An Efficient Framework for Reliable And Personalized Motion Planner in Autonomous Driving

Shu Jiang[1†], Zikang Xiong[1§], Weiman Lin[†], Yu Cao[†], Zhongpu Xia[†], Jinghao Miao[†] and Qi Luo[†]

***Abstract*—Choosing optimal parameters for large-scale, safety-critical, real-world motion planners in autonomous driving systems is crucial. In this paper, we present a highly efficient framework for parameter tuning. This framework introduces a drive-like-human idea to a parametric rule-based planner without compromising safety-related constraints. Our framework evaluates an existing planner's performance with a learning-based critic and automatically tunes the parameters with Bayesian optimization. The learning-based critic replaces hand-crafted tuning criteria and evaluates the planner's performance based on human driving experiences. Our framework frees engineers from tedious parameter-tuning work and makes the planner deployment more scalable without losing its reliability. More interestingly, we demonstrate that the proposed framework captures the driving styles of the training datasets. By customizing the training dataset, the critic guides the framework to personalize the planner with different parameter sets.**

I. Introduction

An autonomous vehicle motion planner generates safe, efficient, and smooth driving plans. Rule-based and learning-based methods are two common approaches to build a motion planner. Rule-based approaches rely on optimization [1], [2]. While providing rigorous guarantees, rule-based approaches also require modeling problems with complex constraints and parameters. On the other hand, learning-based approaches have been extensively investigated lately. The introduction of learning components simplifies the complex modeling procedure and boosts the performance of motion planners. For example, [3], [4], [5] learn planners in the end-to-end fashion and provide significant benefits in terms of scalability and computational time. [6], [7] applied imitation learning for challenging partially observable Markov decision process to motion planning problems. [8], [9] leverage reinforcement learning in motion planning. [10] casts the complex vision planning problem to the planning problem in low-dimensional latent space. In the autonomous driving field, [11], [12], [13] avoid the tedious design process of rules and constraints by learning from human demonstrations. However, the outstanding performance comes with the black-box nature of deep learning, the robustness and safety of which is still an on-going research [14]. That raises concerns in safety-critical domains such as autonomous driving [15].

We propose a hybrid approach to introduce learning-enabled components into reliable rule-based planners [16]. The rule-based motion planners are proved to be reliable over years of daily cruising. They formulate motion planning as constrained optimization problems [17], [18], [19], which are interpretable and reliable since safety and physical constraints are well-formulated. However, the performance of a rule-based planner heavily depends on how the parametric optimization objective function is formulated, which requires tremendous tuning effort to tune for best performance.

Automatic tuning frameworks [20], [21] were proposed to speed up the parameter-tuning process. Typically, these frameworks require an objective function as the performance measurement. By optimizing the objective function, these frameworks improve the performance. Some previous works [22], [23], [24] adopted handcrafted objective functions. However, for autonomous driving, it is hard to design such objective functions because of more nuances in terms of the planner performance and the riders' preference [25]. In contrast, we propose a learning-based objective function called *critic*. Driven by human driving data, this critic quantitatively measures the performance of autonomous driving cars in terms of the similarity to human behaviors. [26] also investigated learning a parametric objective function. However, this objective function is directly optimized by a planner instead of an automatic tuning framework. This approach is hard to generalize because different scenarios usually require different planning objective functions. Hence, they have to collect data to learn different objective functions in each scenario. [13] models human-centered threat assessment with deep learning models generated by network architecture search. The human-centered threat assessment is similar to our learning-based critic. Both of them evaluate the behaviors of autonomous driving cars. However, our work further applies the learnt critic in a parameter tuning framework with more scenario variation.

We integrate the learning-based parameter tuning framework with Apollo rule-based planners. As a result, our framework reconciles the convenience of the learning-based approach and the reliability of the rule-based approach. Our main contributions are as follows:

- We present a learning-based critic to measure motion planner performance in autonomous driving, which removes designers' empirical knowledge from the metric selection.
- With this critic, we build an automatic tuning framework for rule-based planners, which achieves optimal performance in high-fidelity simulation scenarios. The framework reduces the tuning process from weeks to less than two hours.
- The proposed framework allows customizing planners with personalized driving styles reflected in the demonstration datasets.

[1] Authors contributed equally to this paper

† Authors affiliate to Apollo Autonomous Driving USA, 1195 Bordeaux Dr, Sunnyvale, CA 94089, USA

§ Author affiliates to Computer Science Department, Purdue University, IN 47906, USA. He worked on this project during his internship at Apollo Autonomous Driving USA.

II. Problem Statement

Definition II.1 (Parameterized deterministic Policy)**.** A parameterized deterministic policy θ is a mapping from the set of environment configuration sequence C to the ego vehicle's configuration sequence $\hat{C}$. The mapping is deterministic when the parameters Φ are fixed.

Given predicted environment configuration sequence $C = [c_1, c_2, \ldots, c_n]$, where n is the planning horizon, the output speed plan is a configuration sequence $\hat{C} = \theta_\Phi^{sp}(C)$, where $\hat{C} = \varnothing$ (i.e., fail to find a feasible solution and a conservative safe plan will be the replacement) or $[\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_n]$. Comparing with predicted configuration c_i, the configuration $\hat{c}_i$ includes additional information about the ego vehicle with the plan produced by θ_Φ^{sp}.

Definition II.2 (Policy Critic)**.** A policy critic, denoted as f_{critic}, measures the quality of a policy by evaluating the action sequence generated by this policy in a given environment.

$f_{\text{critic}}(\theta_\Phi^{sp}, C)$ measures the policy's speed plan w.r.t the predicted configurations C. The policy critic could be rule-based [24] or learning-based. Given the ego vehicle states and its surrounding environment, including path and obstacles, the policy critic measures the speed plans over the whole planning time horizon.

