Decision-Making for Autonomous Vehicles With Interaction-Aware Behavioral Prediction and Social-Attention Neural Network

Xiao Li, Kaiwen Liu, H. Eric Tseng, Anouck Girard, *Senior Member, IEEE*, and Ilya Kolmanovsky, *Fellow, IEEE*

***Abstract*— Autonomous vehicles need to accomplish their tasks while interacting with human drivers in traffic. It is thus crucial to equip autonomous vehicles with artificial reasoning to better comprehend the intentions of the surrounding traffic, thereby facilitating the accomplishments of the tasks. In this work, we propose a behavioral model that encodes drivers' interacting intentions into latent social-psychological parameters. Leveraging a Bayesian filter, we develop a receding-horizon optimization-based controller for autonomous vehicle decision-making which accounts for the uncertainties in the interacting drivers' intentions. For online deployment, we design a neural network architecture based on the attention mechanism which imitates the behavioral model with online estimated parameter priors. We also propose a decision tree search algorithm to solve the decision-making problem online. The proposed behavioral model is then evaluated in terms of its capabilities for real-world trajectory prediction. We further conduct extensive evaluations of the proposed decision-making module, in forced highway merging scenarios, using both simulated environments and real-world traffic datasets. The results demonstrate that our algorithms can complete the forced merging tasks in various traffic conditions while ensuring driving safety.**

***Index Terms*— Autonomous vehicles, imitation learning, interaction-aware driving, neural networks, traffic modeling.**

I. Introduction

ONE of the challenges in autonomous driving is interpreting the driving intentions of other human drivers. The communication between on-road participants is typically nonverbal, and relies heavily on turn/brake signals, postures, eye contact, and subsequent behaviors. In uncontrolled traffic scenarios, e.g., roundabouts [1], unsignalized intersections [2], and highway ramps [3], drivers need to negotiate their order of proceeding. Fig. 1 illustrates a forced merging scenario at a highway entrance, where the ego vehicle in red attempts to merge into the highway before the end of the ramp. This merging action affects the vehicle behind in the lane being merged, and different social traits of its driver can result in different responses to the merging intent. A cooperative driver may choose a lane change to promote the merging process, while an egoistic driver may maintain a constant speed and disregard the merging vehicle. Therefore, understanding the latent intentions of other drivers can help the ego vehicle resolve conflicts and accomplish its task.

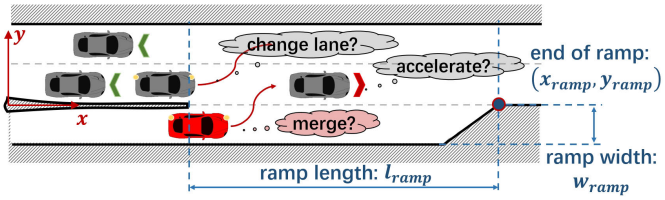

Fig. 1. Schematic of the highway forced merging scenario. An on-ramp ego vehicle in red is merging onto the highway while interacting with the highway vehicles in gray.

In this article, we specifically focus on the highway forced merging scenario illustrated in Fig. 1 and the objective of transitioning the ego vehicle onto the highway from the ramp in a timely and safe manner. The difficulty of developing suitable automated driving algorithms for such scenarios is exacerbated by the fact that stopping on the ramp in noncongested highway traffic could be dangerous.

The forced merging has been addressed in the automated driving literature from multiple directions. In particular, learning-based methods have been investigated to synthesize controllers for such interactive scenarios. End-to-end planning methods [4] have been proposed to generate control inputs from Lidar point clouds [5] and RGB images [6]. Reinforcement learning (RL) algorithms have also been considered to learn end-to-end driving policies [7]. A comprehensive survey of RL methods for autonomous driving applications is presented in [8]. Meanwhile, imitation learning-based methods have been exploited to emulate expert driving behaviors [9]. However, the end-to-end learning-based controllers lack interpretability and are limited in providing safety guarantees in unseen situations. To address these concerns, researchers have explored the integration of learning-based methods with planning and control techniques. Along these lines, model predictive control (MPC) algorithms [10] and tree-structured planners [11] have been integrated with machine-learning models for trajectory prediction and planning. Meanwhile, inverse RL methods have also been explored to predict

Received 30 October 2023; revised 7 July 2024; accepted 26 August 2024. Date of publication 3 October 2024; date of current version 26 June 2025. This work was supported by the University of Michigan/Ford Motor Company Alliance Program. Recommended by Associate Editor Mario Zanon. *(Corresponding author: Xiao Li.)*

Xiao Li, Kaiwen Liu, Anouck Girard, and Ilya Kolmanovsky are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: hsiaoli@umich.edu; kwliu@umich.edu; anouck@umich.edu; ilya@umich.edu).

H. Eric Tseng is with the Electrical Engineering Department, the University of Texas at Arlington, TX 76010 USA (e-mail: hongtei.tseng@uta.edu).

Digital Object Identifier 10.1109/TCST.2024.3460650

drivers' behavior for planning purposes [12], [13]. However, the learning-based modules in these systems may have limited capability to generalize and transfer to unobserved scenarios or behaviors.

There also exists extensive literature on modeling the interactive and reactive behaviors between drivers using model-based approaches. Scenario tree-based method, such as branch MPC [14], [15], have been explored. Assuming drivers maximize their rewards [12], game-theoretic approaches have been proposed to model traffic interactions, such as level-k hierarchical reasoning framework [16], potential games [17], and Stackelberg games [13]. In the level-k game-theoretic models, approaches to estimating drivers' reasoning levels have been proposed [18]. A novel leader–follower game-theoretic controller (LFGC) has been developed for decision-making in forced merging scenarios [19]. However, solving game-theoretic problems could be computationally demanding and has limited scalability to a larger number of interacting drivers or longer prediction horizons. To be able to account for the uncertainty in the interactions, probabilistic methods, leveraging either Bayesian filter [19] or particle filter [20] with partially observable Markov decision process (POMDP) [19], have also been implemented to encode and estimate the uncertain intent of other drivers as hidden variables.

In this article, we consider the social value orientation (SVO) from social psychology studies [21] to model drivers' interactions. The SVO quantifies subjects' tendencies toward social cooperation [22] and has been previously used to model drivers' cooperativeness during autonomous vehicle decision-making in [13] and [23]. In addition, researchers have combined SVO-based rewards with RL to generate pro-social autonomous driving behaviors [24], [25] or synthesize realistic traffic simulation with SVO agents [26]. In our proposed behavioral model, we consider both social cooperativeness and the personal objectives of the interacting drivers. Leveraging a Bayesian filter, we propose a decision-making module that accounts for pairwise interactions with other drivers, and computes a reference trajectory for the ego vehicle under the uncertain cooperative intent of other drivers. The method proposed in this article differs from the previous work [27] in the following aspects: 1) instead of using an action space with a few coarse action primitives, we synthesize a state-dependent set of smooth and realistic trajectories as our action space; 2) we design a social-attention neural network (SANN) architecture to imitate the behavioral model that structurally incorporates the model-based priors and can be transferred to various traffic conditions; 3) we develop a decision-tree search algorithm for the ego vehicle's decision-making, which guarantees safety and scalability; and 4) we conduct an extensive evaluation of the behavioral model in predicting real-world trajectory and demonstrate the decision-making module capabilities in forced merging scenarios on both simulations and real-world datasets, which is not done in [27].

The proposed algorithm has several distinguished features.

1) The behavioral model incorporates both the driver's social cooperativeness and personal driving objectives, which produces rich and interpretable behaviors.

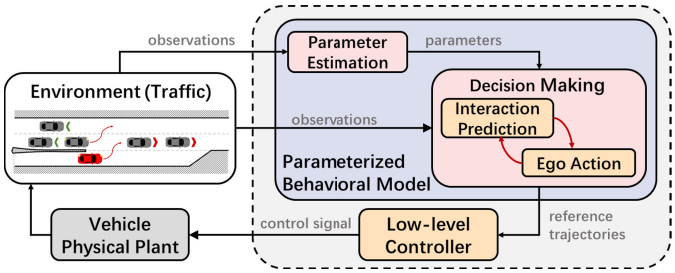

Fig. 2. Proposed algorithm architecture for autonomous vehicles decision–making and control.

2) The proposed decision-making module handles the uncertainties in the driving intent using a Bayesian filter and generates smooth and realistic reference trajectories for the downstream low-level vehicle controller.
3) Differing from pure learning-based methods, the designed SANN incorporates social-psychological model-based priors. It imitates the behavioral model and is transferable across different traffic conditions while providing better online computation efficiency.
4) The decision-making module utilizes an interaction-guided decision tree search algorithm, which ensures probabilistic safety and scales linearly with the number of interacting drivers and prediction horizons.
5) The behavioral model is evaluated in predicting real-world trajectories. This model demonstrates good quantitative accuracy in short-term prediction and provides qualitative long-term behavioral prediction.
6) The decision-making module is tested in the forced merging scenarios on a comprehensive set of environments without retuning the model hyperparameters. The proposed method can safely merge the ego vehicle into the real-world traffic dataset [28] faster than the human drivers, and into diverse Carla [29] simulated traffic with different traffic conditions.

This article is organized as follows: In Section II, we describe the model preliminaries, including the vehicle kinematics model, the action space with the lane-change modeling, and the choice of model hyperparameters. In Section III, we present the behavioral model, which can be utilized to predict interacting drivers' trajectories given their latent driving intentions. In Section IV, we discuss the SANN architecture that imitates the behavioral model and is suitable for online deployment. In Section V, we introduce the decision-making module of our ego vehicle together with a decision tree search algorithm that incorporates the SANN and improves computation efficiency. In Section VI, we report the results of real-world trajectory prediction using the behavioral model for the forced merging test on a real-world dataset and in simulations. Finally, conclusions are given in Section VII.

II. System and Model Preliminaries

In this article, we design a modularized algorithm architecture for decision-making and control of the autonomous (ego) vehicle in the forced merging scenario. In this framework (see Fig. 2), we develop a parameterized behavioral model for modeling the behavior of interacting drivers. Leveraging this model and observed traffic interactions, we can estimate

the latent driving intentions of interacting drivers as model parameters. Thereby, we can predict their future trajectories in response to the action of the ego vehicle. Based on the observations, and the predictions, a high-level decision-making module optimizes a reference trajectory for the ego vehicle merging into the target highway lane while ensuring its safety in the traffic. A low-level controller controls the vehicle throttle and steering angle to track the reference trajectory.

This chapter first introduces the vehicle kinematics model in Section II-A. Then, we present a state-dependent action space (in Section II-B) of the vehicle, which is a set of trajectories synthesized from the kinematics model. We discuss the detailed lane change trajectory modeling in Section II-C together with model hyperparameter identification from a naturalistic dataset [28].

A. Vehicle Kinematics

We use the following continuous-time bicycle model [30] to represent the vehicle kinematics:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} v\cos(\varphi+\beta)\\ v\sin(\varphi+\beta)\\ \dfrac{v}{l_r}\sin(\beta)\\ a\end{bmatrix} + \tilde{w}$$
$$\beta = \arctan\left(\frac{l_r}{l_r+l_f}\tan\delta\right) \tag{1}$$

where x, v, and a are the longitudinal position, velocity, and acceleration of the vehicle center of gravity (CoG), respectively; y is the lateral position of the CoG; φ is the heading angle of the vehicle; β is the sideslip angle; δ is the front wheel steering angle; l_r and l_f denote the distance from the vehicle CoG to the front and rear wheel axles, respectively; $\tilde{w} \in \mathbb{R}^4$ is a disturbance representing unmodeled dynamics.

We assume that all the highway vehicles, together with the ego vehicle, follow this kinematics model. We then derive discrete-time kinematics from (1) assuming zero-order hold with the sampling period of ΔT sec. This leads to the discrete-time kinematics model

$$s^i_{k+1} = f\left(s^i_k, u^i_k\right) + \tilde{w}^i_k, \quad i = 0, 1, 2, \ldots \tag{2}$$

where the subscript k denotes the discrete time instance $t_k = k\Delta T$ sec; the superscript i designates a specific vehicle, where we use $i = 0$ to label the ego vehicle and $i = 1, 2, \ldots$ for other interacting vehicles; $s^i_k = [x^i_k, y^i_k, \varphi^i_k, v^i_k]^{\mathrm{T}}$ and $u^i_k = [a^i_k, \delta^i_k]^{\mathrm{T}}$ are the state and control vectors of vehicle i at time instance t_k. Then, we can use this discrete kinematics model to synthesize vehicle trajectories. Note that there are other vehicle kinematics and dynamics models that could potentially represent vehicle behaviors [30] more realistically. The model (2) was chosen as it provides adequate accuracy for the purpose of decision-making (planning) of the ego vehicle while it is simple and computationally efficient [31].

B. Trajectory Set as Action Space

Given the initial vehicle state and the kinematics model (2) neglecting the disturbance $\tilde{w}^i_k$ and using different control signal profiles, we can synthesize various trajectories with duration $N\Delta T$ sec for trajectory prediction and planning. For the ith vehicle at time t_k, we assume that vehicle's action space is $\Gamma(s^i_k) = \left\{\gamma^{(m)}(s^i_k)\right\}_{m=1}^{M}$ where each individual element $\gamma^{(m)}(s^i_k) = \left\{s^i_n\right\}_{n=k}^{k+N+1}$ is a trajectory of time duration $N\Delta T$ sec synthesized using a distinct control sequence $\left\{u^i_n\right\}_{n=k,\ldots,k+N}$ via the kinematics model (2), and M is the number of trajectories in $\Gamma(s^i_k)$. Here, we neglect the superscript m in u^i_n for simplicity. We select 225 different control sequences such that the number of considered trajectories is finite, i.e., $M \leq 225$ for all $\Gamma(s^i_k)$ and all initial state s^i_k. As shown in Fig. 3, the action space $\Gamma(s^i_k)$ depends on the current vehicle state s^i_k for two reasons: With a fixed control sequence $\left\{u^i_n\right\}_n$, the resulted trajectory from (2) varies with the initial condition s^i_k; A safety filter is implemented for $\Gamma(s^i_k)$ such that all trajectories intersect with the road boundaries are removed, which is also dependent on s^i_k. The chosen 225 control sequences generate trajectories that represent a set of plausible driving behaviors (see Fig. 3); we believe that this set is sufficiently large for the tasks of trajectory prediction and planning. Note that the trajectory set can be further enlarged with more diverse control sequences if necessary.

