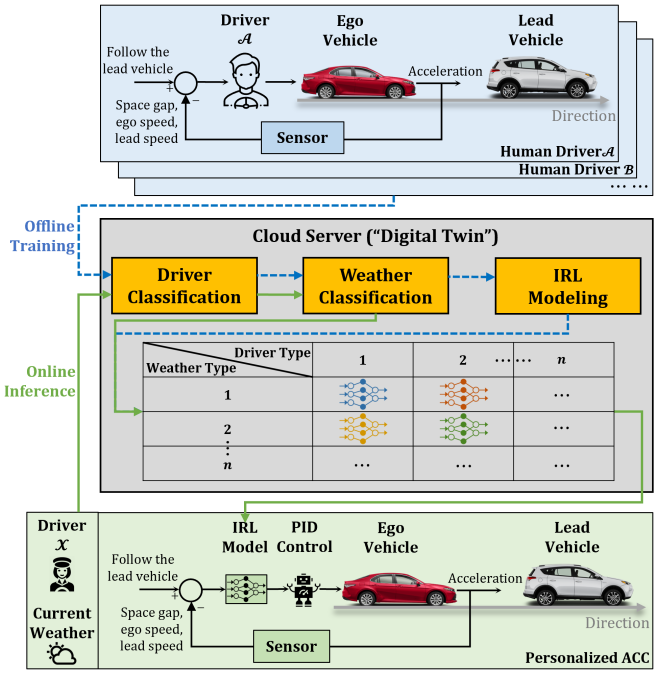

Fig. 1. System architecture of the proposed personalized adaptive cruise control (P-ACC) system.

IRL models, which are stored as a two-dimensional table on the "Digital Twin".

The online inference process (i.e., P-ACC on) is depicted by the green pipeline in Fig. 1. The system first samples a short manually driving trajectory and sends it along with the current weather condition to the cloud server. Similar to the offline training process, the information goes through the driver classification module and weather classification module to determine which subcategories this driver belongs to. Then, the corresponding IRL model is sent back to the vehicle to instruct the local controller.

B. Car-Following Dynamics

In this work, we assume the system is governed by the second-order dynamics:

$$x = \begin{pmatrix} p \\ v \\ g \end{pmatrix} \tag{1}$$

$$\begin{aligned} \dot{x} = \begin{pmatrix} \dot{p} \\ \dot{v} \\ \dot{g} \end{pmatrix} &= \begin{pmatrix} v \\ a \\ v_f - v \end{pmatrix} \\ &= A \begin{pmatrix} p \\ v \\ g \end{pmatrix} + B \cdot a + C \cdot v_f \end{aligned} \tag{2}$$

where p and v are the position and speed of the ego vehicle, respectively; g is the distance gap between the ego vehicle and the preceding vehicle; a is the acceleration of the ego vehicle; and v_f is the speed of the preceding vehicle.

In addition, the car-following behavior can be described as a Markov-Decision Process (MDP), which is defined as a tuple $\{S, A, T, r, \gamma\}$. S is the state space spanned by v and g; A is the action space spanned by a. T is the transition probability matrix; r is the reward function that can represent the driver's personalized driving style when performing the car-following task; γ is the discount rate representing the degree of emphasis that the driver has on the previous reward; To define the transition matrix T, we discretize the car following kinematics as follows:

$$v(t+1) = v(t) + a(t) \cdot \Delta t + \sigma_v \tag{3}$$

$$g(t+1) = g(t) + [v_f(t) - v(t)] \cdot \Delta t + \frac{1}{2} \cdot a(t) \cdot \Delta t^2 + \sigma_g \tag{4}$$

Because the driver may not be able to observe the states or perform actions perfectly, the Gaussian noises, σ_v and σ_g, are added to both the transition of v and g. It should also be noted that the speed of the preceding vehicle, v_f, is not part of the MDP, but can be obtained while driving.

C. Assumptions and Specifications

We define the discrete 2D state space based on the range of v and g for the car-following task. v ranges from 0 to 36m/s with the interval of 0.5m/s, while g ranges from 0 to 120m with the interval of 0.5m.

Also, we presume that the human driver is rational and that his or her actions are near-optimal in terms of the cumulative reward function defined as follows:

$$v(\xi) = \sum_{t=0}^{N} \gamma^t \cdot r_t(s) = \sum_{t=0}^{N} \gamma^t \cdot \boldsymbol{\alpha}^T \cdot \boldsymbol{\Phi}(s) \tag{5}$$

where ξ is the car-following trajectory, and N is the time horizon. As shown in Equation (5), we assume that the instantaneous reward r can be expressed in the span of the reward basis $\boldsymbol{\Phi}$, whose dimension equals to the number of features, and $\boldsymbol{\alpha}$ is a vector denoting the weight of each reward basis.