Definition II.3 (Parameterized Policy Tuning)**.** Parameterized policy tuning aims at finding an optimal parameter set for the parameterized policy to achieve the best performance in simulation scenarios. A policy critic measures the performance.

$$\Phi^* = \arg\min_{\Phi} F_{\text{critic}}(\theta_\Phi^{sp}, \mathcal{C}) \tag{1}$$

where the F_{critic} is a composition of f_{critic}, and $\mathcal{C}$ is a set of predicted environment configurations generated in various challenging scenarios.

In a driving scenario, our planner generates a sequence of speed plans. Single speed plan may fail to reflect the planner's performance in the scenario. Hence, it is crucial to evaluate over a range of speed plans. We designed a composition function f_{comp} for this goal. f_{comp} combines the policy critic cost $f_{\text{critic}}(\theta_\Phi^{sp}, \mathcal{C}_i)$, from each speed plan for the final cost. Formally,

$$\Phi^* = \arg\min_{\Phi} f_{\text{comp}}(f_{\text{critic}}(\theta_\Phi^{sp}, \mathcal{C}_0), ..., f_{\text{critic}}(\theta_\Phi^{sp}, \mathcal{C}_N)) \tag{2}$$

where $\mathcal{C}_0$ to $\mathcal{C}_N$ are environment configurations. To efficiently tune the parameterized policy, we have to address two problems:

- Define F_{critic} that can capture the characteristics of a range of speed plans.
- Efficiently optimize on the F_{critic} for an optimal planner parameter set.

For the second problem, we apply Bayesian Optimization (BO) [27], a widely-used black box optimization technique to search for the optimal parameter Φ^* that minimizes F_{critic} in Sec. III-D.

III. Methods

We obtained a learning-based critic that serves as the core of an optimization objective function. This automatic tuning framework efficiently finds the optimal parameter set for a rule-based speed planner.

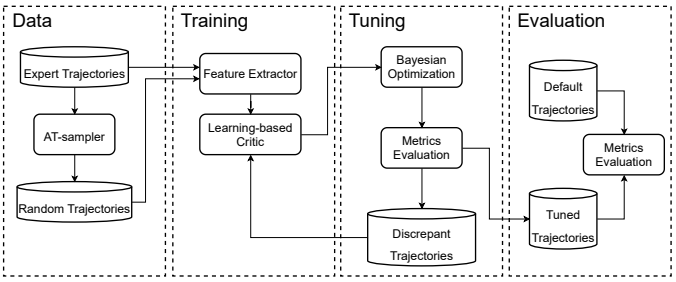


Figure 1. Framework Structure

An overview of our framework is in Fig. 1. It consists of data, training, tuning, and evaluation phases. In the data collection phase, expert trajectories are collected from hired professional drivers as human demonstrations, and random trajectories are generated from an AccelerationTime-sampler (AT-sampler). In the training phase, the learning-based critic is trained with extracted features of trajectories. Afterward, Bayesian optimization is applied to optimize an objective function built from the learning-based critic. As some discrepant results may appear in the tuning phase, we collect them and send them back to the training phase to refine the critic. Finally, the trajectories generated by the tuned planner are evaluated on multiple metrics in the evaluation phase.

A. Feature Extraction

Extracting critical information from the configuration $\hat{c}_i$ is one of the key steps to effectively learn the policy critic f_{critic}. We extract planning-relevant features to avoid overfitting location-related information.

A feature extractor, denoted as $\mathcal{E}$, extracts planning-relevant feature. The planning-relevant feature $\mathcal{E}(\hat{c}_i) \in \mathbb{R}^{54}$ includes speed feature $\texttt{fea}_s \in \mathbb{R}^3$, path feature $\texttt{fea}_p \in \mathbb{R}^3$ and obstacle feature $\texttt{fea}_o \in \mathbb{R}^{48}$. $\texttt{fea}_s$ contains the speed, denoted as v, acceleration, denoted as a, and jerk, denoted as j, of the ego vehicle. $\texttt{fea}_p$ represents path characteristics, including speed limit, denoted as v_{limit}, heading angle α, and curvature κ. $\texttt{fea}_o$ is features of obstacle in left-front, front, right-front, left-rear, rear, and right-rear of the ego vehicle. An 8-dimensional feature characterizes the obstacles in each direction, including obstacle type o_{type}, relative position, speed, acceleration in Frenet Frames [18] and Euclidean distance to ego vehicle, denoted as d_o as demonstrated in Fig. 2.

Consider a speed plan $\hat{C}$, applying the $\mathcal{E}$ to each configuration in $\hat{C}$ generates a feature sequence $[\mathcal{E}(\hat{c}_1), ..., \mathcal{E}(\hat{c}_n)]$. Each feature sequence is also associated with a goal feature $\texttt{fea}_g \in \mathbb{R}^4$ which includes scenario related features, such as cruising or picking-up customers. Thus, we write such feature sequence τ as $\tau = [\mathcal{E}(\hat{c}_1), ..., \mathcal{E}(\hat{c}_n) | \texttt{fea}_g]$ We denote $\tau = \mathcal{E}(\hat{C})$ and $\hat{C} = \theta_\Phi^{sp}(C)$. Hence, $\tau = \mathcal{E}(\theta_\Phi^{sp}(C))$. Since the f_{critic} only exploits the features in τ, in the following

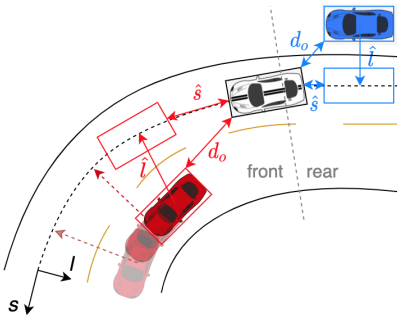
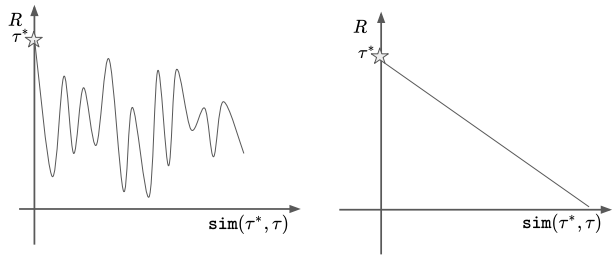


Figure 2. Obstacle feature demonstration. Obstacles are projected to the $s-l$ Frenet frame formed by the planned path. We consider obstacles in left-front, front, right-front, left-rear, rear and right-rear w.r.t the ego vehicle (white car).