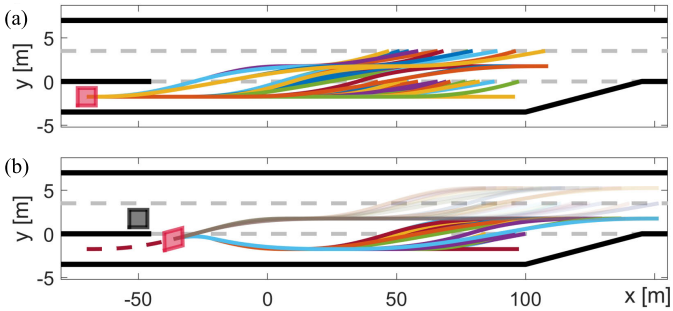

Fig. 3. Examples of trajectory set $\Gamma(s^i_k)$ with duration 6 s. (a) Trajectory set of $M = 109$ encompasses behaviors of lane keeping, lane change, and coupled longitudinal and lateral behavior (e.g., lane change with longitudinal acceleration/deceleration). (b) Trajectory set of $M = 129$ contains actions of lane change abortion and remerge after aborting the previous lane change. Other normal lane change trajectories are in semi-transparent lines. Likewise, the lane change abortion behaviors are also coupled with various longitudinal acceleration/deceleration profiles.

Meanwhile, we assume a complete lane change takes $T_{\text{lane}} = N_{\text{lane}}\Delta T$ sec to move w_{lane} meters from the center line of the current lane to that of adjacent lanes. We set $N_{\text{lane}} < N$ to allow trajectory sets to contain complete lane change trajectories. As shown in Fig. 3(a), the trajectory set comprises 109 regular driving trajectories for an on-ramp vehicle that is intended to merge. This trajectory set considers varieties of driver's actions such as lane keeping with longitudinal accelerations/decelerations, merging with constant longitudinal speed, merging with longitudinal acceleration/deceleration, and accelerating/decelerating before or after merging. Moreover, we also include the behavior of aborting the lane change [see Fig. 3(b)]. For a lane-changing vehicle, this is a regular "change-of-mind" behavior to avoid collision with nearby highway vehicles. For longitudinal behaviors, we also assume speed and acceleration/deceleration limits of $[v_{\min}, v_{\max}]$ and $[a_{\min}, a_{\max}]$ for all vehicle at all times. Namely, the trajectory sets and control sequences satisfy $a^i_n \in [a_{\min}, a_{\max}]$ and $v^i_n \in [v_{\min}, v_{\max}]$ for all $s^i_n \in \gamma^{(m)}(s^i_k)$, $n = k, \ldots, k+N+1$ and for

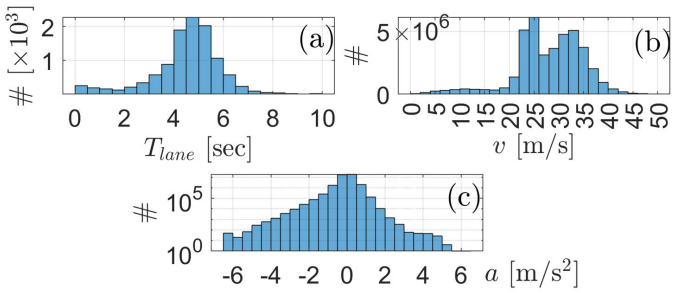

Fig. 4. Histogram of vehicle driving statistics in the high-D dataset [28]. (a) Time duration for a complete lane change. (b) Longitudinal velocity. (c) Longitudinal acceleration/deceleration (y-axis in log scale).

all $\gamma^{(m)}(s_k^i) \in \Gamma(s_k^i), m = 1, \ldots, M(s_k^i)$. The speed limits are commonly known quantities on highways and the longitudinal acceleration/deceleration is typically limited by the vehicle's performance limits.

C. Trajectory Hyperparameters and Lane Change Behavior

We use a naturalistic highway driving high-D [28] dataset to identify the model hyperparameters, i.e., $v_{\min}$, $v_{\max}$, $a_{\min}$, $a_{\max}$, w_{lane}, and T_{lane}. The statistics visualized in Fig. 4 are obtained from data of 110 500 vehicles driven over 44 500 km. The minimum speed is set to $v_{\min} = 2$ m/s since most of the vehicles have speeds higher than that and the maximum speed limit of the dataset is $v_{\max} = 34$ m/s. The majority of longitudinal accelerations and decelerations of high-D vehicles are within the range of $[a_{\min}, a_{\max}] = [-6, 6]$ m/s^2. The lane width $w_{\text{lane}} = 3.5$ m as in high-D dataset. We select $T_{\text{lane}} = N_{\text{lane}}\Delta T = 4$ s since most of the high-D vehicles take between 4 and 6 s to change lanes. We keep these hyperparameters fixed for the following discussion and experiments. Note that these parameters can be identified similar to different values in other scenarios if necessary.

In terms of lane change behaviors, given an acceleration sequence $\{a_k^i\}_k$, we can derive the steering profile $\{\delta_k^i\}_k$ of a lane change trajectory from fifth-order polynomials [32]

$$\begin{aligned} x(t|\{p_j\}) &= p_0 + p_1 t + p_2 t^2 + p_3 t^3 + p_4 t^4 + p_5 t^5 \\ y(t|\{q_j\}) &= q_0 + q_1 t + q_2 t^2 + q_3 t^3 + q_4 t^4 + q_5 t^5 \end{aligned} \tag{3}$$

which represents a vehicle lane change between time $t = 0$ and $t = T_{\text{lane}}$ sec. Such lane change trajectories are commonly used in vehicle trajectory planning and control [33], [34].

Suppose, without loss of generality, that the lane change starts and ends at $t_0 = 0$ and $t_{N_{\text{lane}}} = T_{\text{lane}}$ sec, respectively. The following procedure is utilized to determine the trajectory and steering profile during a lane change: at time $t_k = k\Delta T$ sec, given vehicle state $s_k^i = [x_k^i, y_k^i, \varphi_k^i, v_k^i]^{\mathrm{T}}$ and lateral target lane center y_{target}, we first solve for the coefficients $\{p_{k,j}\}$ and $\{q_{k,j}\}$ in (3) from the following two sets of boundary conditions:

$$\begin{cases} x(t_k) = x_k^i, \quad \dot{x}(t_k) = v_k^i, \quad \ddot{x}(t_k) = a_k^i \\ y(t_k) = y_k^i, \quad \dot{y}(t_k) = \dot{y}_k^i, \quad \ddot{y}(t_k) = \ddot{y}_k^i \end{cases}$$

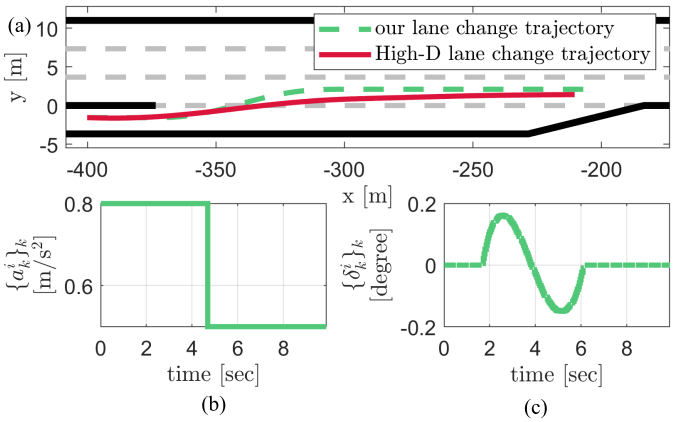

Fig. 5. Lane change trajectory synthesized using a given acceleration sequence. (a) Synthesized trajectory using our algorithm compared with a real-world lane change trajectory in the high-D dataset. (b) Designed acceleration sequence $\{a_k^i\}_k$. (c) Derived steering sequence $\{\delta_k^i\}_k$ from (2).

$$\begin{cases} x(T_{\text{lane}}) = x_k^i + v_k^i(T_{\text{lane}} - t_k) \\ \qquad\qquad + \dfrac{1}{2}a_k^i(T_{\text{lane}} - t_k)^2 \\ \dot{x}(T_{\text{lane}}) = v_k^i + a_k^i(T_{\text{lane}} - t_k)\ \ddot{x}(T_{\text{lane}}) = a_k^i \\ y(T_{\text{lane}}) = y_{\text{target}},\ \dot{y}(T_{\text{lane}}) = 0,\ \ddot{y}(T_{\text{lane}}) = 0 \end{cases} \tag{4}$$

where we assume initial/terminal conditions $\dot{y}(0) = \dot{y}(T_{\text{lane}}) = 0$ and $\ddot{y}(0) = \ddot{y}(T_{\text{lane}}) = 0$, i.e., zero lateral velocity and acceleration at the beginning and the end of a lane change. Recursively, initial conditions $\dot{y}(t_k) = \dot{y}_k^i$ and $\ddot{y}(t_k) = \ddot{y}_k^i$ at step k can be computed using (3) with the coefficients $\{q_{k-1,j}\}$ at previous step $k-1$; and we assume a constant longitudinal acceleration a_k^i throughout the lane change process. Then, we can compute $s_{k+1}^i = [x_{k+1}^i, y_{k+1}^i, \varphi_{k+1}^i, v_{k+1}^i]^{\mathrm{T}}$ from (3) using the following equations:

$$\begin{aligned} x_{k+1}^i &= x\left(t_{k+1}|\{q_{k,j}\}_j\right),\ y_{k+1}^i = y\left(t_{k+1}|\{q_{k,j}\}_j\right) \\ \varphi_{k+1}^i &= \arctan\left(\dot{y}\left(t_{k+1}|\{q_{k,j}\}_j\right)/\dot{x}\left(t_{k+1}|\{q_{k,j}\}_j\right)\right) \\ v_{k+1}^i &= \dot{x}\left(t_{k+1}|\{q_{k,j}\}_j\right). \end{aligned} \tag{5}$$

Repeating this procedure for $k = 0, 1, \ldots, N_{\text{lane}} - 1$, we can synthesize a smooth lane change trajectory $\{s_k^i\}_{k=0,\ldots,N_{\text{lane}}}$ with corresponding acceleration sequences $\{a_k^i\}_{k=0,\ldots,N_{\text{lane}}-1}$. Fig. 5 illustrates this approach to the lane change modeling. Given an acceleration sequence $\{a_k^i\}_k$ [see Fig. 5(b)] from one of the 225 control sequences $\{u_k^i\}_k$, we leverage (3) and produce a smooth lane change trajectory that qualitatively matches with a real-world (high-D) lane change trajectory. Meanwhile, the resulting steering angle profile $\{\delta_k^i\}_k$ [see Fig. 5(c)] is similar to those from human driving [35], [36].

III. Social Behavior Modeling

In this section, we model the two components of drivers' driving incentives that motivate them to take action from the trajectory sets defined in Section II. The first component consists of each individual driver's objectives as a personal reward in Section III-A. The second component uses an SVO-based reward to incorporate the drivers' social cooperativeness

(see Section III-B). In Section III-C, we integrate this reward model into the interacting vehicle's decision-making process.

A. Driver's Driving Objectives and Personal Rewards

Similar to the previous work [27], we model the personal reward of the ith driver who interacts with an adjacent vehicle j using the following formula:

$$\begin{aligned} &r\left(\gamma_{n_1}^{n_2}(s_k^i), \gamma_{n_1}^{n_2}\left(s_k^j\right)|w^i\right) \\ &\quad = \neg c\left(\gamma_{n_1}^{n_2}(s_k^i), \gamma_{n_1}^{n_2}\left(s_k^j\right)\right) \\ &\qquad \cdot \left[h\left(s_{k+n_2}^i, s_{k+n_2}^j\right) \quad \tau\left(s_{k+n_2}^i\right) \quad e\left(\gamma_{n_1}^{n_2}(s_k^i)\right)\right] \cdot w^i \end{aligned} \tag{6}$$

where s_k^i, s_k^j are the current states of the vehicles i, j; $\gamma_{n_1}^{n_2}(s_k^i) = \{s_{k+n}^i\}_{n=n_1}^{n_2} \subset \gamma(s_k^i)$ is a segment of the trajectory $\gamma(s_k^i) \in \Gamma(s_k^i)$, and $0 \le n_1 \le n_2 \le N+1$; $\gamma_{n_1}^{n_2}(s_k^j)$ is defined likewise; $\neg$ is the logical negative operator; $w^i \in \mathbb{R}^3$ is a vector of weights so that the personal reward is a weighted summation of personal objectives h, τ, and e. Here, c, h, τ, and e are four functions that capture different aspects of drivers' driving objectives.