As recommended by [16], we use a Gaussian-like kernel function to be the reward basis to improve the nonlinear representation ability. For each state s, its relationship with any n state, s_i, in the state space is mapped using the kernel function as follows:

$$\boldsymbol{\Phi}(s) = \begin{bmatrix} \Phi_1(s) \\ \Phi_2(s) \\ \cdots \\ \Phi_n(s) \end{bmatrix} = \begin{bmatrix} \mathrm{K}(s, s_1) \\ \mathrm{K}(s, s_2) \\ \cdots \\ \mathrm{K}(s, s_n) \end{bmatrix} \tag{6}$$

$$\mathrm{K}(s, s_i) = \exp\left(-\frac{|s - s_i|^2}{\sigma^2}\right) \tag{7}$$

$$|s - s_i|^2 = (v - v_i)^2 + (g - g_i)^2 \tag{8}$$

It should be noted that the value of σ is selected manually based on the resolution of the state space to obtain a balance between underfitting and overfitting.

III. Methodology

In this section, we introduce the proposed system in detail for both modeling the driver's preference and controller that will be used to follow the preceding vehicle. We first discuss the data clustering in the perspective of environmental factors and driver's type in Section III-A. Then, we describe the specific training and validation procedures of IRL in Section III-B. Next, the method of personalized desired gap calculation is discussed in Section III-C. Finally, Section III-D elaborates the controller designed to calculate the acceleration for the ego vehicle based on the personalized desired gap table.

A. Data Clustering

For large scale implementation of the proposed system, training and storing the personalized models for all drivers individually is not time-efficient or storage-efficient. Also, it would become extremely difficult to select suitable pre-trained models for the new driver or the driver in the new scenarios. On the other hand, the preference of the similar drivers is relatively close. This similarity can be easily defined using the unsupervised clustering algorithm. As distinct environmental circumstances such as weather conditions change, different drivers can have different driving styles. In other words, driving in different weather conditions can be regarded as different tasks for the same driver.

Therefore, we propose to use **Algorithm 1** to classify the data before training. The aim of this procedure is to assign driver type and weather type to the demonstration trajectories. Also, the classification boundary is learnt for online inference. For each trajectory, the weather condition can be acquired from the OpenWeatherMap API [19] given the initial time and position. Then, we select the car-following related features, i.e., average speed and average time gap, to classify the drivers. If the instant speed of a trajectory is less than the threshold τ, the calculation of the time gap will be unreliable. Therefore, we discard those data points as shown in line 6 to line 11. The extracted feature for driver and weather type classification is packed in line 17 and 18. Using the K-mean algorithm with predefined number of classes, the label of each trajectory can be assigned. Finally, we apply Support Vector Machine (SVM) to learn the classification boundary.

B. Car-Following Preference Modeling Using IRL

According to the maximum entropy IRL (Max-Ent IRL) proposed by Ziebart [15], the probability of a trajectory is proportional to the sum of the exponential rewards accumulated along the trajectory as described in Equation (9).

$$\begin{aligned} p(\xi \mid \alpha) &= \frac{1}{Z(\alpha)} \exp\left(\sum_t R_\alpha(s_t)\right) \\ &= \frac{1}{Z(\alpha)} \exp\left(\sum_t \boldsymbol{\alpha}^T \boldsymbol{\Phi}(s_t)\right) \end{aligned} \tag{9}$$

Algorithm 1: Data classification based on driver types and weather types

Data: Demonstration trajectories: $\boldsymbol{\xi} = \{\xi_1, \xi_2, ..., \xi_n\}$, Number of driver types: p, Number of weather types: q
Result: Classification boundary: CB, Classification result: $CR = \{[DT_i, WT_i], ...\}_n$, Driver type: DT_i, Weather type: WT_i

```
DT_features = [], WT_features = []
for ξ_i in ξ do
    [rain, snow, cloud, visibility,
     time_of_day]=OpenWeatherMap(ξ_i[0].time,
     ξ_i[0].latitude, ξ_i[0].longitude)
    for time in ξ_i.total_time do
        time_gap_mean, speed_mean = 0
        if ξ_i[time].speed < τ then
            ξ_i.total_time −= 1
            continue
        else
            time_gap = ξ_i[time].gap/ξ_i[time].speed
        end
        time_gap_mean += time_gap
        speed_mean += ξ_i[time].speed
    end
    time_gap_mean /= ξ_i.total_time
    speed_mean /= ξ_i.total_time
    DT_features.push([time_gap_mean, speed_mean])
    WT_features.push([rain, snow, cloud, visibility,
     time_of_day])
end
CR=[k_mean(DT_features, p), k_mean(WT_features, q)]
CB=SVM(CR)
```