(a) Maximize Entropy (b) Similarity Regression

Figure 3. Learned reward function with entropy and similarity measurements

sections, we will simplify the $f_{\text{critic}}(\theta_{\Phi}^{sp}, C)$ defined in Sec. II as $f_{\text{critic}}(\tau)$.

B. Critic Design

We construct the f_{critic} with the inverse reinforcement learning with human driving data and a sample-based planner.

1) Inverse Reinforcement Learning (IRL): IRL's core mission is learning a reward function such that it maximizes the reward of the given demonstration dataset $\mathcal{D}^* \sim \pi^*$, which is sampled from the optimal demonstration π^*. [28] proposed the max-entropy IRL, which simultaneously maximizes the reward of $\tau \in \mathcal{D}^*$ and the entropy of all the possible trajectories. The reward function $R_\varphi(\tau)$ is parameterized by φ. The loss function is

$$\mathcal{L}_{inv} = -\sum_{\tau^* \in \mathcal{D}^*} R_\varphi(\tau^*) + M \log \sum_{\tau \in \mathcal{D}} e^{R_\varphi(\tau)}, \tag{3}$$

where M is the size of $\mathcal{D}^*$. $\mathcal{D}$ is a large dataset that represents all the valid trajectories. The goal is to find a φ^* such that $\varphi^* = \arg\min_\varphi \mathcal{L}_{inv}$. On one hand, minimizing $-\sum_{\tau^* \in \mathcal{D}^*} R_\varphi(\tau^*)$ raises the reward of τ^*. On the other hand, minimizing $M \log \sum_{\tau \in \mathcal{D}} e^{R_\varphi(\tau)}$ increases the entropy of all trajectories' reward.

We further improve the inverse reinforcement approach from two aspects. First, we enhance the loss function with similarity measurement. This loss function leads to an optimization-friendly critic. Second, we generate the large dataset $\mathcal{D}$ with an AT-sampler, which significantly shrinks the data sampling space.

2) Similarity: Since our critic f_{critic} is also a function that characterizes the quality of trajectories, it seems to be quite straightforward to let $f_{\text{critic}} = R_\varphi$. More precisely, because we minimize the *cost* in (1), we can let $f_{\text{critic}} = -R_\varphi$. However, a typical inverse reinforcement learning approach does not consider the *similarity* between trajectories, which is a crucial feature required by our critic.

To illustrate the role of similarity measurement, we provide an example in Fig. 3. The y-axis represents reward, and the x-axis $\texttt{sim}(\tau, \tau^*)$ signifies the similarity to one optimal trajectory τ^*. We define the similarity with the ℓ_1 distance between the (normalized) speed feature $\texttt{fea}_s \in \mathbb{R}^3$.

$$\texttt{sim}(\tau_1, \tau_2) = ||\texttt{fea}_s^1 - \texttt{fea}_s^2|| \tag{4}$$

Although when $\texttt{sim}(\tau, \tau^*) = 0$, the R is maximized in both Fig. 3(a) and Fig. 3(b), the reward function R learned with $\mathcal{L}_{inv}$ in Fig. 3(a) is not as smooth as Fig. 3(b). This is because maximizing entropy requires the entropy between different trajectories as large as possible, which causes a more steeping reward function. Compared with Fig. 3(b), Fig. 3(a) has more local optima, which results in more difficult optimization. An important property we expect is that $R_\varphi(\tau)$ gives a quantitative measurement for the similarity to an optimal demonstration trajectory - we expect a higher reward if one trajectory is more similar to the demonstration. This property tells the optimization algorithm the distance to the optimal solution and thus benefits the optimization algorithm.

3) IRL-inspired Critic Loss Function: With the similarity measurement in mind, our parameterized critic $f_{\text{critic},\varphi}$ is optimized with the following loss function

$$\mathcal{L}_{\text{critic}} = \sum_{\tau^* \in \mathcal{D}^*} f_{\text{critic},\varphi}(\tau^*) \tag{5a}$$

$$+ \sum_{\tau^* \in \mathcal{D}^*} \sum_{\tau \in \mathcal{D}} ||f_{\text{critic},\varphi}(\tau) - \texttt{sim}(\tau, \tau^*)|| \tag{5b}$$

Minimizing (5b) means regression $f_{\text{critic},\varphi}(\tau)$ with $\texttt{sim}(\tau, \tau^*)$. Minimizing (5a) decreases the *cost* of τ^*. To avoid that $f_{\text{critic},\varphi}(\tau^*)$ keeps decreasing, we bound $f_{\text{critic},\varphi}(\tau)$ greater than 0. Compared with max-entropy loss which maximizes the reward and entropy, our loss function minimizes the cost and regression on the similarity.

4) Sampler: Another missing piece yet is how to generate a large dataset $\mathcal{D}$ that represents the whole trajectories. We designed a sampler to generate $\mathcal{D}$ effectively. Given one demonstration trajectory, a sampler generates around 1000 speed profiles by tweaking the acceleration in $\texttt{fea}_s \in \mathbb{R}^3$ of the demonstration trajectory. To avoid generating unrealistic samples and reduce sample space, we infer the speed and jerk feature from the sampled acceleration feature. The generated samples in $\mathcal{D}$ are associated with the demonstration trajectories in $\mathcal{D}^*$. We use this corresponding relationship during the training. Furthermore, we only compute the loss with trajectories with such association, which ensures every trajectory in $\mathcal{D}$ only has one corresponding optimal demonstration.