1) Collision avoidance $c \in \{0, 1\}$: $c\left(\gamma_{n_1}^{n_2}(s_k^i), \gamma_{n_1}^{n_2}(s_k^j)\right) = 1$ implies vehicle i following trajectory $\gamma_{n_1}^{n_2}(s_k^i)$ collides with vehicle j which follows trajectory $\gamma_{n_1}^{n_2}(s_k^j)$, and $c = 0$ indicates that two vehicles' trajectories are free of collision with each other. This is used to penalize collisions between trajectories.
2) Safety consciousness $h \in [0, 1]$: If vehicle j is the leading vehicle of vehicle i, $h(s_{k+n_2}^i, s_{k+n_2}^j)$ computes a normalized time-to-collision (TTC) at the end of their corresponding trajectories $\gamma_{n_1}^{n_2}(s_k^i), \gamma_{n_1}^{n_2}(s_k^j)$. A larger h implies a larger TTC with the leading vehicle. The safety consciousness can encourage vehicles to keep an appropriate headway distance and be conscious of potential collisions.
3) Traveling time $\tau \in [0, 1]$: $\tau(s_{k+n_2}^i)$ measures the closeness between the vehicle's final state in the trajectory $\gamma_{n_1}^{n_2}(s_k^i)$ with its destination, where a larger value implies shorter distance to the goal. Including τ in the reward reflects the objective of shortening the traveling time, e.g., merging to the highway as soon as possible for the on-ramp vehicles.
4) Control effort $e \in [0, 1]$: $e\left(\gamma_{n_1}^{n_2}(s_k^i)\right)$ takes a lower value if $\gamma_{n_1}^{n_2}(s_k^i)$ is a lane-changing trajectory segment or generated with longitudinal acceleration/deceleration. The control effort captures drivers' desire to keep the lane and constant speed to avoid both longitudinal and lateral maneuvers.

We refer the readers to our previous work [27] for more detailed descriptions of the four functions. Similar methods that model the driver's driving objectives have also been reported in [19], [37], [38], [39], and [40]. The weight w^i is the latent model parameter in the reward function $r(\cdot|w^i)$. Different weights reflect distinct personal goals and, therefore, embed various driving behaviors. For example, a driver considering personal reward with $w^i = [0, 0, 1]^{\mathrm{T}}$ may keep the lane and drive at a constant speed, thereby maximizing the reward via minimizing the control effort. Another driver with weights $w^i = [1, 0, 0]^{\mathrm{T}}$ tries to maximize the headway distance and might change lanes to overtake a leading vehicle if there is one.

B. SVO and Multimodal Reward

The personal reward function $r(\cdot|w^i)$ captures drivers' decision-making as maximizing their own gain in the traffic interaction. However, this model does not encode the behaviors of cooperation and competition. For example, a highway driver observing the merging intention of an on-ramp vehicle might slow down to yield. In social psychology studies [21], [22], the notion of SVO was proposed to model this cooperative/competitive behavior in experimental games, and it has more recently been applied to autonomous driving [13]. Taking inspiration from this work, we use the driver's SVO to incorporate the personal reward with the driver's tendency toward social cooperation.

To this end, we assume each vehicle i interacts with the adjacent vehicle $j \in A(i)$, where $A(i)$ contains indices of all the adjacent vehicles around vehicle i. We model driver i's intention using a multimodal reward function of the form

$$\begin{aligned} &R\left(\gamma_{n_1}^{n_2}(s_k^i), \boldsymbol{\gamma}_{n_1}^{n_2}(\boldsymbol{s}_k^{-i})|\sigma^i, w^i\right) \\ &\quad = \alpha(\sigma_i) \cdot r\left(\gamma_{n_1}^{n_2}(s_k^i), \gamma_{n_1}^{n_2}\left(s_k^j\right)|w^i\right) \\ &\qquad + \beta(\sigma_i) \cdot \mathop{\mathbb{E}}_{j \in A(i)}\left[r\left(\gamma_{n_1}^{n_2}\left(s_k^j\right), \gamma_{n_1}^{n_2}(s_k^i)|w^j\right)\right] \end{aligned} \tag{7}$$

where $\boldsymbol{s}_k^{-i} = [s_k^0, s_k^1, s_k^2, \ldots]$ is the aggregated state of all the adjacent vehicles of vehicle i; $\boldsymbol{\gamma}_{n_1}^{n_2}(\boldsymbol{s}_k^{-i}) = [\gamma_{n_1}^{n_2}(s_k^0), \gamma_{n_1}^{n_2}(s_k^1), \gamma_{n_1}^{n_2}(s_k^2), \ldots]$ concatenates one possible trajectory segment $\gamma_{n_1}^{n_2}(s_k^j)$ for each vehicle $j \in A(i)$; The SVO σ^i is another latent model parameter. It takes one of the four values corresponding to four SVO categories and the values of $\alpha(\sigma^i)$ and $\beta(\sigma^i)$ are specified as follows:

$$(\alpha, \beta) = \begin{cases} (0, 1), & \text{if } \sigma^i = \text{“altruistic”} \\ (1/2, 1/2), & \text{if } \sigma^i = \text{“prosocial”} \\ (1, 0), & \text{if } \sigma^i = \text{“egoistic”} \\ (1/2, -1/2), & \text{if } \sigma^i = \text{“competitive.”} \end{cases} \tag{8}$$

In (7), $\alpha(\sigma^i)$ weighs the self-reward while $\beta(\sigma^i)$ weighs an averaged reward to the other vehicles. We also note that the weight w^j is an internal parameter of vehicle j and is a latent variable affecting the decision of vehicle i. Similar to [27], we assume $w^j = [1/3, 1/3, 1/3]$ in (7) for $j \in A(i)$. The rationale behind this assumption is that an altruistic or prosocial (or competitive) driver of vehicle i is likely to cooperate (or compete) with other drivers in all three objectives if they do not know others' actual intentions.

Using this multimodal reward, we can model each driver's intention to achieve their personal objectives (reflected in w^i) and, to a certain extent, cooperate with others (encoded in σ^i). For example, suppose two highway drivers with the same personal weights $w^i = [0, 0, 1]^{\mathrm{T}}$ encounter a merging on-ramp vehicle. A "egoistic" highway driver values the control effort heavily and, therefore is likely to keep the lane at a constant

speed and ignore the merging vehicle. On the contrary, a "prosocial" highway driver might consider changing lanes or slowing down to promote on-ramp merging action such that the net reward in (7) is larger.

C. Driving Behavior Model

Using the multimodal reward, we can decode/infer drivers' intentions from their actions/trajectories, which can be represented by the model parameters w^i, σ^i. We formalize this process into a behavioral model

$$\gamma^*(s_k^i) = \underset{\gamma(s_k^i)\in\Gamma(s_k^i)}{\operatorname{argmax}} \; Q\left(\boldsymbol{s}_k^{-i}, \gamma(s_k^i)|\sigma^i, w^i\right) \tag{9}$$

where $\gamma^*(s_k^i)$ is the resulting reference trajectory for vehicle i and Q denotes the corresponding cumulative reward function. This cumulative reward admits the following form:

$$Q\left(\boldsymbol{s}_k^{-i}, \gamma\left(s_k^i\right)|\sigma^i, w^i\right) = \mathop{\mathbb{E}}_{\boldsymbol{\gamma}\left(\boldsymbol{s}_k^{-i}\right)\in\boldsymbol{\Gamma}\left(\boldsymbol{s}_k^{-i}\right)}\left[\sum_{n=0}^{\lfloor N/N'\rfloor}\lambda^n R\left(\gamma_{nN'}^{(n+1)N'}\left(s_k^i\right), \boldsymbol{\gamma}_{nN'}^{(n+1)N'}\left(\boldsymbol{s}_k^{-i}\right)|\sigma^i, w^i\right)\right] \tag{10}$$

where $\lambda \in (0,1)$ is a discount factor; the summation is a cumulative reward of vehicle i over a $N\Delta T$ sec look-ahead/prediction horizon, and this reward is obtained according to (7); $N'\Delta T$ $(N' < N)$ in second denotes the sampling period that the driver updates its decision; $\lfloor x \rfloor$ denotes the largest integer lower bound of $x \in \mathbb{R}$; the expectation averages the reward over all possible aggregated trajectories in the set

$$\boldsymbol{\Gamma}\left(\boldsymbol{s}_k^{-i}\right) = \left\{\boldsymbol{\gamma}\left(\boldsymbol{s}_k^{-i}\right) : \gamma\left(s_k^j\right) \in \Gamma(s_k^j),\, j \in A(i)\right\}. \tag{11}$$

After obtaining the optimal $\gamma^*(s_k^i)$ using (9), we compute the control signal $u_n^i = [a_n^i, \delta_n^i]^{\mathrm{T}}$ at each time step $n = k, \ldots, k+N'$ to track this reference trajectory $\gamma^*(s_k^i)$ for one sampling period $N'\Delta T$ sec. Then, we update the reference trajectory using (9) after $N'\Delta T$ sec. Eventually, using the behavioral model (9), we formulate the decision-making process of a highway driver motivated by the reward (10) and a combination of social psychological model parameters σ^i, w^i, while the driving behaviors are formalized as a receding-horizon optimization-based trajectory-tracking controller with a horizon of $\lfloor N/N' \rfloor$. However, solving this problem online can be computationally demanding. We can use a neural network to learn the solutions of (9) offline from a dataset, thereby imitating this behavioral model for online deployment.

IV. Interaction-Aware Imitation Learning With Attention Mechanism

Based on (9), the decision-making process of vehicle i is deterministic given $s_k^i, \boldsymbol{s}_k^{-i}$. However, the evaluation of cumulative reward (10) requires forward simulation of the traffic from the initial states s_k^i, $\boldsymbol{s}_k^{-i}$ which is computationally demanding. To alleviate the computational burden, we implement a neural network to approximate (i.e., imitate) the behavior model (9), (10). Instead of learning a one-hot encoding and to include stochasticity in the decision-making process, we prescribe a policy distribution from (10) using a softmax decision rule [41] according to

$$\pi\left(\gamma\left(s_k^i\right)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right) \propto \exp\left[Q\left(\boldsymbol{s}_k^{-i}, \gamma\left(s_k^i\right)|\sigma^i, w^i\right)\right]. \tag{12}$$

Note that π takes values in $\mathbb{R}^{225}$, where we assign zero probabilities in π for the unsafe trajectories $\gamma_{\text{unsafe}}(s_k^i) \notin \Gamma(s_k^i)$ filtered out in Section II-B. Then, we can train a neural network mapping $\pi_{NN}\left(\gamma(s_k^i)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right)$ to approximate the actual behavioral model π using minimization of a modified Kullback–Leibler divergence according to the loss function

$$\begin{aligned}\mathcal{L}\left(\pi, \pi_{NN}|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right) &= \sum_{m=1}^{225}\Big\{\pi\left(\gamma^{(m)}\left(s_k^i\right)\right)\cdot\log\left[\pi\left(\gamma^{(m)}\left(s_k^i\right)\right)+\epsilon\right] \\ &\quad - \pi\left(\gamma^{(m)}\left(s_k^i\right)\right)\cdot\log\left[\pi_{NN}\left(\gamma^{(m)}\left(s_k^i\right)\right)+\epsilon\right]\Big\}\end{aligned} \tag{13}$$

where a positive constant $\epsilon \ll 1$ is chosen to avoid zero probability inside the logarithm for numerical stability, and for simplicity, we omit the terms $\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}$ in the notation of π, π_{NN}. This loss function $\mathcal{L}(\cdot) \geq 0$ measures the similarity between two discrete probability distributions where smaller loss implies more similar distributions.

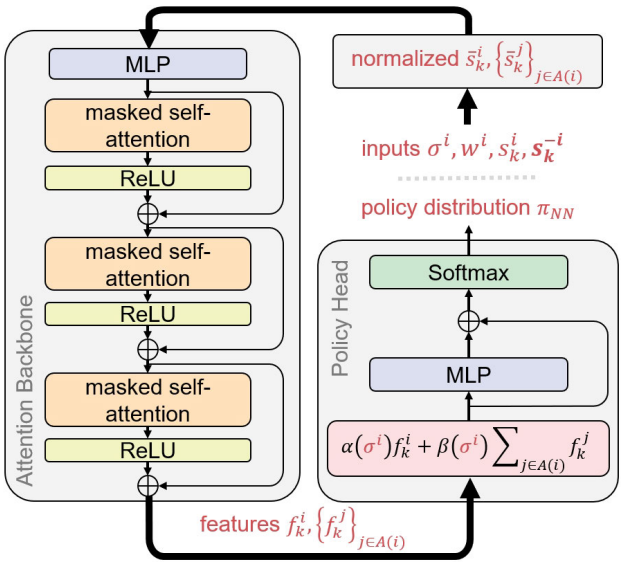

Fig. 6. Schematic of our SANN architecture: The attention backbone takes the normalized input vectors and produces their corresponding feature vectors via the attention mechanism [42]. The policy head fits a policy distribution π_{NN} from the resulting feature vectors incorporating the driver i's SVO σ^i.

For the neural network, we adopt a SANN architecture (see Fig. 6) that comprises three components. The input normalization (see Section IV-A) derives a set of normalized vectors from inputs $\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}$ accounting for different highway structural dimensions. The attention mechanism [42] enables information exchanging between input vectors. Following [42], we generate a set of feature vectors via an interaction-aware learning process using the attention backbone in Section IV-B, and we present the attention mechanism in Section IV-C. Finally, the policy head fuses the learned features using the driver's SVO σ^i and imitates the policy distribution π from the behavioral model (see Section IV-D). Properties of the SANN are summarized at end of this section.