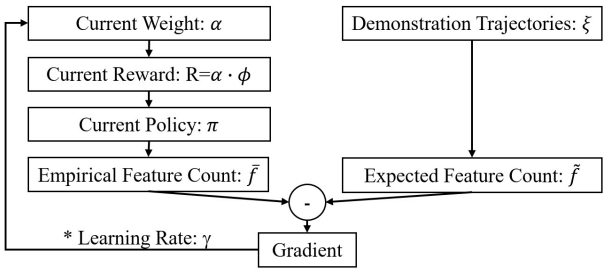

Fig. 2. Flowchart for the maximum entropy IRL (Max-Ent IRL) used in the car-following preference modeling.

where $Z(\alpha)$, called partition function, equals to $\sum_\xi \exp\left(\sum_t R_\alpha(t)\right)$. To recover the reward function, the maximum log likelihood method is used at the demonstration trajectories with respect to the weight of the reward function.

$$L(\alpha \mid \xi) = \max_{\alpha} \sum_{\xi} \log p(\xi \mid \alpha) \tag{10}$$

Then, the gradient of the weight $\boldsymbol{\alpha}$ can be written in the following form:

$$\nabla_\alpha L = \sum_{\xi} p(\xi) \sum_{s \in \xi} \boldsymbol{\Phi}(s) - \sum_{\xi} \mathrm{D}_s \sum_{s \in \xi} \boldsymbol{\Phi}(s) \tag{11}$$

The first term, $\sum_\xi p(\xi) \sum_{s\in\xi} \boldsymbol{\Phi}(s) = \tilde{f}$, is named expected feature count. The second term, $\sum_\xi \mathrm{D}_s \sum_{s\in\xi} \boldsymbol{\Phi}(s) = \bar{f}$, is named empirical feature count, and D_s is the state visitation frequency.

The training process is illustrated in Fig. 2. To calculate the expected feature count, $\tilde{f}$, before the training iterations, we first estimate $p(\xi)$ by calculating the state visitation frequency of the demonstration trajectories. Then, at the first iteration, the reward weight, $\boldsymbol{\alpha}$, is generated randomly. Next, we apply the value iteration algorithm from [20] to recover the value function $v(s)$ and the optimal policy $\pi^*(s, a)$ under the current reward function. The state visitation frequency, D_s, can be calculated by exhaustively traversing the state space with a predefined time horizon given transition probabilities, initial state distribution, and the optimal policy under current reward function [15]. However, this takes extremely long time when the state space is large and/or the time horizon is long. Therefore, in this research, we estimate D_s using the Monte Carlo method, where a number of trajectories are generated randomly governed by the optimal policy of the current iteration. The initial state and the time horizon of each trajectory should randomly match the demonstration trajectories.

C. From Reward to Control Strategy

Based on the recovered reward function from the Max-Ent IRL algorithm, the driver's desired gap at each speed can be calculated using Equation (12)

$$g_{\text{desired}}\,(v) = \arg\max_{g} r(v) \tag{12}$$

Then, the $speed$-$g_{desired}$ table is sent to the vehicle speed controller so that the vehicle can track the desired gap as preferred by the driver. In this work, we implement a PID controller to calculate the suggested acceleration for the vehicle in simulation.

D. Online Inference

Because the training process of IRL is computationally intensive, it cannot enable real-time implementation. The pre-trained models are saved in the cloud, and the corresponding IRL model can be downloaded and applied whenever the driving type has been determined. When a new driver drives in a new scenario for the first time, a sample trajectory data is gathered and uploaded to the cloud. Then the subcategories of models are first selected based on the considered environmental factors. After traversing each selected model and calculating the cumulative reward for the sample trajectory using Equation (5), the one with the highest cumulative reward is chosen as the most appropriate model for usage.