C. Model Design

We need to design an appropriate critic model for our algorithm and data. One choice is the simple Multi-Layer Perceptron (MLP) model. However, such a fully connected model cannot capture the temporal relationship effectively, which hinders the MLP from extracting information from sequence data like speed plans. Models in the Recurrent Neural Network (RNN) family elegantly handle the sequence data. Thus, they are good candidates for the critic model. Furthermore, inspired by [29], [30], we also apply a GRU-Encoder-Decoder (GRU-ED) model to encode the information extracted from the environment as a priori. While the embedding generated by the GRU-ED contains the demonstration information the critic needs to extract from the environment, it also has fewer dimensions than the raw environment data. Such dimension compression benefits our critic model.

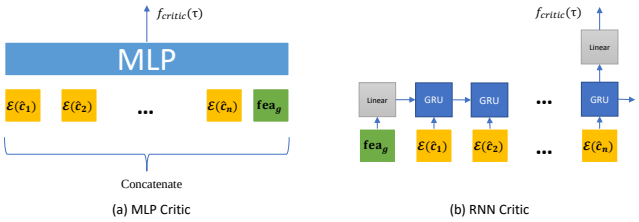


Figure 4. MLP and RNN Critics

1) MLP Critic: The input of MLP critic is a concatenation of a trajectory τ, and it is associated goal features $\mathtt{fea}_g$. The MLP ignores the data's temporal relationship, while the temporal relationship can be crucial for a critic. For example, a critic may focus more on extracted features $\mathcal{E}(\hat{c}_i)$ with smaller i (i.e., features of the nearer future points).

2) RNN Critic: A demonstration of this model is provided in Fig. 4(b) with GRU [31] as RNN cell. A GRU cell has two inputs, the hidden state, and the input state. We feed the τ to an RNN model in sequence and pass the $\mathtt{fea}_g$ to a linear layer, mapping it to the initial hidden state.

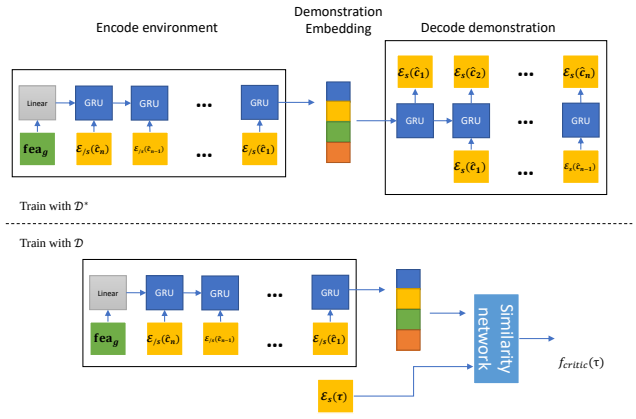


Figure 5. GRU-ED Critic. The input sequence of the encoder is in reversed order, making the demonstration embedding focus on the near future features.

3) GRU-ED Critic: The key idea of the GRU-ED critic is recovering the speed demonstration embedding from the environment. With the embedding compressing the number of dimensions as a priori, we can learn the f_{critic} effectively. In the upper part of Fig. 5, we train an encoder and a decoder together, with the demonstration dataset $\mathcal{D}^*$. The encoder encodes the environment features $\mathcal{E}_{/s}(\hat{c})$ (i.e. all features excepting speed features $\mathtt{fea}_s$) and goal features $\mathtt{fea}_g$ into an embedding. Then, the decoder recovers the speed feature $\mathcal{E}_s(\hat{c})$ from embedding. The $\mathcal{E}_s(\hat{c})$ represents speed features $\mathtt{fea}_s$ of configurations $\hat{c}$. In the lower part of Fig. 5. We apply the pre-trained encoder to generate a demonstration embedding. Then, we feed the demonstration embedding and the speed features to a similarity network. This similarity network predicts a similarity score to a demonstration trajectory. The similarity network can either be MLP or RNN. Although the speed feature data is sequential, because of its small dimensionality ($\mathtt{fea}_s \in \mathbb{R}^3$), when the trajectory length n is small, we choose MLP that performs well in such a low dimension case. Since we want to measure the similarity to a certain demonstration trajectory for any trajectory in a representative dataset $\mathcal{D}$, we train the similarity network with both $\mathcal{D}$ and $\mathcal{D}^*$.

D. Tuning Framework

1) Composition Function: During the planning phase, our speed planner θ_{Φ}^{sp} generates various profiles in the online phase. Hence, it is necessary to evaluate the performance of a parameter Φ over all the speed profiles it generates in different scenarios. One approach to evaluate a planner is computing the average of all the critic costs. However, this approach is less sensitive to extreme values because the low-cost majority hides those high-cost undesired speed profiles. On the contrary, to optimize on the highest critic costs may lead to overlooking the majority performance. The dilemma for the above 2 cases pushes us to propose a maximal k-th composition function that balances the demands from both scenarios,

$$f_{\text{comp}}(f_{\text{critic}}(\tau_1), ..., f_{\text{critic}}(\tau_n)) = \frac{\sum \mathtt{max_k}(f_{\text{critic}}(\tau_1), ..., f_{\text{critic}}(\tau_n))}{k} \tag{6}$$

Here, $\mathtt{max_k}$ returns the greatest k-th elements of all the critic costs. Then, we compute the average of them. Admittedly, we introduce a new hyperparameter k here. However, compared with tuning tones of the planning algorithms, the effort spent on this hyperparameter is negligible.

2) Reliability: The reliability of our framework is defined as three folds: 1) safety, which means no violation of traffic rules and no collisions; 2) feasibility, which means to generate a solution within a given time; 3) completion, which means to complete a desired task such as lane-change or side-pass.