A. Input Normalization

Given different lane dimensions as labeled in Fig. 1, we aim to normalize the inputs σ^i, w^i, s_k^i, s_k^{-i} accordingly to facilitate the neural network training. The normalization produces input vectors $\bar{s}_k^i$ for vehicle i and a set of vectors $\{\bar{s}_k^j\}$ for $j \in A(i)$ according to

$$\bar{s}_k^\iota = \begin{bmatrix} \left(x_k^\iota - l^\iota/2 - x_{\text{ramp}}\right)/l_{\text{ramp}} \\ \left(x_k^\iota + l^\iota/2 - x_{\text{ramp}}\right)/l_{\text{ramp}} \\ \left(y_k^\iota - y_{\text{ramp}}\right)/w_{\text{ramp}} \\ (v_k^\iota - v_{\min})/(v_{\max} - v_{\min}) \\ w^\iota \end{bmatrix}, \quad \iota \in \{i\} \cup A(i) \tag{14}$$

where l^ι is the wheelbase length of vehicle ι; the first two elements of the feature vector (14) are the normalized longitudinal coordinates of the vehicle rear end and front end; $w^\iota = [1/3, 1/3, 1/3]$ for all $\iota \in A(i)$ per Section III-B.

B. Attention Backbone

We design the attention backbone to interchange information between each feature vector corresponding to each interacting driver, which emulates the traffic interaction. We first use a multilayer perceptron (MLP), i.e., a fully connected neural network, to expand the dimension of the inputs $\{\bar{s}_k^\iota\}$ individually from $\mathbb{R}^7$ to $\mathbb{R}^{225}$ according to

$$z_\ell = \sigma_{\text{ReLU}}(W_\ell z_{\ell-1} + b_\ell), \quad \ell = 1, \ldots, L \tag{15}$$

where W_ℓ and b_ℓ are the network parameters of the ℓth layer; $\sigma_{\text{ReLU}}(z) = \max\{0, z\}$ is an element-wise ReLU activation function; $L \in \mathbb{Z}$ is the number of layers; the inputs are $z_0 = \bar{s}_k^\iota \in \mathbb{R}^5$, $\iota \in \{i\} \cup A(i)$; the outputs of the MLP are learned vectors $z_k^\iota = z_L \in \mathbb{R}^{225}$, $\iota \in \{i\} \cup A(i)$. Then, we combine the learned vectors $\{z_k^\iota\}_\iota$, into one matrix $Z = [z_k^i, \ldots, z_k^j, \ldots]^{\text{T}}$ where each row corresponds to a learned feature vector $(z_k^\iota)^{\text{T}}$. The row dimension of Z can vary with the numbers of interacting vehicles in $A(i)$, which is undesirable for forming a batch tensor in neural network training [43]. Thus, we consider a maximum of $N_z - 1$ adjacent vehicles in $A(i)$ such that we can append zero rows to Z and construct $Z \in \mathbb{R}^{N_z \times 225}$. Moreover, we use a mask matrix $H \in \mathbb{R}^{N_z \times N_z}$ to mark down the indices of the appended rows for the latter masked self-attention process. The element of H in the ith row and jth column attains $H_{i,j} = -\infty$ if the ith or jth row vector in Z is an appended zero vector, and obtains $H_{i,j} = 0$ otherwise.

Subsequently, we pass Z through three identical cascaded blocks (see Fig. 6) using the following formula:

$$\begin{aligned} \bar{Z}_\ell &= \texttt{Attention}(Z_{\ell-1} | W_{Q,\ell}, W_{K,\ell}, W_{V,\ell}) \\ Z_\ell &= \sigma_{\text{ReLU}}(\bar{Z}_\ell) + Z_{\ell-1}, \quad \ell = 1, 2, 3 \end{aligned} \tag{16}$$

where $\texttt{Attention}(\cdot)$ denotes the masked self-attention block [42]; $W_{Q,\ell} \in \mathbb{R}^{225 \times |Q|}$, $W_{K,\ell} \in \mathbb{R}^{225 \times |Q|}$, and $W_{V,\ell} \in \mathbb{R}^{225 \times |V|}$ are the parameters named query, key, and value matrix of the ℓth masked self-attention; the inputs are $Z_0 = Z$ and each layer $\ell = 1, 2, 3$ produces $Z_\ell \in \mathbb{R}^{N_z \times 225}$; the summation outside σ_{ReLU} corresponds to the bypass connection in Fig. 6 from the beginning of a masked self-attention block to the summation symbol $\oplus$. This bypass connection is called residual connection [44] and is designed to mitigate the vanishing gradient issue in the back-propagation of deep neural networks. We chose to cascade three such blocks via trading-off between empirical prediction performance and computational cost in comparison with those using different numbers of this block.

C. Attention Mechanism

In the ℓth attention block (16), we leverage the attention mechanism to interchange information between row vectors of the matrix $Z_{\ell-1}$ in the learning process. Specifically, the ℓth masked self-attention block can be represented using the following set of equations:

$$Q_\ell = Z_{\ell-1} W_{Q,\ell} \tag{17a}$$

$$K_\ell = Z_{\ell-1} W_{K,\ell} \tag{17b}$$

$$V_\ell = Z_{\ell-1} W_{V,\ell} \tag{17c}$$

$$E_\ell = (Q_\ell K_\ell^{\text{T}}) \circ \left[1/\sqrt{|Q|}\right]_{225 \times 225} + H \tag{17d}$$

$$P_\ell = \texttt{Softmax}(E_\ell, \text{dim=1}) \tag{17e}$$

$$\bar{Z}_\ell = P_\ell V_\ell \tag{17f}$$

where the row vectors of matrices Q_ℓ, K_ℓ, and V_ℓ are called query, key, and value vectors, respectively, learned from the corresponding row vectors in matrix $Z_{\ell-1}$; $|Q|$ and $|V|$ are the dimensions of the query and value vectors; $[x]_{a \times b}$ is a matrix of size $a \times b$ with all entries equal to x; $\circ$ is the element-wise Hadamard product; the element $e_{i,j}$ in ith row and jth column of E_ℓ represents a normalized similarity score induced by dot-product between ith query vector in Q_ℓ and jth key vector in K_ℓ; E_ℓ essentially encodes how similar two row vectors in $Z_{\ell-1}$ are with each other; (17e) apply $\texttt{Softmax}(\cdot)$ to each column of E_ℓ; the element $p_{i,j}$ in ith row and jth column of P_ℓ is equal to $\exp(e_{i,j})/\sum_k \exp(e_{k,j})$ such that each column vector of P_ℓ is a weight vector; each row vector in $\bar{Z}_\ell$ is a weighted summation of value vectors in V_ℓ using the weights learned in P_ℓ.

Notably, in the first layer $\ell = 1$, the mask H in (17d) sets the similarities between appended zero row vectors and other row vectors in $Z_0 = Z$ to $-\infty$. Subsequently, if the ith or jth row vector in Z is an appended zero vector, (17e) results in weights $p_{i,j} = 0$ in P_1 and (17f) yields the ith or jth row also a zero vector in $\bar{Z}_1$. Furthermore, the computation in (16) inherits the zero-row vectors from $\bar{Z}_1$ to Z_1. Inductively, through $\ell = 1, 2, 3$, the attention backbone preserves the zero rows in $Z_0 = Z$. Eventually, the attention backbone outputs $Z_3 = [f_k^i, \ldots, f_k^j, \ldots]^{\text{T}}$ where each row vector $(f_k^\iota)^{\text{T}}$ is a learned feature vector corresponds to the input row vector $(z_k^\iota)^{\text{T}}$ in $Z = [z_k^i, \ldots, z_k^j, \ldots]^{\text{T}}$.

D. Policy Head

Similar to (7), we introduce the notion of cooperation/competition into learning. We use the SVO category σ^i of the ith driver to combine the learned information $\{f_k^j\}_{j \in A(i)}$ from adjacent vehicles $j \in A(i)$ with f_k^i of the driver i and

attain a single vector $\bar{f}_k^i \in \mathbb{R}^{225}$ according to

$$\bar{f}_k^i = \alpha(\sigma^i) f_k^i + \beta(\sigma^i) \Sigma_{j \in A(i)} f_k^j. \tag{18}$$

We use another MLP similar to (15) with a bypass residual connection. This is followed by the final element-wise `Softmax` activation function that admits the following form:

$$\texttt{Softmax}(z)_i = \exp(z_i) / \Sigma_i \exp(z_i) \tag{19}$$

where z is a column vector and z_i is the ith element in z. The `Softmax` calculates a probability distribution as the policy distribution output $\pi_{NN} \in \mathbb{R}^{225}$.

The SANN architecture provides several advantages.

1) The normalization process normalizes the input information using lane and vehicle dimensions which improves the prediction robustness to different highway structural dimensions and vehicle model types.
2) The learning process is interaction-aware. The attention backbone interchanges information between each feature vector corresponding to each interacting driver, which captures the intertraffic dependencies in the personal reward (6).
3) The learning process is cooperation-aware. The policy head fuses the learned features using the driver's SVO σ^i. This process emulates (7) and introduces the notion of cooperation/competition into learning.
4) The SANN incorporates the behavioral model priors σ^i, w^i that are later estimated by a Bayesian filter. This offers better online transferability to different drivers.
5) The SANN is permutation invariant, namely, interchanging the order of the inputs in $\{\bar{s}_k^j\}_{j \in A(i)}$ will not alter the value of the learned features $\{f_k^j\}_{j \in A(i)}$ or affect the final policies π_{NN}. Given the interacting drivers $j \in A(i)$ geographically located in a 2-D plane, the SANN learned policy should not be affected by the artificial information carried in the input order.
6) The SANN can handle variable numbers of inputs up to a maximum number N_z. This offers better transferability to different traffic conditions/densities.

These properties might not necessarily be preserved by other networks, e.g., graph neural networks [45], the LSTM [46]. Then, we use this neural network behavioral model to predict the trajectories of interacting vehicles for the decision-making of our ego vehicle in the forced merging scenario.

V. Decision-Making Under Cooperation Intent Uncertainty

We use the Bayesian filter to infer drivers' latent model parameters from observed traffic interactions (see Section V-A). Then, based on the behavioral model, we incorporate the predictions of interacting drivers' behavior into a receding-horizon optimization-based controller to generate reference trajectories for the ego vehicle (see Section V-B). In Section V-C, we use an interaction-guided decision tree search algorithm to solve this optimization problem and integrate it with the SANN for prediction. The learned SANN improves online prediction efficiency and while the tree-search algorithm offers a probabilistic safety guarantee and good scalability.

A. Bayesian Inference of Latent Driving Intentions

At each time step k, we assume the ego vehicle 0 interacts with adjacent vehicle $i \in A(0)$ and can observe its nearby traffic state $s_k^i, \boldsymbol{s}_k^{-i}$. Assuming that the ith driver's decision-making process follows the policy (12), we need to infer the latent social psychological parameters σ^i, w^i in order to predict the future behavior/trajectory of vehicle i. We assume observation history of traffic around vehicle i is available

$$\xi_k^i = \left[s_0^i, \boldsymbol{s}_0^{-i}, s_1^i, \boldsymbol{s}_1^{-i}, \ldots, s_k^i, \boldsymbol{s}_k^{-i}\right] \tag{20}$$

which contains the state s_n^i of vehicle i and the aggregated state $\boldsymbol{s}_n^{-i}$ of adjacent vehicles around vehicle i for all time step $n = 0, 1, \ldots, k$. Per observability consideration, we also assume that the history ξ_k^i starts at a time instance t_0 when the ego vehicle is initialized in this merging scenario at s_0^0. Then, we use the following Bayesian filter to recursively estimate a posterior distribution of the latent parameters σ^i, w^i from ξ_k^i.

Proposition 1: Given a prior distribution $\mathbb{P}(\sigma^i, w^i | \xi_k^i)$ and assume the unmodeled disturbance $\tilde{w}_k^i \sim \mathcal{N}(0, \Sigma)$ in (2) follows a zero-mean Gaussian distribution, then the posterior distribution $\mathbb{P}(\sigma^i, w^i | \xi_{k+1}^i)$ admits the following form:

$$\mathbb{P}(\sigma^i, w^i | \xi_{k+1}^i) = \frac{1}{N(\xi_{k+1}^i)} \cdot D(s_{k+1}^i, \sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}) \cdot \mathbb{P}(\sigma^i, w^i | \xi_k^i) \tag{21}$$

where $N(\xi_{k+1}^i)$ is a normalization factor, and

$$D(s_{k+1}^i, \sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}) = \sum_{\gamma(s_k^i) \in \Gamma(s_k^i)} \Big[\mathbb{P}(\tilde{w}_k^i = s_{k+1}^i - \gamma_1^1(s_k^i)) \cdot \pi\left(\gamma(s_k^i) | \sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right)\Big]. \tag{22}$$

This additive Gaussian disturbance is commonly used in the vehicle kinematics models [47] per consideration of various factors such as tire friction and air drag [48]. We note that (22) represents a transition probability of a driver moving from s_k^i to s_{k+1}^i following the kinematics (2) and policy (12). We initialize the Bayesian filter with a uniform distribution. Meanwhile, we can replace π in (22) with π_{NN} for faster online Bayesian inference. We also provide a proof of the proposition in the following.