IV. Experiments and Results

A. Numerical Simulation with Naturalistic Driving Data

We collect the naturalistic driving data using a Lexus prototype vehicle in Michigan and California, where two

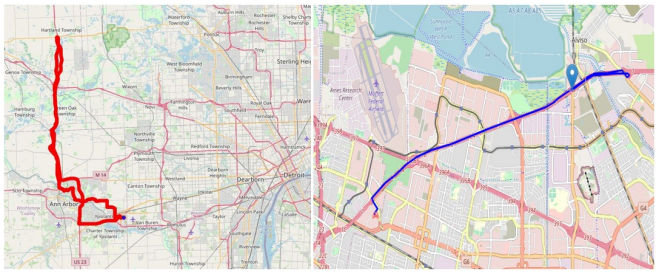

Fig. 3. Example trajectories of the naturalistic driving data collected by our test vehicle in Ann Arbor, MI (left) and Mountain View, CA (right).

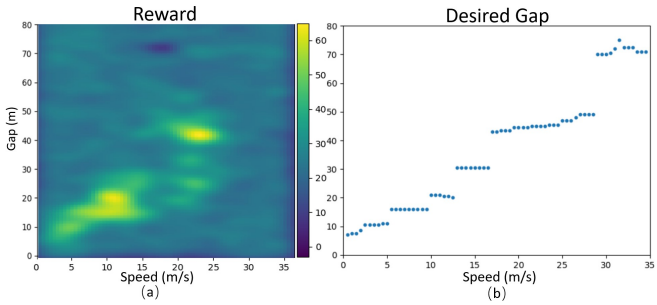

Fig. 4. IRL model visualization: (a) recovered reward function of naturalistic driving data, (b) desired gap of the ego vehicle.

example trajectories are shown in Fig. 3. Based on the front vehicle detection utilizing the ACC radar, the car-following events can be selected from the raw data. Then, the selected trajectories are downsampled at 10Hz for training and validation. To better understand the driver's driving style, the trajectories with full spectrum of speed (ranges from 0 to 35m/s) are selected for IRL modeling.

The recovered reward function in $v-g$ space is shown on the left side of Fig. 4, and the reward value is color coded. The relationship between vehicle speed and desired gap is then calculated as shown on the right side of Fig. 4. As can be observed from this figure, the desired gap increases as the vehicle speeds up.

After training, we select three pieces of leader speed profiles with different characteristics and speed ranges for validation purpose (see the top three plots in Figure 5). We use the root mean square percentage error (RMSPE) of both speed and gap, as defined in Equation (13), to evaluate the accuracy of the proposed IRL-based car-following model.

$$\mathrm{RMSPE}(x) = \sqrt{\frac{\sum_t [\hat{x}(t) - x(t)]^2}{\sum_t x(t)^2}} \tag{13}$$

For comparison, the Intelligent Driver Model (IDM) [4] is also implemented as the baseline. The IDM model is defined by Equation (14) and (15), and the parameters used in this work are listed in TABLE I.

$$\dot{v} = a\left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, v-v_f)}{g}\right)^2\right) \tag{14}$$

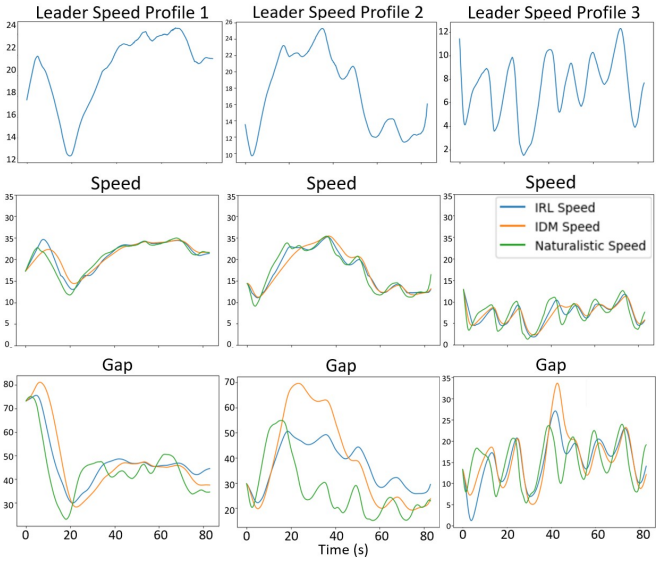

Fig. 5. Results of numerical simulation

TABLE I
IDM PARAMETERS

Variable	Description	Value
v_0	Desired velocity	35 m/s
T	Safe time headway	1.5 s
a	Maximum acceleration	0.73 m/s^2
b	Comfortable Deceleration	1.67 m/s^2
δ	Acceleration exponent	4
s_0	Minimum distance	2 m