The first fold of safety reliability is naturally guaranteed because safety constraints from the original rule-based planners are preserved. The feasibility and completion regarding the performance reliability are integrated with a variant of the F_{critic}. The feasibility is measured by the fallback rate, which shows percentage of plans without feasible solution, denoted as $r_{\text{fallback}} = n_{\text{fallback}}/n_{\text{total}}$, where the n_{fallback} is the

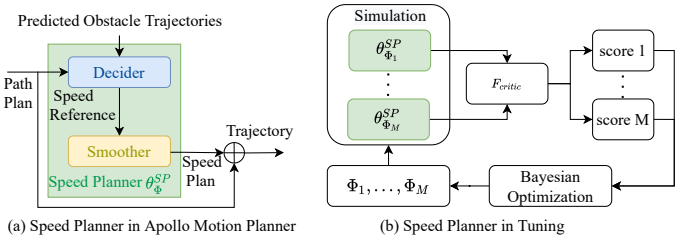
fallback count and n_{totals} is the number of plans. The fallback plan is still a collision-free plan but might include conservative behaviors. The mission completion constant, denoted as $c_{\text{completion}}$, checks whether the planner completes simulation scenarios. A scenario is incomplete if a vehicle fails some metrics, such as reaching a checkpoint within a certain time duration, changing to the target lane, or keeping a minimal distance to an obstacle. $c_{\text{completion}} = 1.0$ when a scenario is not completed, otherwise it is zero. The final objective function is $F_{\text{critic}}(\theta_{\Phi}^{sp}, \mathcal{C}) + r_{\text{fallback}} + c_{\text{completion}}$. Minimizing the r_{fallback} and $c_{\text{completion}}$ provides us with enhanced reliability.

3) Tuning: Finally, we applied the Bayesian optimization [27] to our objective in (1). The Bayesian optimization queries F_{critic} with the most promising parameters Φ in its search space and achieves the highest upper-confidence bound of a Gaussian process regression surrogate model, which approximates the F_{critic}. With this optimization process, parameter tuning is fully automated with high efficiency in parameter tuning efforts. Moreover, the framework is deployed on cloud-based parallel computation workstations, which significantly reduces the total parameter tuning time as presented in Sec. IV.

IV. Experiments and Results

A. Speed Planner and Framework

In this section, we introduce the parametric speed planner [16] and show its parameter tuning process.



(a) Speed Planner in Apollo Motion Planner (b) Speed Planner in Tuning

Figure 6. Proposed framework applied to the Apollo speed planner

The speed planner contains a decider and a smoother as shown in Fig. 6(a). The speed decider generates a rough speed reference by making yield or overtake decisions on surrounding dynamic obstacles. The planning trajectory is discretized as the time-distance graph in Fig. 7(d). The cost_d is defined with the relationship between the speed points and projected decisions in the time-distance graph [20]. Since we only cares about tuning the parameters of the decider, we simplify the cost_d as

$$\text{cost}_d = f_d(\mathbf{w}_d, c_{ob}, c_a, \hat{C}) \tag{7}$$

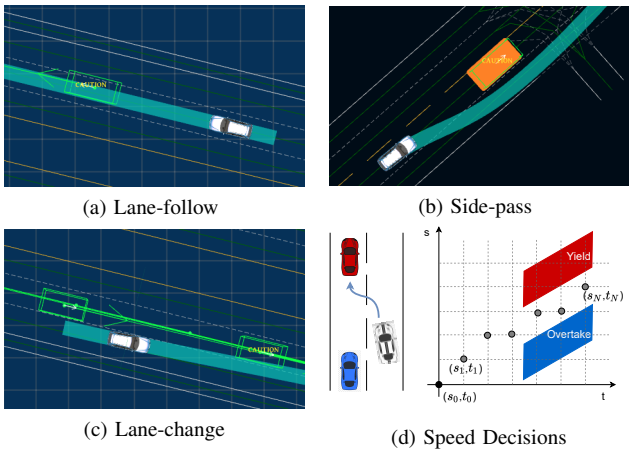
where the $\mathbf{w}_d$ is a vector of weights for different cost terms. The cost terms include the obstacles distance-related costs c_{ob} and acceleration-related costs c_a. These cost terms are computed with speed plan $\hat{C}$. The composition function f_d applies the weights $\mathbf{w}_d$ to different cost terms. An optimization algorithm is utilized to find the rough speed guideline with minimum cost.

The speed smoother generates smoothed speed plans by connecting adjacent sparse speed reference points with a cubic polynomial, while assume that the jerk is constant. The cubic polynomial is generated by solving a constrained nonlinear quadratic optimization [17]. The speed smoother's costs are

$$\text{cost}_s = f_s(\mathbf{w}_s, c_j, c_a, c_{\text{cen}}, c_r, \hat{C}) \tag{8}$$

where $\mathbf{w}_s$ is the costs weight vector. The cost terms include jerk cost c_j, acceleration cost c_a, centripetal cost c_{cen} and reference cost c_r. These costs are computed with speed plan $\hat{C}$. f_s is a composition function applying the weights $\mathbf{w}_s$ to different cost terms.

Fig. 6(b) shows how the proposed framework generates optimal planner. Initially, a batch of parameter set $\theta_{\Phi}^{sp} = \{\mathbf{w}_d, \mathbf{w}_s\}$ is parallelly evaluated with the objective function F_{critic}. The Bayesian optimization module utilizes the evaluating scores to search for better parameters. The framework iteratively runs the above process until converging. Finally, we pick up the parameters set with the lowest F_{critic} score as the best parameter set.