Proof: We apply the Bayesian rule to rewrite the posterior distribution according to

$$\begin{aligned}
&\mathbb{P}(\sigma^i, w^i | \xi_{k+1}^i) \\
&= \mathbb{P}(\sigma^i, w^i | s_{k+1}^i, \boldsymbol{s}_{k+1}^{-i}, \xi_k^i) \\
&= \frac{\mathbb{P}(\boldsymbol{s}_{k+1}^{-i} | \xi_k^i)}{\mathbb{P}(s_{k+1}^i, \boldsymbol{s}_{k+1}^{-i} | \xi_k^i)} \cdot \mathbb{P}(s_{k+1}^i | \sigma^i, w^i, \xi_k^i) \cdot \mathbb{P}(\sigma^i, w^i | \xi_k^i) \\
&= \frac{1}{N(\xi_{k+1}^i)} \cdot D(s_{k+1}^i, \sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}) \cdot \mathbb{P}(\sigma^i, w^i | \xi_k^i)
\end{aligned}$$

where we define $N(\xi_{k+1}^i) = (\mathbb{P}(s_{k+1}^i, \boldsymbol{s}_{k+1}^{-i} | \xi_k^i)) / (\mathbb{P}(\boldsymbol{s}_{k+1}^{-i} | \xi_k^i))$ and rewrite the transition probability

$$D(s_{k+1}^i, \sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}) = \mathbb{P}(s_{k+1}^i | \sigma^i, w^i, \xi_k^i)$$

$$= \sum_{\gamma(s_k^i)\in\Gamma(s_k^i)} \Big[\mathbb{P}\big(\tilde{w}_k^i = s_{k+1}^i - \gamma_1^1(s_k^i)\big) \cdot \pi\big(\gamma(s_k^i)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\big)\Big].$$

□

B. Receding-Horizon Optimization-Based Control

Leveraging the posterior from (21), we use a receding-horizon optimization-based controller to incorporate the trajectory predictions (12) of interacting vehicles $i \in A(0)$ and plan a reference trajectory for the ego vehicle according to

$$\gamma^*(s_k^0) = \underset{\gamma(s_k^0)\in\Gamma(s_k^0)}{\operatorname{argmax}} \; Q_0\big(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\big). \tag{23}$$

Similar to Section III-C, we compute the control signal $u_n^0 = [a_n^0, \delta_n^0]^{\mathrm{T}}$ at each time step $n = k, \ldots, k+N'$ to track this reference trajectory $\gamma^*(s_k^0)$ for one control sampling period $N'\Delta T$ sec. Then, we update the reference trajectory using (23) after $N'\Delta T$ sec. Meanwhile, the cumulative reward function $Q_0\big(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\big)$ admits the following form:

$$Q_0\Big(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\Big) = \frac{1}{|A(0)|} \sum_{i\in A(0)} \Big[\mathop{\mathbb{E}}_{\sigma^i, w^i \sim \mathbb{P}(\sigma^i, w^i|\xi_k^i)} Q_0'\Big(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)|\sigma^i, w^i\Big)\Big] \tag{24}$$

where the function value Q_0' is computed according to

$$Q_0'\Big(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)|\sigma^i, w^i\Big) = \mathop{\mathbb{E}}_{\gamma(s_k^i)\sim\pi\left(\cdot|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right)} \Big[\sum_{n=0}^{\lfloor N/N'\rfloor} r_0\Big(\gamma_{nN'}^{(n+1)N'}(s_k^0), \gamma_{nN'}^{(n+1)N'}(s_k^i)\Big)\Big] \tag{25}$$

and the ego vehicle acts to minimize its traveling time and avoid collision, thereby the ego reward function r_0 attains the following form:

$$r_0\big(\gamma_{n_1}^{n_2}(s_k^0), \gamma_{n_1}^{n_2}(s_k^i)\big) = \neg c\big(\gamma_{n_1}^{n_2}(s_k^0), \gamma_{n_1}^{n_2}(s_k^i)\big) \cdot \tau\big(s_{k+n_2}^0\big).$$

The value of reward Q_0 is the averaged cumulative reward over pairwise interactions with all vehicles $i \in A(0)$ using predictions of their future trajectories. Equation (24) defines the expectation of the reward function with respect to the behavioral model parameters σ^i, w^i, while (25) samples trajectory predictions $\gamma(s_k^i)$ of vehicle i from the policy π conditioned on σ^i, w^i. In (25), π can be replaced by π_{NN} learned by the SANN to speed up the computations. Nonetheless, solving the problem (23) requires an exhaustive search over the trajectory set $\Gamma(s_k^0)$ that can be computationally demanding. Instead, we can treat the trajectory set $\Gamma(s_k^0)$ as a decision tree (see Fig. 3) and a tree-search-based algorithm can be developed to improve the computation efficiency.

Algorithm 1 Interaction-Guided Decision Tree Search

Require: $\Gamma(s_k^0)$, $A(0)$, π_{NN}, and ξ_k^i, $\Gamma(s_k^i)$, $\mathbb{P}\left(\sigma^i, w^i|\xi_{k-1}^i\right)$ for all $i \in A(0)$

initialize $Q_0\left(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\right) = 0$ for all $\gamma(s_k^0) \in \Gamma(s_k^0)$

$\mathbb{P}\left(\sigma^i, w^i|\xi_k^i\right) \propto \Big[\sum_{\gamma(s_k^i)\in\Gamma(s_k^i)} \mathbb{P}\left(\tilde{w}_k^i = s_{k+1}^i - \gamma_1^1(s_k^i)\right) \cdot \pi_{NN}\left(\gamma(s_k^i)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right)\Big] \cdot \mathbb{P}\left(\sigma^i, w^i|\xi_{k-1}^i\right)$ for all $i \in A(0)$ ▷ Bayesian filter using SANN imitated behavioral model

$\mathbb{P}\left(\gamma(s_k^i)|\xi_k^i\right) = \sum_{\sigma^i, w^i} \pi_{NN}\left(\gamma(s_k^i)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right) \cdot \mathbb{P}\left(\sigma^i, w^i|\xi_k^i\right)$ for all $i \in A(0)$ ▷ interaction behavior prediction using SANN imitated behavioral model

sort $A(0)$ w.r.t. $\left((x_k^0 - x_k^i)^2 + (y_k^0 - y_k^i)^2\right)^{1/2}$, $i \in A(0)$ in a descending order

for $i \in A(0)$ **do** ▷ prioritize search over closer interactions

  parfor $\gamma(s_k^0) \in \Gamma(s_k^0)$ **do** ▷ parallel search

    for $n = 0 \to \lfloor N/N' \rfloor$ **do** ▷ over prediction horizons

      $c_n = \sum_{\gamma(s_k^i)\in\Gamma(s_k^i)} \mathbb{P}\left(\gamma(s_k^i)|\xi_k^i\right) \cdot c\left(\gamma_{nN'}^{(n+1)N'}(s_k^0), \gamma_{nN'}^{(n+1)N'}(s_k^i)\right)$ ▷ probability of collision with i

      if $c_n > 0.5$ **then** ▷ trim unsafe decision tree branch

        $\Gamma(s_k^0) \leftarrow \Gamma(s_k^0) \setminus \Gamma_{\text{unsafe}}(s_k^0)$, $\Gamma_{\text{unsafe}}(s_k^0) := \left\{\gamma \in \Gamma(s_k^0) | \gamma_{nN'}^{(n+1)N'} = \gamma_{nN'}^{(n+1)N'}(s_k^0)\right\}$, and terminate all parallel search branches in $\Gamma_{\text{unsafe}}(s_k^0)$

      else

        $Q_0\left(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\right) \leftarrow Q_0\left(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\right) + \lambda^n \cdot r_0\left(\gamma_{nN'}^{(n+1)N'}(s_k^0), \gamma_{nN'}^{(n+1)N'}(s_k^i)\right) \cdot \mathbb{P}\left(\gamma(s_k^i)|\xi_k^i\right)$ ▷ update discounted cumulative reward

      end if

    end for

  end parfor

end for

$\gamma^*(s_k^0) = \operatorname{argmax}_{\gamma(s_k^0)\in\Gamma(s_k^0)} Q_0\left(\boldsymbol{s}_k^{-0}, \gamma(s_k^0)\right)$

return $\gamma^*(s_k^0)$

C. Interaction-Guided Decision Tree Search

For online deployment, we incorporate the SANN-imitated behavioral model π_{NN} into a decision tree search Algorithm 1 to facilitate the Bayesian filtering, trajectory prediction of interacting vehicles, and decision-making of the ego vehicle. As shown in Fig. 3, the trajectory set $\Gamma(s_k^0)$ can be viewed as a decision tree and is initiated from the current state s_k^0 as the tree root. Each trajectory $\gamma(s_k^0)$ is a branch in the decision tree $\Gamma(s_k^0)$, each state in a trajectory is a node, and each (directed) edge encodes reachability from one node to another in a certain trajectory/branch. Meanwhile, two distinct branches $\gamma^{(m_1)}(s_k^0)$, $\gamma^{(m_2)}(s_k^0)$ can share the same trajectory segments, i.e., $(\gamma^{(m_1)})_0^{n_1}(s_k^0) = (\gamma^{(m_2)})_0^{n_1}(s_k^0) = \gamma_0^{n_1}(s_k^0)$. Algorithm 1 searches over this decision tree, updates the cumulative reward (24) for each branch, and trims unsafe branches, thereby, improving the searching efficiency. For example in Fig. 3(b), the on-ramp ego vehicle is trying to merge where there is a following highway vehicle in gray. The normal lane-changing trajectories/branches share the same subset of initial trajectory segments $\gamma_0^{n_1}(s_k^0)$ and are highly likely to cause a collision with the highway vehicle. Therefore, we can trim all the lane change trajectories

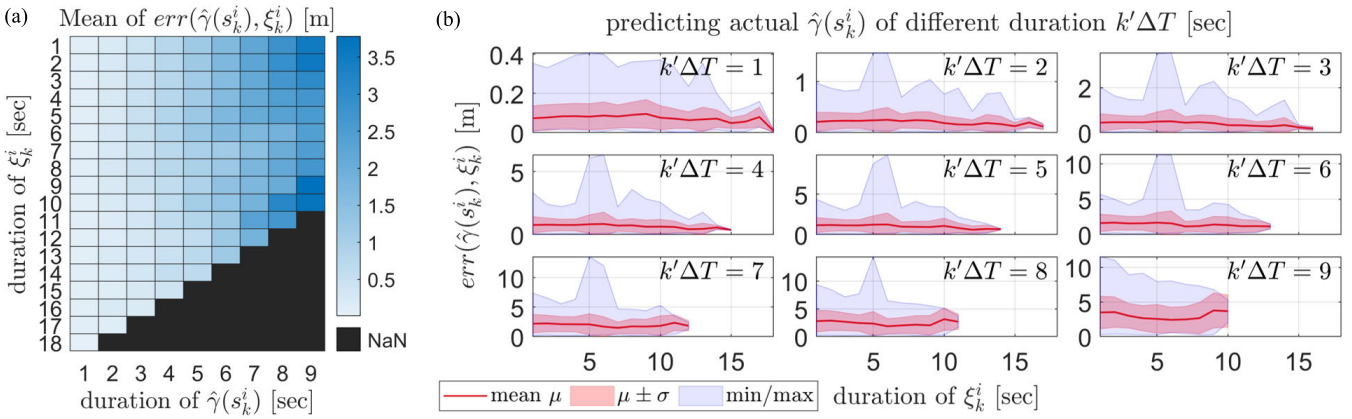

Fig. 7. Error statistics of trajectory prediction comprises 6774 high-D trajectories of in total 28 078 driving seconds. (a) Each grid reports the mean of prediction errors using ξ_k^i, $\hat{\gamma}(s_k^i)$ of the same lengths. (b) Each subplot visualizes the mean prediction errors (red lines) corresponding to $\hat{\gamma}(s_k^i)$ of the same duration versus variable duration of ξ_k^i. We use red shaded areas to denote the one standard deviation and use blue shaded areas to represent the minimum/maximum error bounds.

from the decision tree [shown by semi-transparent lines in Fig. 3(b)] and terminate further searches along these unsafe branches.

This process is formalized in Algorithm 1. In lines 2 and 3, we use the SANN behavioral model π_{NN} to update the posterior distribution in the Bayesian filter (see Proposition 1) and predict trajectory distributions that are used to compute the reward (24). Since a closer interacting vehicle is more likely to collide with the ego vehicle in the near future, we rank the indices of the adjacent vehicles in $A(0)$ in descending order with respect to their Euclidean distances to our ego vehicle. The three for-loops in line 5, 6, 7 of Algorithm 1 search over interactions with different vehicles, branches, and prediction horizons of a branch, respectively. In lines 8–13, the sorted set $A(0)$ prioritizes collision checks with closer interacting vehicles and trims the unsafe branches as early as possible. We trim all branches with the common set of nodes $\gamma_{nN'}^{(n+1)N'}(s_k^0)$ if the ego vehicle of trajectory segment $\gamma_{nN'}^{(n+1)N'}(s_k^0)$ has a probability of collision with the ith vehicle higher than a threshold of 0.5. Otherwise, we will update the cumulative reward according to line 12 in Algorithm 1. Eventually, in line 17, we solve (23) by simply choosing the branch with the maximum cumulative reward.

We also note that the three for-loops enable linear scalability of this algorithm with respect to both the number of interacting drivers in $A(0)$ and the number of prediction horizons $\lfloor N/N' \rfloor$. For a forced merging scenario with 2∼4 interacting drivers and $\lfloor N/N' \rfloor = 12$, the execution of Algorithm 1 takes 0.116 ± 0.034 s on an Intel i9-13 900F CPU and 32 GB memory which can be further broken down as follows: inference (line 2) takes 0.0154 ± 0.0022 s; prediction (line 3) takes 0.0086 ± 0.0021 s; search/control (line 4 to the end) takes 0.0921 ± 0.0342 s.