TABLE II
RESULTS OF NUMERICAL SIMULATION

	Speed RMSPE			**Gap RMSPE**		
	Profile 1	Profile 2	Profile 3	Profile 1	Profile 2	Profile 3
IRL	5.7%	6.6%	22.6%	16.9%	51.5%	33.5%
IDM	7.8%	10.5%	25.7%	26.6%	71.1%	39.1%
Improvement	26.9%	30.1%	12.1%	36.5%	27.6%	14.3%

$$s^*(v, v-v_f) = s_0 + v\cdot T + \frac{v\cdot(v-v_f)}{2\sqrt{ab}} \tag{15}$$

The lower six plots in Fig. 5 demonstrate the simulated speed and gap of both IRL and IDM compared with the real-world data. It shows that IRL can outperform the IDM model in terms of reproducing the speed and gap trajectories. TABLE II shows the quantitative results measured by RMSPE. As we can see from the table, the IRL model performs better in all three leader speed profiles for both speed and gap.

B. Human-in-the-Loop Simulation with the Game Engine

Game engines enable the design of video games for software developers, which typically consist of a rendering engine for 2-D or 3-D graphics, a physics engine for collision detection and response, and a scene graph for the manage-

Clear Sky (Day) Clear Sky (Night) Foggy

Fig. 6. Three weather conditions are simulated in the game engine: Clear Sky (Day), Clear Sky (Night), and Foggy.

ment of multiple elements (e.g., models, sound, scripting, threading, etc.). Along with the rapid development of game engines in recent years, their functions have been broadened to a wider scope: data visualization, training, medical, and military use. Game engines have also become popular options in the development of advanced vehicular technology [21], which have been used to study driver behaviors [22], prototype connected vehicle systems [23], [24], and simulate autonomous driving [25], [26].

In this work, human-in-the-loop simulations are conducted on a customized driving simulation platform built in our previous work [27]. In the simulation environment, a three-lane freeway scenario is built, and four drivers participate in the test. The first three drivers provide manual driving data for training the IRL models at three different weather conditions, including clear sky (day), clear sky (night), and foggy weather (see Fig. 6). When the trained P-ACC system is activated, the corresponding driver monitors the system and reacts by pushing the acceleration pedal or brake pedal if he/she feels uncomfortable. The fourth driver provides a sample trajectory, and then the most suitable pre-trained model is applied by calculating the cumulative reward as described in Section III-D. Similar to the other three drivers, the reaction of the fourth driver is also recorded for performance measurement.

Fig. 7 shows the desired gap of Driver A in three different weather conditions. As shown in the figure, Driver A prefers to maintain a smaller gap in the clear sky (day) condition, medium gap in the clear sky (night) condition, and larger gap in the foggy weather condition. This reveals that Driver A performs more aggressively when the driving condition (e.g., visibility) is good. Also, in the foggy weather, it can be noticed that the desired gap of about 20m/s differs from the neighboring speed. This shows that the driver does not always perform optimally as preferred, and this imperfection is more significant in the worse driving condition.

The driver's comfort and trust of a P-ACC model can be quantitatively measured using the takeover percentage. Takeover denotes the status where the driver steps onto the acceleration pedal or brake pedal when he/she feels uncomfortable. The takeover percentage is the takeover time divided by the overall P-ACC activation time, and the results are presented in TABLE III. The results show that drivers are satisfied with not only the IRL model trained specifically from his/her demonstrations, but also the pre-trained model

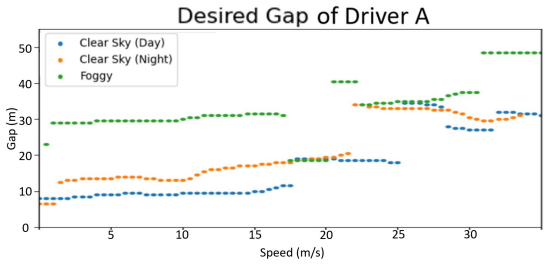

Fig. 7. Desired distance gap of driver A in three different weather conditions.

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

Driver	Max-Ent IRL	IDM	Improvement
A	3.5%	18.5%	81.1%
B	1.5%	17.4%	91.4%
C	1.6%	12.0%	86.7%
D (Untrained)	2.2%	33.5%	93.4%

selected based on the maximum cumulative reward criterion.