(a) Lane-follow (b) Side-pass (c) Lane-change (d) Speed Decisions

Figure 7. (a) to (c) are the interactive scenarios in Apollo high-fidelity simulation environment; (d) is a time-distance graph in lane-change

B. Critic Model Accuracy

We evaluated four critic models as introduced in Sec. III-C, they are 1) MLP critic; 2) RNN critic; 3) GRU-ED considering the full 3-D speed plan feature, denoted as GRU-ED3 and 4) GRU-ED critic considering the speed and acceleration features only, denoted as GRU-ED2. We design the GRU-ED2 because one can always infer jerk given a sequence of acceleration. The similarity network is trained with the proposed IRL-inspired method, as introduced in Sec. III-C. The critic training results are shown in Table I. The $\mathcal{L}_{\text{exp}}$ is the loss for the expert human-driving trajectories defined as (5a). Small $\mathcal{L}_{\text{exp}}$ means the critic predicts low cost for expert human-driving trajectories. The total critic loss $\mathcal{L}_{\text{critic}}$, is defined as (5). Both GRU-ED2 and GRU-ED3 encoder-decoder has decent evaluation error. GRU-ED3 has the lowest evaluation error.

C. Framework in Simulation

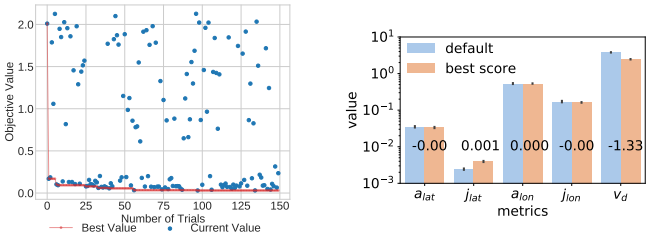
1) Experiment Setup: The framework is validated in lane-follow, side-pass and lane-change scenarios. Fig. 7(a) is a lane-follow scenario, where the ego vehicle (white car) adjusts its

Table I
CRITIC MODEL ACCURACY

Critic Model	Encoder-Decoder			Regression	
	e_v	e_a	e_j	$\mathcal{L}_{\text{exp}}$	$\mathcal{L}_{\text{critic}}$
MLP	n/a	n/a	n/a	0.0555	0.0699
RNN	n/a	n/a	n/a	0.0303	0.1031
GRU-ED3	0.0464	0.0167	0.0098	**0.0164**	**0.04069**
GRU-ED2	0.0463	0.0115	n/a	0.0491	0.06761

speed when the leading obstacle vehicle (green cube) suddenly slows down or stops. Fig. 7(b) is a side-pass scenario. The leading vehicle (green cube marked with orange patch) stopped and blocked the way, and the ego vehicle takes an adjacent lane to bypass the leading vehicle. Fig. 7(c) shows a lane-change scenario, where the ego vehicle merges into the left-hand-side target lane. There is continuous traffic in the target lane.

2) Time Efficiency: The proposed framework aims to find the parameter set of a speed planner with the lowest score defined in III-D2. Fig. 8(a) shows the optimization history of a side-pass scenario. The best value decreases over trials, while the tuning framework keeps exploring new regions.



(a) Tuning history

(b) Tuned planner performance

Figure 8. Tuning with critic Model, GRU3-ED, in lane-following scenario.

Table II
COMPUTATION TIME OF FRAMEWORK WITH DIFFERENT CRITIC MODELS.

N_p	Critic Model	N_{best}	N_{all}	t_{one} (sec)	t_{all} (hours)
5	MLP	126	150	140.88	0.97
	RNN	93	150	129.72	0.90
	GRU-ED3	76	150	140.46	0.97
	GRU-ED2	97	150	114.32	0.79
7	MLP	37	150	241.05	1.67
	RNN	115	150	248.309	1.72
	GRU-ED3	130	150	275.21	1.91
	GRU-ED2	21	150	244.48	1.69
5 or 7	HandTuned	n/a	n/a	n/a	~ 40.0

Table II shows the time efficiency of the tuning framework with different critics. The number of planner parameters is denoted as N_p. In each trial, the framework takes t_{one} time to perform simulation and optimization (around 95% of t_{one}), feature extraction from simulation result (around 10 sec) and critic score generation (less than 1 sec). More parameters need more time for each trial, because the Bayesian optimization requires more time as the number of parameters increases. The trial number N_{best} denotes the iteration discovering the optimal parameters. The total trial number is N_{all}. The total time cost t_{all} is the time to finish all trials. Note that since our framework is parallelized, there is $t_{all} \ll t_{one} \times N_{all}$. Currently, we are using six computing nodes (i.e., six paralleled processes). t_{all} could be further reduced with more computation resource. The tuning environment is a cloud cluster with a 130-core CPU clocked at 2.6 GHz. Compared to manual tuning, which is around 40 hours (8 hours by 5 days), the efficiency has improved over 95%. Note that, based on our experiences, the geographic region difference is non-trivial to the deployment of planners. That means the motion planner needs to be constantly re-tuned when being deployed to different cities. The tuning time and engineering efforts are significantly reduced by the proposed framework for large-scale deployment.

3) Performance: The performance of the tuned speed planner is compared in five dimensions as shown in Table III to V. The performance columns show the mean value of speed plans. The comfort related metrics are longitudinal acceleration and lateral acceleration, denoted as a_{lon} and a_{lat}, respectively; lateral jerk and longitudinal jerk are denoted as j_{lon} and j_{lat}, respectively. The efficiency metric is the difference between the average speed and road speed limits, denoted as v_d. For all these metrics, the smaller values are better. Note that even if we did not explicitly include those metrics in the objective function of Bayesian optimization, the speed planner from our proposed framework achieves better performance in terms of comfort and efficiency. Fig. 8(b) shows that the efficiency is improved as the v_d decreases 1.33, with limit compromise in comfort evaluated by other metrics. The number on each pair of bars is the difference between speed plans from default and tuned speed planners.

Since speed decisions on obstacles are fixed in speed decider, we start from tuning five parameters of the speed smoother in (8) for lane-follow and side-pass scenarios. In lane-change scenarios, the ego vehicle can overtake a vehicle in the target lane and merge by cutting in; or yield to the obstacle and merge by following. A less optimal speed plan may result in the failure of the change-lane decision [32] [33]. Thus, two extra decision related parameters from (7) are included in lane-change scenarios.