Remark 1: The prediction in Algorithm 1 is open-loop, i.e., it does not predict other vehicles' reactions to the ego actions in the prediction. A game-theoretic formulation, similar to [16] and [19], can be adapted in Algorithm 1; unfortunately, from the computational viewpoint, such an approach scales exponentially with both the number of interacting drivers and the number of prediction horizons.

Remark 2: The proposed approach can be adapted to other driving scenarios, e.g., roundabout and unsignalized intersection, with proper extensions of the state-dependent trajectory set $\Gamma(s_k^i)$, the driving objectives in Section III-A, and training dataset of π_{NN}.

VI. Simulation and Experimental Results

We first present both qualitative and quantitative results of real-world trajectory prediction using our Behavioral Model and the Bayesian Filter in Section VI-A. We report the performance of the SANN imitation learning in Section VI-B. We provide extensive evaluations of the proposed decision-making framework in forced merging tasks using our simulation environment (see Section VI-C), the real-world high-D traffic dataset [28] (see Section VI-D), and the Carla simulator [29] (see Section VI-E). We note that we **do not** need to retune the model hyperparameters or retrain the SANN for different forced merging evaluation environments. The demonstration videos ademonstration videos are available in https://xiaolisean.github.io/publication/2023-10-31-TCST2024.

A. Real-World Trajectory Prediction

We present quantitative results (see Fig. 7) including a qualitative example (see Fig. 8) of predicting real-world trajectories using our behavioral model and the Bayesian filter. In our first task, we aim to reproduce a real-world trajectory from a naturalistic high-D [28] dataset. In a high-D traffic segment (see Fig. 8) of 12 s, we identify a target vehicle i (in green) that is overtaking its leading vehicle 2 (in orange). We initialize a virtual vehicle (in red) at $t_k = 0$ using the actual vehicle state s_k^i, and we set $\sigma^i =$ "competitive," $w^i = [0, 1, 0]^{\mathrm{T}}$ in the behavioral model (9). Afterward, we control this virtual vehicle using (9) assuming no tracking error, and we set the control sampling period to $N'\Delta T = 0.5$ s. We compare the synthesized trajectories using our behavioral model with the actual ones in Fig. 8. Our behavioral model demonstrates a good prediction accuracy from 0 to 6 s, which is adequate for a predictive controller with a sampling period $N'\Delta T \leq 6$ s. We also note that the prediction error is large at $t_k = 12$ s because the error accumulates over longer prediction windows. Our prediction captures the overtaking trajectories and qualitatively demonstrates the

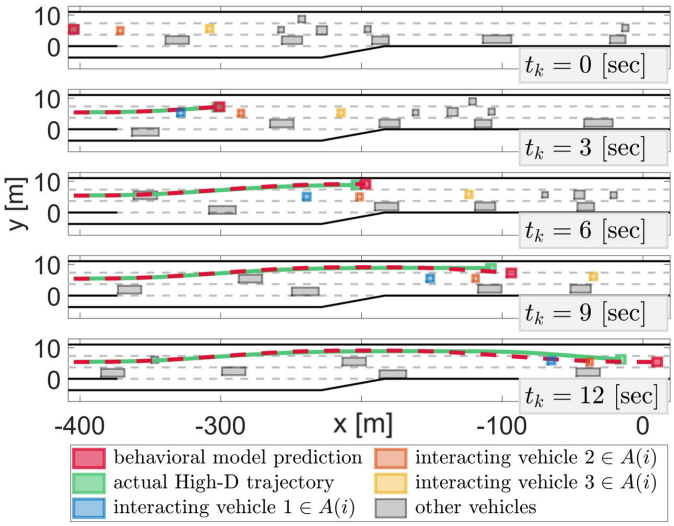

Fig. 8. Reproducing real-world overtaking trajectory: The trajectories of the virtual vehicle i (in red) are synthesized using our behavioral model (9) considering its driver of model parameters σ^i = "competitive," $w^i = [0, 1, 0]^{\mathrm{T}}$. This virtual "competitive" driver overtakes the vehicle 2, thereby, minimizing the traveling time τ. The resulting trajectories in red match the actual trajectories in green.

effectiveness of our method in modeling real-world driving behavior.

Fig. 7 summarizes the error statistics of the trajectory prediction. In online application scenarios, given an observed interacting history ξ_k^i, we recursively infer the latent behavioral model parameters σ^i, w^i using the Bayesian filter (21) as a posterior distribution $\mathbb{P}(\sigma^i, w^i|\xi_k^i)$. Subsequently, the interacting vehicles' trajectories are predicted as a distribution using policy (12) according to $\mathbb{P}(\gamma(s_k^i)|\xi_k^i) = \sum_{\sigma^i, w^i} \pi\left(\gamma(s_k^i)|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right) \cdot \mathbb{P}(\sigma^i, w^i|\xi_k^i)$. We quantify the prediction error between the actual trajectory $\hat{\gamma}(s_k^i) = \{\hat{s}_n^i\}_n^{k+k'}$ and the predicted trajectory distribution $\mathbb{P}(\gamma(s_k^i)|\xi_k^i)$ using the following metric:

$$\mathtt{err}\left(\hat{\gamma}(s_k^i), \xi_k^i\right) = \mathop{\mathbb{E}}_{\gamma(s_k^i)\sim\mathbb{P}(\gamma(s_k^i)|\xi_k^i)} \left[\frac{1}{k'+1} \sum_{s_n^i \in \gamma(s_k^i), n=k}^{n=k+k'} \left\| \begin{bmatrix} x_n^i - \hat{x}_n^i \\ y_n^i - \hat{y}_n^i \end{bmatrix} \right\|_2 \right] \tag{26}$$

where $k'\Delta T$ in second is the duration of the actual trajectory $\hat{\gamma}(s_k^i)$. This metric computes the expected ℓ_2-norm error in position prediction averaged over time steps.

We sample different traffic segments of different duration from the high-D dataset, and each traffic segment is bisected by time instance t_k into training segments ξ_k^i and prediction segment $\hat{\gamma}(s_k^i)$ corresponding to a sampled vehicle i. We apply the aforementioned procedure to each training segment ξ_k^i and calculate the prediction error (26) using the corresponding prediction segment $\hat{\gamma}(s_k^i)$. Meanwhile, in the sequel, we assume w^i in a finite set

$$W = \left\{ \begin{matrix} [0, 0, 1], [0, 1, 1]/2, [0, 1, 0], [1, 1, 1]/3, \\ [1, 0, 1]/2, [1, 1, 0]/2, [1, 0, 0] \end{matrix} \right\} \tag{27}$$

to reduce the dimension of parameter space (σ^i, w^i). Subsequently, we have 22 possible combinations of (σ^i, w^i): We assign seven different $w^i \in W$ to three $\sigma^i \neq$ "altruistic"; if $\sigma^i =$ "altruistic," the weights w^i do not matter for the altruistic driver, as defined in (7) and (8).

As shown in Fig. 7(a), the mean prediction errors are below 4 m for predictions of 1–9 s ahead and using training segments of 1–18 s. We also note that longer training segments ξ_k^i [see Fig. 7(b)] reduce both the prediction error standard deviation and its maximum values. However, for longer prediction windows of 7–9 s [see Fig. 7(b)], the error accumulates and leads to larger standard deviations and mean errors which are also observed in Fig. 8. For shorter prediction windows of 1–6 s, we have a maximum error below 5 m for most of the cases and the standard deviation is smaller than 1.5 m [see Fig. 7(b)]. The results in Figs. 7 and 8 provide evidence that our algorithms have good quantitative accuracy over shorter prediction windows, and good qualitative accuracy over longer prediction windows. Based on these considerations, we set the trajectory length as $N\Delta T = 6$ s, so that we have a good prediction performance over a shorter prediction window of 6 s. This duration covers a complete lane change of $T_{\text{lane}} = 4$ s, and suffice the task of predicting interacting vehicles' trajectories for ego vehicle control.

B. Imitation Learning With SANN

The goal of imitation learning is to train the SANN to mimic the behavioral model. Namely, the predicted policy π_{NN} should match the actual one π from behavioral model (12) accurately. We leverage the high-D dataset [28] to synthesize realistic traffic $s_k^i, \boldsymbol{s}_k^{-i}$ for training the SANN. We randomly sample a frame from the High-D traffic together with a target vehicle i, thereby we can extract states $s_k^i, \boldsymbol{s}_k^{-i}$ of vehicles i and its interacting vehicles. Together with sampled parameters σ^i, w^i, we can compute the actual policy distribution $\pi\left(\cdot|\sigma^i, w^i, s_k^i, \boldsymbol{s}_k^{-i}\right)$ using (12). We repeat this procedure to collect a dataset of 183 679 data points. We decompose the dataset into training (70%), validation (15%), and test datasets (15%). The MLP in the attention backbone has three layers of sizes 7, 32, and 225, respectively. The MLP in the policy head has two layers of sizes 225, 225. Furthermore, we set $|Q| = 32$, $|V| = 225$, and the maximum number of feature vectors as $N_z = 9$.

We use Python with Pytorch [43] to train the SANN using the training dataset and the loss function (13) which depends on different σ^i, w^i. We train the SANN for 1000 epochs using a batch stochastic gradient descent algorithm with momentum (batch size of 200). We set the initial learning rate and momentum to 0.01 and 0.9, respectively. We also adopt a learning rate scheduler that decreases the learning rate by half every 200 epochs. The training process takes in total 7 h (25 s per epoch) on a computer with 16 GB RAM and Intel Xeon CPU E3-1264 V3. We evaluate the performance of the SANN in the task of predicting the policy distributions in the test dataset, and the results are reported in Fig. 9. For the majority of the test cases, the SANN achieves losses smaller than 0.4 which corresponds to a good statistical performance. Moreover, in the example (see Fig. 9) with relatively larger losses of 0.8514, the learned policy (in blue) also qualitatively matches the actual one (in red). To conclude, the SANN imitates the proposed behavioral model with good accuracy. Hence, as presented in Algorithm 1, we use this SANN learned

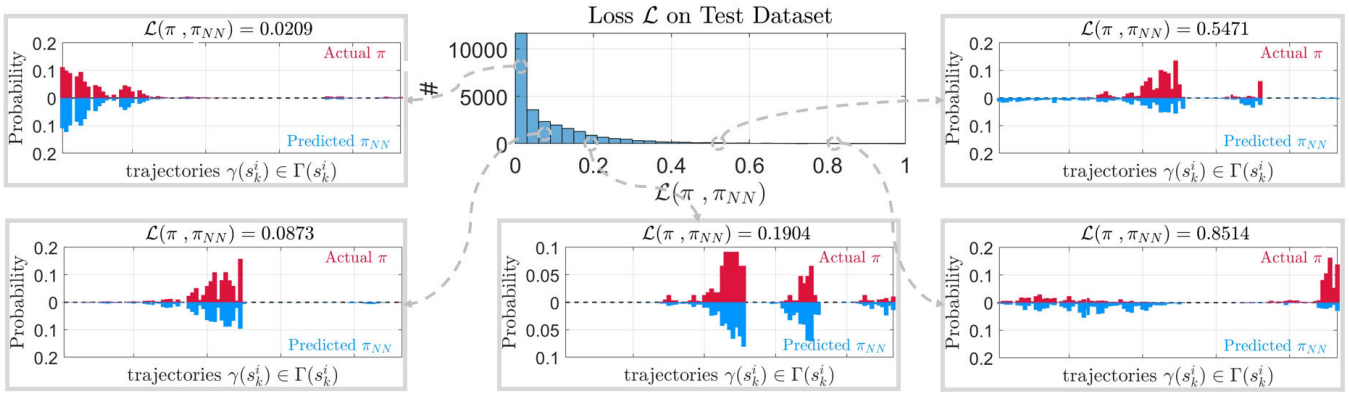

Fig. 9. Test statistics and examples of imitation learning on the test dataset. The histogram presents the loss statistics between the SANN learned policy distributions π_{NN} and the actual ones π computed from the behavioral model (12). Five qualitative examples are visualized in call-out boxes: The y-axis shows the probability of driver i taking a certain trajectory $\gamma(s_k^i)$. For comparison, policies π and π_{NN} are plotted above and below the dashed line, respectively, in mirror reflection.

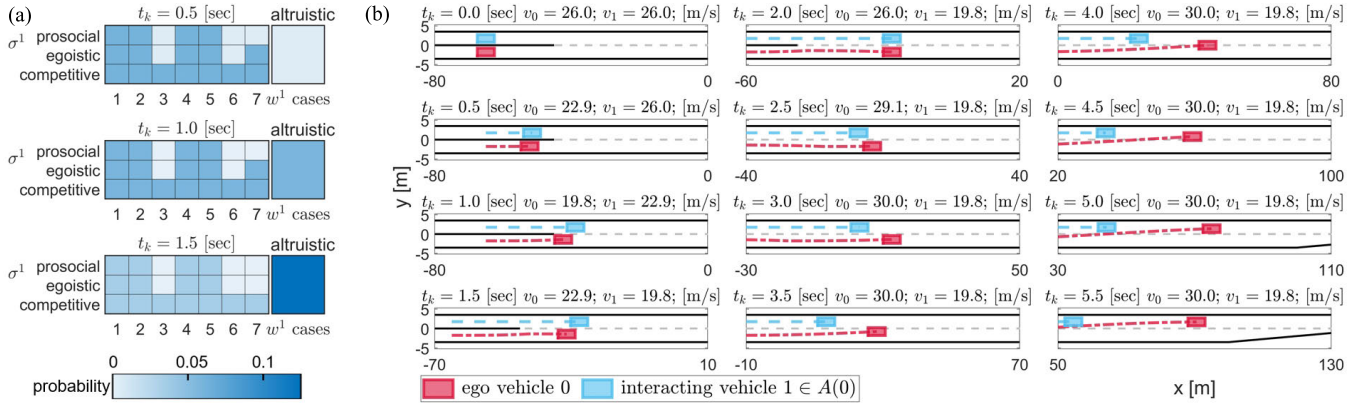

Fig. 10. Forced merging example of proactive interaction in a simulation. (a) Each subplot reports a posterior distribution of $\mathbb{P}(\sigma^i, w^i | \xi_k^i)$ at a certain time instance t_k from the Bayesian filter (21). The x-axis shows seven cases of $w^1 \in W$, and the y-axis shows three different SVO categories with $\sigma^i =$ "altruistic" stands along. (b) Each subplot visualizes a frame of highway interaction between the ego vehicle and vehicle 1 controlled by Algorithm 1 and behavioral model (9), respectively.

policy to perform Bayesian filtering and trajectory prediction to facilitate the online decision-making for our ego vehicle.