V. CONCLUSION AND FUTURE WORK

In this research, we have presented a personalized car-following system, namely P-ACC system. We have first categorized car-following scenarios using environmental factors such as weather conditions. Then, the model-based maximum entropy Inverse Reinforcement Learning has been applied to learn the specific driving style from demonstrations for each subcategory. The relationship between the driver's desired gap and current speed can be derived from the recovered reward function. The PID controller has been implemented so that the ego vehicle can maintain the preferred gap with its leader. The driver without trained model can also use the other driver's model under the same weather conditions using the maximum cumulative reward criterion. Numerical simulation with real-world naturalistic driving data has shown that, the accuracy of reproducing the real-world driving profile improves up to 30.1% in terms of speed and 36.5% in terms of distance gap, when P-ACC is compared with the IDM. Game engine-based human-in-the-loop simulation has demonstrated that, the takeover frequency of the driver during the usage of P-ACC decreases up to 93.4%, compared with that during the usage of IDM-based ACC.

For future work, online IRL training can be incorporated into the current system so that the driving style can be learnt incrementally. In addition, if the local driving model changes, the federated learning functionality is expected to adjust the models on the clouds accordingly. Finally, instead of only using the takeover as the measurement of the driver's degree of satisfaction, it can be considered as the feedback to the online IRL process while the P-ACC is activated.

REFERENCES

[1] Toyota. (2021) Toyota Safety Sense: The Standard for Safety. [Online]. Available: https://www.toyota.com/safety-sense/

[2] Volkswagen. (2021) Adaptive Cruise Control. [Online]. Available: https://www.volkswagen-newsroom.com/en/adaptive-cruise-control-acc-3664

[3] Ford. (2021) Adaptive Cruise Control. [Online]. Available: https://www.ford.com/technology/driver-assist-technology/adaptive-cruise-control/

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

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

[6] G. F. Newell, "A simplified car-following theory: a lower order model," *Transportation Research Part B: Methodological*, vol. 36, no. 3, pp. 195–205, 2002.

[7] A. Khodayari, A. Ghaffari, M. Nouri, S. Salehinia, and F. Alimardani, "Model predictive control system design for car-following behavior in real traffic flow," in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*. IEEE, 2012, pp. 87–92.

[8] B. Gao, K. Cai, T. Qu, Y. Hu, and H. Chen, "Personalized adaptive cruise control based on online driving style recognition technology and model predictive control," *IEEE transactions on vehicular technology*, vol. 69, no. 11, pp. 12 482–12 496, 2020.

[9] P. Angkititrakul, C. Miyajima, and K. Takeda, "Modeling and adaptation of stochastic driver-behavior model with application to car following," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 814–819.

[10] J. Hongfei, J. Zhicai, and N. Anning, "Develop a car-following model using data collected by" five-wheel system"," in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, vol. 1. IEEE, 2003, pp. 346–351.

[11] L. Chong, M. M. Abbas, and A. Medina, "Simulation of driver behavior with agent-based back-propagation neural network," *Transportation Research Record*, vol. 2249, no. 1, pp. 44–51, 2011.

[12] X. Huang, J. Sun, and J. Sun, "A car-following model considering asymmetric driving behavior based on long short-term memory neural networks," *Transportation research part C: emerging technologies*, vol. 95, pp. 346–362, 2018.

[13] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[22] 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*, pp. 1–1, 2020.

[14] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[15] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[16] H. Gao, G. Shi, G. Xie, and B. Cheng, "Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making," *International Journal of Advanced Robotic Systems*, vol. 15, no. 6, p. 1729881418817162, 2018.

[17] L. Guo and Y. Jia, "Inverse model predictive control (impc) based modeling and prediction of human-driven vehicles in mixed traffic," *IEEE Transactions on Intelligent Vehicles*, 2020.

[18] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A Digital Twin paradigm: Vehicle-to-Cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference*, May 2020, pp. 1–6.

[19] OpenWeatherMap. (2021) Weather API. [Online]. Available: https://openweathermap.org/api

[20] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

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

[23] 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.

[24] 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.

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[26] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[27] Z. Wang, K. Han, and P. Tiwari, "Digital Twin Simulation of Connected and Automated Vehicles with the Unity Game Engine," *TechRxiv*, 5 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/Digital_Twin_Simulation_of_Connected_and_Automated_Vehicles_with_the_Unity_Game_Engine/14582943