Table III
TUNED SPEED PLANNER PERFORMANCE IN LANE-FOLLOW SCENARIOS

Critic Model	Critic Score	Performance				
		a_{lon}	j_{lon}	a_{lat}	j_{lat}	v_d
MLP	0.0289	0.5574	0.2199	0.0357	0.0041	2.5765
RNN	0.0348	0.6924	0.7107	0.1718	0.0353	3.0294
GRU-ED3	0.0057	0.5336	0.1627	0.0335	0.0040	2.4622
GRU-ED2	0.0063	0.5397	0.1237	0.0354	0.0039	2.6181
HandTuned	n/a	0.5466	0.2298	0.0364	0.0036	2.9962

Table IV
TUNED SPEED PLANNER PERFORMANCE IN SIDE-PASS SCENARIOS

Critic Model	Critic Score	Performance				
		a_{lon}	j_{lon}	a_{lat}	j_{lat}	v_d
MLP	0.2368	1.0440	0.9468	0.0611	0.0272	6.4267
RNN	0.2336	0.6691	0.4596	0.0608	0.0224	6.4963
GRU-ED3	0.0280	0.6795	0.5127	0.0497	0.0219	6.9751
GRU-ED2	0.0304	0.5459	0.4057	0.0554	0.0246	6.9127
HandTuned	n/a	0.7747	0.4276	0.0506	0.0259	7.0588

In side-pass scenarios, the difference to speed limit v_d is higher because the ego vehicle pauses for a certain time to

Table V
TUNED SPEED PLANNER PERFORMANCE IN LANE-CHANGE SCENARIOS

Critic Model	Critic Score	Performance				
		a_{lon}	j_{lon}	a_{lat}	j_{lat}	v_d
MLP	0.1329	0.6548	0.7584	0.2005	0.0341	3.2399
RNN	0.0026	0.8386	0.3434	0.1392	0.0293	4.1935
GRU-ED3	0.0044	0.7947	0.3592	0.0439	0.0101	2.7693
GRU-ED2	0.0003	0.2668	0.6776	0.2253	0.0254	4.8979
HandTuned	n/a	0.7697	0.3899	0.1062	0.0289	3.5655

make sure the leading vehicle is indeed a blocker before taking the action of side-pass. In conclusion, compared to the hand-tuned speed planner, speed planners generated from the proposed framework, have achieved equal or better performance in terms of comfort and efficiency of speed plans.

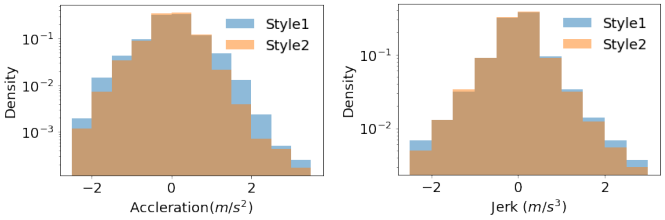
4) Reliability: As mentioned in Sec. III-D2, the safety is guaranteed by the rule-based planner with collision-avoidance constraints. The planners generated from the proposed framework inherits the safety related constraints. Thus, the safety is not compromised. For challenging scenarios, when a rule-based optimizer fails to find a feasible solution within a certain time, a back-up planner generates fall-back trajectories to prepare the vehicle to stop safely. There might be some harsh brakes induced by fall-back trajectories. Thus, it is better to avoid the fall-back trajectories. With the proposed framework, the fall-back ratio is reduced by over 60% compared to hand-tuned parameters, and the total fall-back ratio is reduced to less than 1.0%, as shown in Table VI. The incompleteness ratio of the challenging interaction scenarios in simulation is reduced from 11.76% to 0.0%.

Table VI
FALLBACK RATIO COMPARISON

Scenario	HandTuned	FrameworkTuned	Reduced
Lane-follow	4.37%	1.42%	67.51%
Side-pass	1.94%	0.75%	61.44%
Lane-change	5.03%	0.73%	66.56%

D. Personalized Driving Style

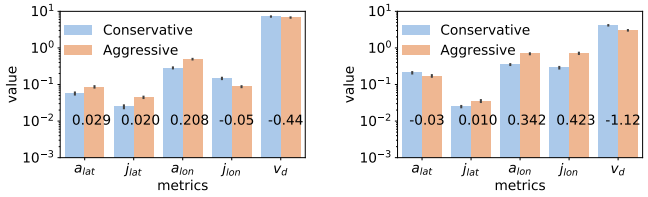
To find the parameters with different driving styles, we categorize human-driving into two styles: Style 1 is an efficiency-oriented driving style, which rewards more on desired speed and punishes less on high acceleration or jerks. This driving pattern might be less comfortable and can also be called as *sporty/aggressive* driving style. In contrast, Style 2 is a comfort-oriented driving style, which rewards more on low acceleration and punishes less if total travel time is long. This driving pattern can be called as *leisure/conservative* driving style. Among different driving style datasets, we compared the longitudinal acceleration and jerk distributions, considering they are among the most common features for categorizing driving style [34], [35]. Fig. 9 shows that there is more acceleration/deceleration in Style 1 dataset compared to Style 2 dataset. Similarly, the jerk distribution in Style 1 dataset is flatter than those of Style 2 dataset, which indicates there are frequent acceleration changes. Both driving styles are extracted from professional drivers in this paper. Their driving behaviors are collision-risk free and maintain courtesy to surrounding vehicles. Thus, we do not consider the relative speed to neighbors and distance to the front car [36] to categorize driving style. Note that human driving styles can be subdivided when more features are considered [35], which can be applied to critic design in future work.



(a) Acceleration Distributions (b) Jerk Distributions

Figure 9. Dataset comparison between different driving styles.