C. Forced Merging in Simulations

We set up a simulation environment (see Fig. 10) where the interacting vehicle is controlled by the behavioral model (9) with $\sigma^1 =$ "altruistic." We use the proposed Algorithm 1 to control the ego vehicle. For this and the following experiments, we set the control sampling period to $N'\Delta T = 0.5$ s for both Algorithm 1 and the behavioral model. Instead of passively inferring the driver's intention from its behavior, the Algorithm 1 controls the ego vehicle and exhibits a merging strategy with proactive interaction to test the interacting driver's intention.

Specifically, in Fig. 10(b), the interacting driver 1 first drives at a constant speed from 0 to 0.5 s. Meanwhile, from 0 to 1 s, the ego vehicle 0 is not certain if driver 1 will yield the right of way for its merging attempt and, therefore it tentatively merges after vehicle 1 with longitudinal deceleration. Meanwhile, in the time interval between 0.5 and 1.5 s, the "altruistic" driver of vehicle 1 notices the merging intention of the ego vehicle and decelerates to promote merging. In the aforementioned interactions, the ego vehicle gradually updates its belief of the latent model parameters σ^1, w^1 of driver 1. As shown in Fig. 10(a), our algorithm correctly infers the "altruistic" identity of driver 1. Subsequently, the ego vehicle aborts the merging action due to safety concerns with longitudinal acceleration to build up speed advantage for future merging. Then, in the time interval between 2.0 and 5.5 s, being aware of the yielding behavior, the ego vehicle remerges before vehicle 1 with longitudinal acceleration. This simulation example provides evidence that our method can effectively interpret the driving intentions of other drivers. Moreover, our decision-making module can leverage this information to facilitate the forced merging task while ensuring the safety of the ego vehicle.

D. Forced Merging in Real-World Traffic Dataset

We further evaluate the performance of our method in the high-D [28] real-world traffic dataset. There are 60 recordings in the high-D dataset where the recordings 58–60 correspond to highways with ramps. In recordings 58–60, we identify in total 75 on-ramp vehicles (high-D target vehicles) that merge into highways. For each one of the 75 vehicles, we extract its initial state at a recording frame when it appears on the ramp. Then, we initialize a virtual ego vehicle using this state and control this virtual ego vehicle using our decision-making module. Other vehicles are simulated using

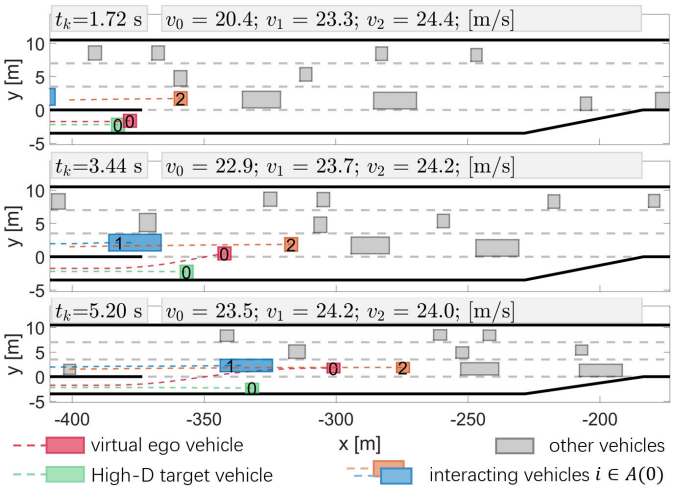

Fig. 11. Forced merging evaluation example on the high-D dataset. Our ego vehicle (in red) accelerates first to create sufficient gaps between highway vehicles 1 and 2, which are driving approximately at constant speeds. The ego vehicle successfully merges into the highway in 5.2 s which is faster than the actual human driver (in green).

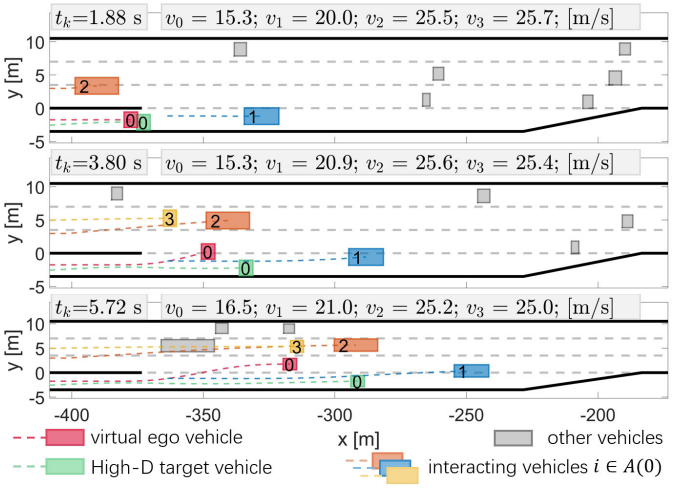

Fig. 12. Another forced merging evaluation example on the high-D dataset. The interacting vehicle 2 changes its lane in order to promote the merging action of the on-ramp vehicle. Our ego vehicle (in red) merges into the highway once observing the lane change behavior of vehicle 2. The ego vehicle successfully merges into the highway in 5.72 s which is faster than the actual human driver (in green).

the actual traffic recordings, where we neglect the interaction between the virtual ego vehicle and the high-D target vehicle. Eventually, we generate 75 forced merging test cases repeating this procedure over all target vehicles. We visualize two test cases out of 75 as examples in Figs. 11 and 12. The two interacting vehicles 1 and 2 in Fig. 11 are approximately driving at constant speeds. Therefore, our decision-making module accelerates the virtual ego vehicle to a comparable speed and merges the virtual ego vehicle in between the two interacting vehicles. In the second example (see Fig. 12), our decision-making module can interpret the yielding behavior of vehicle 2 and, thereby merge the ego vehicle highway in a timely manner. In both examples, our algorithm is able to complete the forced-merging task faster than the real-world human drivers (in green boxes).

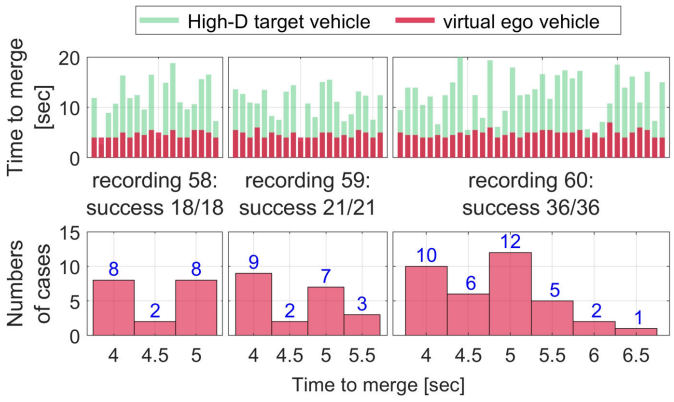

Fig. 13. Forced merging test statistics in high-D. The results are reported separately for the three recordings, and our method achieves a 100% success rate. (Upper) We visualize the time-to-merge of the virtual ego vehicles (red bars) in comparison with the ones of the actual high-D vehicles (green bars). (Lower) Three histograms visualize the number of cases in which our method takes certain seconds to merge the virtual ego vehicle.

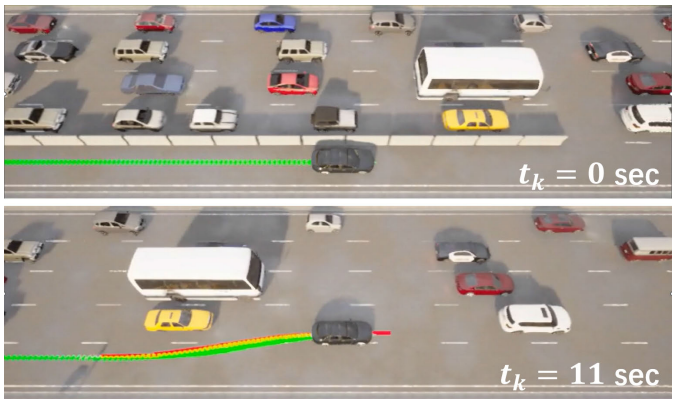

Fig. 14. Forced merging example in the Carla simulator: the ego vehicle is the black vehicle in the middle lower side of each subplot with its trajectories in green lines and reference trajectories (from our decision-making module) in red lines. The lower level PID controller accurately tracks the reference trajectory, enabling safe highway merging.

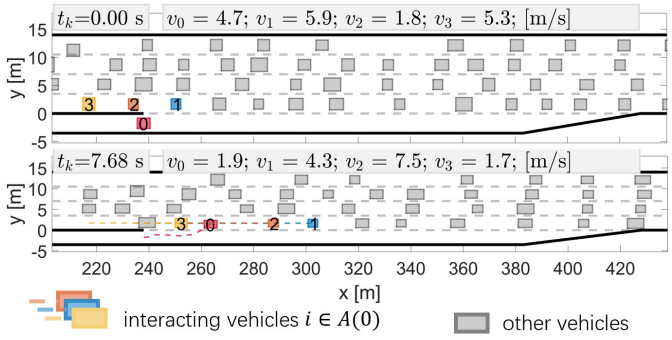

Fig. 15. Forced merging example in a dense Carla platoon: we use the default Carla Traffic Manager to control the highway vehicles. Our ego vehicle actively searches for gaps in the platoon and successfully merges into this dense traffic. $v_0 \leq v_{\min}$ is due to the tracking error of the PID controller.

Meanwhile, we identify a test case failure if our method fails to merge the virtual ego vehicle into the highway before the end of the ramp, or if the ego vehicle collides with other vehicles and road boundaries. Otherwise, we consider it a success case. In the three recordings, we have 18, 21, and 36 test cases, respectively. Our method can achieve a success rate of 100% in all recordings. We also report the results of the time it took to merge the virtual ego vehicle in Fig. 13.

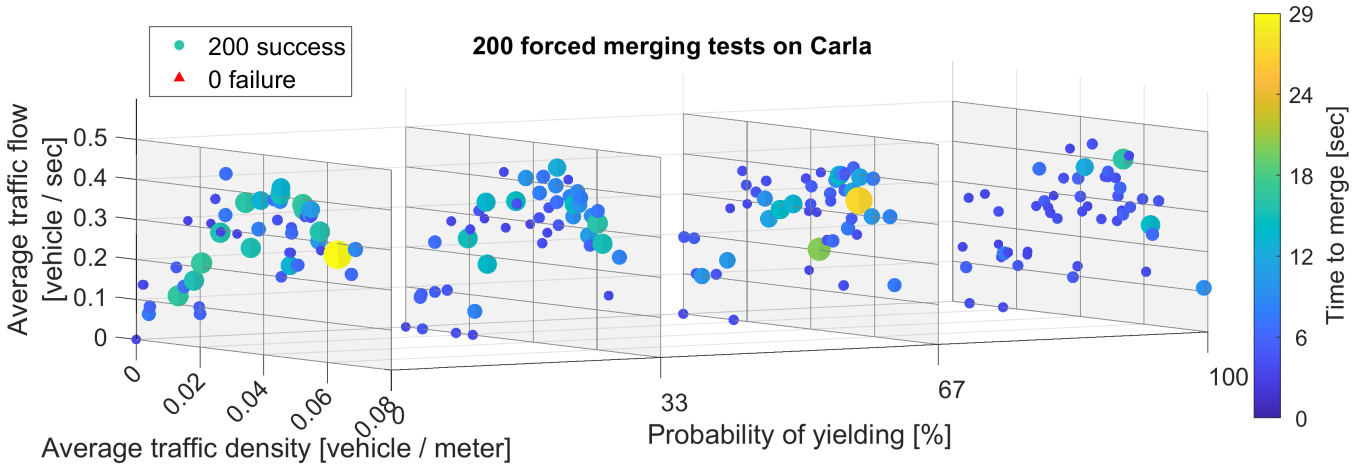

Fig. 16. Forced merging test statistics in the Carla simulator with different traffic conditions. The x-axis shows four different traffic settings where vehicles have different probabilities of being willing to yield to the ego vehicle. We have 50 test cases for each of the four traffic settings. Each dot is a single test case where the color and the size of the dot reflect the time needed to merge the ego into the highway. The y- and z-axes report the average density and flow of the traffic in the ROI.

Compared to the actual human driver of the target vehicle (green bars), our decision-making module can merge the ego vehicle into the highway using a shorter time. Meanwhile, for 64 out of the 75 test cases, our method completes the task within 5 s while we model a complete lane change with $T_{\text{lane}} = 4$ s in Section II-C. The results demonstrate that our method can successfully complete the forced merging task with traffic in real-world recordings in a timely and safe manner. Though the traffic is realistic in the dataset, we also note that there is no actual interaction between the virtual ego vehicle and other vehicles. Namely, the recorded vehicles in the dataset will not respond to the action of our virtual ego vehicle.