We trained planning critics on both datasets separately and used the same tuning framework with different critics to tune the speed planner in side-pass and lane-change scenarios. Fig. 10 shows the planner tuned with an aggressive critic generates speed plans that are closer to the speed limit, while the comfort-related measurements such as acceleration and jerk are much higher compared to speed plans generated by the conservative-critic-tuned speed planner. The results show that the learning-based critic naturally captures the driving style. With our tuning framework, the rule-based planner also reflects the corresponding driving style.



(a) Side-pass (b) Lane-change

Figure 10. Speed plan comparison.The y-axis is in log scale. The number on each pair of bars is the difference between aggressive style (Style 1) value and conservative style (Style 2).

V. CONCLUSION

Building a comprehensive and one-size-fits-all metric for a motion planner is challenging since the driving/riding experience depends on personal preferences. We presented an IRL-inspired learning-based critic as a customizable motion planner metric to solve the challenging problem. The critic extracted a latent space embedding of a human driving plan based on environment and measured the similarity between motion-planner-generated plan and human-driver plan. Utilizing this critic, we further built a framework to automatically guide rule-based speed planners to make human-like driving plans by choosing an optimal parameter set. This framework removed the human effort in tedious parameter tuning, reduced parameter tuning time, and made the planning module deployment more scalable. Furthermore, the physical and safety constraints in rule-based planners were preserved, which maintained its

reliability. The framework with four different critic models is compared over three interactive driving scenarios. The tuned planners achieved better efficiency without sacrificing comfortable riding experience. Furthermore, by training with different human driving datasets, the critic was able to extract different driving styles. These driving styles were further reflected in the rule-based speed planner tuned by the proposed framework. As a result, a personalized planner is achieved. In the future, we will extend this framework from speed planner to the complete motion planner, to deal with various scenarios with more complicated environment components.

ACKNOWLEDGMENT We would like to thank our colleague, Wu, Szuhao, for his help in conducting experiments.

REFERENCES

[1] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[2] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[3] R. Strudel, R. Garcia, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," *arXiv preprint arXiv:2008.11174*, 2020.

[4] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6582–6588.

[5] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[6] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-driven planning via imitation learning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[7] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation," in *Conference on Robot Learning*. PMLR, 2017, pp. 271–280.

[8] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[9] A. Francis, A. Faust, H.-T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with prm-rl," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1115–1134, 2020.

[10] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2555–2565.

[11] J. Zhou, R. Wang, X. Liu, Y. Jiang, S. Jiang, J. Tao, J. Miao, and S. Song, "Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization," *arXiv preprint arXiv:2103.01882*, 2021.

[12] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[13] D. Shin, H.-g. Kim, K.-m. Park, and K. Yi, "Development of deep learning based human-centered threat assessment for application to automated driving vehicle," *Applied Sciences*, vol. 10, no. 1, p. 253, 2020.

[14] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3578–3586.

[15] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.

[16] "Baidu Apollo open platform," http://apollo.auto/.

[17] Y. Zhang, H. Sun, J. Zhou, J. Hu, and J. Miao, "Optimal trajectory generation for autonomous vehicles under centripetal acceleration constraints for in-lane driving scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3619–3626.

[18] Y. Zhang, H. Sun, J. Zhou, J. Pan, J. Hu, and J. Miao, "Optimal vehicle path planning using quadratic optimization for baidu apollo open platform," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 978–984.

[19] R. He, J. Zhou, S. Jiang, Y. Wang, J. Tao, S. Song, J. Hu, J. Miao, and Q. Luo, "Tdr-obca: A reliable planner for autonomous driving in free-space environment," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 2927–2934.

[20] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2020.

[21] R. Burger, M. Bharatheesha, M. van Eert, and R. Babuška, "Automated tuning and configuration of path planning algorithms," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4371–4376.

[22] J. Cano, Y. Yang, B. Bodin, V. Nagarajan, and M. O'Boyle, "Automatic parameter tuning of motion planning algorithms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8103–8109.

[23] M. Moll, C. Chamzas, Z. Kingston, and L. E. Kavraki, "Hyperplan: A framework for motion planning algorithm selection and parameter optimization," *IEEE Robot. Autom. Letters*, 2021.

[24] Y. Wang, S. Jiang, W. Lin, Y. Cao, L. Lin, J. Hu, J. Miao, and Q. Luo, "A learning-based automatic parameters tuning framework for autonomous vehicle control in large scale system deployment," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 2919–2926.

[25] R. Chandra, U. Bhattacharya, T. Mittal, X. Li, A. Bera, and D. Manocha, "Graphrqi: Classifying driver behaviors using graph spectrums," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4350–4357.

[26] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "Appld: Adaptive planner parameter learning from demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4541–4547, 2020.

[27] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[28] 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.

[29] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," 2020.

[30] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," 2020.

[31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[32] A. Lombard, F. Perronnet, A. Abbas-Turki, and A. El Moudni, "On the cooperative automatic lane change: Speed synchronization and automatic "courtesy"," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1655–1658.

[33] J. Kim, J.-H. Park, and K.-Y. Jhang, "Decoupled longitudinal and lateral vehicle control based autonomous lane change system adaptable to driving surroundings," *IEEE Access*, vol. 9, pp. 4315–4334, 2020.

[34] N. M. Yusof, J. Karjanto, J. Terken, F. Delbressine, M. Z. Hassan, and M. Rauterberg, "The exploration of autonomous vehicle driving styles: preferred longitudinal, lateral, and vertical accelerations," in *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*, 2016, pp. 245–252.

[35] E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha, "Identifying driver behaviors using trajectory features for vehicle navigation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3445–3452.

[36] C. Basu, Q. Yang, D. Hungerman, M. Sinahal, and A. D. Draqan, "Do you want your autonomous car to drive like you?" in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE, 2017, pp. 417–425.