E. Forced Merging in Diverse Carla Traffic

We set up a forced merging scenario in the Carla simulator [29] (see Fig. 14) and test our decision-making module in diverse and reactive simulated traffic. The vehicles in the traffic are controlled by the default Carla Traffic Manager. In the previous experiments, we assumed that the ego vehicle was able to accurately track the reference trajectories $\gamma^*(s_k^0)$ from Algorithm 1. To mimic the real application scenario of our algorithm, we developed a PID controller for reference trajectory tracking. As visualized in the system diagram of Fig. 2, Algorithm 1 in the high-level decision-making module updates optimal reference trajectories $\gamma^*(s_k^0)$ every $N'\Delta T = 0.5$ s. Meanwhile, to track this reference trajectory $\gamma^*(s_k^0)$, the low-level PID controller computes the steering and throttle signal of the actual vehicle plant (simulated by Carla using Unreal Engine) at 20 Hz.

Fig. 15 illustrates our method capability to merge the ego vehicle into a dense highway platoon in Carla. From 0 to 4.60 s, the ego vehicle attempts to merge between vehicles 1 and 2 and, eventually aborts lane change due to safety concerns. From 4.60 to 6.12 s, vehicle 3 is decelerating due to a small headway distance. And vehicle 2 is accelerating seeing an enlarged headway distance as a result of the acceleration of vehicle 1 before 4.60 s. Consequently, the gap between vehicles 2 and 3 is enlarged due to this series of interactions from 4.60 to 6.12 s. Taking advantage of these interactions, our ego vehicle decides to accelerate and merge before vehicle 3 at 4.60 s.

Moreover, we are interested in quantifying the state of the traffic that the ego vehicle directly interacts with, i.e., the traffic in the lane near the ramp. We first define the traffic region of interest (ROI) as the traffic in the near-ramp lane between the longitudinal position $x_{\text{ramp}} - l_{\text{ramp}}$ and x_{ramp} (see Fig. 1). Subsequently, we define the following two variables: We compute the average traffic flow as the number of vehicles entering this traffic ROI over the entire simulation time; and we calculate the traffic density as the number of vehicles located in this traffic ROI over the length of this region l_{ramp} at a certain time instance. Then, we average the traffic density over measurements taken every $\Delta T = 0.05$ s which yields the average traffic density.

The two variables quantify the averaged speed and density of the near-ramp traffic that the ego vehicle directly interacts with. For the example in Fig. 15, the traffic in the ROI has an average traffic density of 0.055 vehicle/m and an average traffic flow 0.3646 vehicle/s. We also note that the vehicles controlled by the default Carla Traffic Manager have no interpretation of other drivers' intentions and, therefore, are unlikely to yield to the ego vehicle. This introduces more difficulties in merging into a dense vehicle platoon (such as the example in Fig. 15) even for human drivers.

Therefore, we introduce another variable, i.e., the probability of yielding, to have more diversified evaluations for our method. Specifically, for traffic with zero probability of yielding, we control all vehicles in the traffic using the default Carla Traffic Manager. For that being set to a nonzero value of $X\%$, the vehicles in the traffic ROI have $X\%$ probability of yielding to the on-ramp ego vehicle. The yielding behavior is generated by the Cooperate Forced Controller [49]. Similar to the intelligent driver model (IDM) [50] that controls a vehicle to follow the leading vehicle ahead, the Cooperate Forced

Controller is a car-following controller derived from the IDM, but assuming the on-ramp ego vehicle is the leading vehicle.

We test our algorithm at various traffic conditions with different settings. As shown in Fig. 16, we achieve a 100 % success rate among 200 test cases where we have different traffic densities and different probabilities of yielding for interacting vehicles. In the majority of the test cases, the ego vehicle takes less than 9 s to merge. Generally, in denser traffic, our algorithm needs more time to interact with more vehicles and, thus, more time to merge. With a higher probability of yielding, the algorithm takes less time to complete the merging tasks on average. Notably, our algorithm can achieve a 100% success rate even in traffic with zero probability of yielding.

VII. Conclusion

In this article, we proposed an interaction-aware decision-making module for autonomous vehicle motion planning and control. In our approach, interacting drivers' intentions are modeled as hidden parameters in the proposed behavioral model and are estimated using a Bayesian filter. Subsequently, interacting vehicles' behaviors are predicted using the proposed behavioral model. For the online implementation, the SANN was designed and utilized to imitate the behavioral model in predicting the interacting drivers' behavior. The proposed approach is easily transferable to different traffic conditions. In addition, the decision tree search algorithm was proposed for faster online computation; this algorithm leads to linear scalability with respect to the number of interacting drivers and prediction horizons. Finally, a series of studies, based on naturalistic traffic data and simulations, were performed to demonstrate the capability of the proposed decision-making module. In particular, we have shown that the behavioral model has good prediction accuracy, and the proposed algorithm is successful in merging the ego vehicle in various simulations and real-world traffic scenarios, without the need for returning the model hyperparameters or retraining the SANN.

References

[1] E. Polders, S. Daniels, W. Casters, and T. Brijs, "Identifying crash patterns on roundabouts," *Traffic Injury Prevention*, vol. 16, no. 2, pp. 202–207, Feb. 2015.

[2] K. Haleem and M. Abdel-Aty, "Examining traffic crash injury severity at unsignalized intersections," *J. Saf. Res.*, vol. 41, no. 4, pp. 347–357, 2010.

[3] A. T. McCartt, V. S. Northrup, and R. A. Retting, "Types and characteristics of ramp-related motor vehicle crashes on urban interstate roadways in northern Virginia," *J. Saf. Res.*, vol. 35, no. 1, pp. 107–114, Jan. 2004.

[4] Y. Hu et al., "Planning-oriented autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 17853–17862.

[5] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "LiDAR-based driving path generation using fully convolutional neural networks," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6.

[6] D. Barnes, W. Maddern, and I. Posner, "Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 203–210.

[7] M. Huegle, G. Kalweit, M. Werling, and J. Boedecker, "Dynamic interaction-aware scene understanding for reinforcement learning in autonomous driving," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2020, pp. 4329–4335.

[8] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[9] Y. Pan et al., "Agile autonomous driving using end-to-end deep imitation learning," 2017, *arXiv:1709.07174*.

[10] S. Bae et al., "Lane-change in dense traffic with model predictive control and neural networks," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 2, pp. 646–659, Mar. 2023.

[11] Y. Chen, P. Karkus, B. Ivanovic, X. Weng, and M. Pavone, "Tree-structured policy planning with learned behavior models," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7902–7908.

[12] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proc. Robot., Sci. Syst.*, vol. 2, Ann Arbor, MI, USA, 2016, pp. 1–9.

[13] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 50, pp. 24972–24978, 2019.

[14] Y. Chen, U. Rosolia, W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Interactive multi-modal motion planning with branch model predictive control," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5365–5372, Apr. 2022.

[15] R. Oliveira, S. H. Nair, and B. Wahlberg, "Interaction and decision making-aware motion planning using branch model predictive control," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2023, pp. 1–8.

[16] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 727–733.

[17] M. Liu, I. Kolmanovsky, H. E. Tseng, S. Huang, D. Filev, and A. Girard, "Potential game-based decision-making for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8014–8027, Aug. 2023.

[18] R. Tian, M. Tomizuka, and L. Sun, "Learning human rewards by inferring their latent intelligence levels in multi-agent games: A theory-of-mind approach with application to driving data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 4560–4567.

[19] K. Liu, N. Li, H. E. Tseng, I. Kolmanovsky, and A. Girard, "Interaction-aware trajectory prediction and planning for autonomous vehicles in forced merge scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 474–488, Jan. 2023.

[20] S. Hoermann, D. Stumper, and K. Dietmayer, "Probabilistic long-term prediction for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 237–243.

[21] D. M. Messick and C. G. McClintock, "Motivational bases of choice in experimental games," *J. Experim. Social Psychol.*, vol. 4, no. 1, pp. 1–25, Jan. 1968.

[22] W. B. G. Liebrand, "The effect of social motives, communication and group size on behaviour in an N-person multi-stage mixed-motive game," *Eur. J. Social Psychol.*, vol. 14, no. 3, pp. 239–264, Jul. 1984.

[23] D. Lenz, T. Kessler, and A. Knoll, "Tactical cooperative planning for autonomous highway driving using Monte-Carlo tree search," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 447–453.

[24] L. Crosato, C. Wei, E. S. L. Ho, and H. P. H. Shum, "Human-centric autonomous driving in an AV-pedestrian interactive environment using SVO," in *Proc. IEEE 2nd Int. Conf. Hum.-Mach. Syst. (ICHMS)*, Sep. 2021, pp. 1–6.

[25] L. Crosato, H. P. H. Shum, E. S. L. Ho, and C. Wei, "Interaction-aware decision-making for automated vehicles using social value orientation," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 2, pp. 1339–1349, Feb. 2023.

[26] Z. Peng, Q. Li, K. M. Hui, C. Liu, and B. Zhou, "Learning to simulate self-driven particles system with coordinated policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 10784–10797.

[27] X. Li, K. Liu, H. Eric Tseng, A. Girard, and I. Kolmanovsky, "Interaction-aware decision-making for autonomous vehicles in forced merging scenario leveraging social psychology factors," 2023, *arXiv:2309.14497*.

[28] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2118–2125.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[30] R. Rajamani, *Vehicle Dynamics and Control*. Berlin, Germany: Springer, 2011.

[31] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2015, pp. 1094–1099.
[32] I. Papadimitriou and M. Tomizuka, "Fast lane changing computations using polynomials," in *Proc. Amer. Control Conf.*, 2003, pp. 48–53.
[33] M. Yue, X. Hou, X. Zhao, and X. Wu, "Robust tube-based model predictive control for lane change maneuver of tractor-trailer vehicles based on a polynomial trajectory," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 12, pp. 5180–5188, Dec. 2020.
[34] F. You, R. Zhang, G. Lie, H. Wang, H. Wen, and J. Xu, "Trajectory planning and tracking control for autonomous lane change maneuver based on the cooperative vehicle infrastructure system," *Exp. Syst. Appl.*, vol. 42, no. 14, pp. 5932–5946, Aug. 2015.
[35] D. D. Salvucci and A. Liu, "The time course of a lane change: Driver control and eye-movement behavior," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 5, no. 2, pp. 123–132, 2002.
[36] D. D. Salvucci, "Modeling driver behavior in a cognitive architecture," *Hum. Factors*, vol. 48, no. 2, pp. 362–380, Jun. 2006.
[37] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1537–1542.
[38] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based MDP for robust single-lane autonomous driving behavior under uncertainties," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2586–2592.
[39] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2765–2771.
[40] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2017.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[42] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–15.
[43] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[45] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.
[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[47] C. Chen, M. Rickert, and A. Knoll, "Motion planning under perception and control uncertainties with space exploration guided heuristic search," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 712–718.
[48] P. Vial and V. Puig, "Kinematic/dynamic SLAM for autonomous vehicles using the linear parameter varying approach," *Sensors*, vol. 22, no. 21, p. 8211, Oct. 2022.
[49] C.-W. Kong, K. Liu, H. E. Tseng, I. Kolmanovsky, and A. Girard, "Simulation based methodology for assessing forced merging strategies for autonomous vehicles," in *Proc. Amer. Control Conf. (ACC)*, May 2023, pp. 4412–4418.
[50] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, p. 1805, 2000.

Xiao Li received the B.S. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019, and the M.S. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2021, where he is currently pursuing the Ph.D. degree in aerospace engineering.

His research interests include learning-based methods in constrained optimization and in human-in-the-loop decision-making for autonomous agents.

Kaiwen Liu received the B.S. degree in electrical and computer engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017, and the M.S. degree in mechanical engineering and the Ph.D. degree in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2019 and 2023, respectively.

His research interests include decision-making with human interactions and learning methods for constrained control problems.

H. Eric Tseng received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1986, and the M.S. and Ph.D. degrees in mechanical engineering from the University of California, Berkeley, CA, USA, in 1991 and 1994, respectively.

In May 2024, he joined the University of Texas at Arlington as a distinguished university professor in the Department of Electrical Engineering. In 1994, he joined Ford Motor Company. At Ford (1994–2022), he had a productive career and retired as a Senior Technical Leader of Controls and Automated Systems in Research and Advanced Engineering. Many of his contributed technologies led to production vehicles implementation. He has over 100 U.S. patents and over 160 publications.

Dr. Tseng is a member of the National Academy of Engineering as of 2021. His technical achievements have been honored with Ford's annual technology award, the Henry Ford Technology Award, on seven occasions. Additionally, he was the recipient of the Control Engineering Practice Award from the American Automatic Control Council in 2013, and the recipient of the Soichiro Honda Medal from the American Society of Mechanical Engineering in 2024.

Anouck Girard (Senior Member, IEEE) received the Ph.D. degree in ocean engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2002.

She has been with the University of Michigan, Ann Arbor, MI, USA, since 2006, where she is currently a Professor of robotics and aerospace engineering and the Director of the Robotics Institute. She was a Fulbright Scholar with the Dynamic Systems and Simulation Laboratory, Technical University of Crete, Kounoupidiana, Greece, in 2022. Her current research interests include vehicle dynamics and control and decision systems.

Ilya Kolmanovsky (Fellow, IEEE) received the Ph.D. degree in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 1995.

He is currently a Pierre T. Kabamba Collegiate Professor with the Department of Aerospace Engineering, University of Michigan. His research interests include control theory for systems with state and control constraints and in aerospace and automotive control applications.