IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 July 2022, accepted 14 July 2022, date of publication 25 July 2022, date of current version 8 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3193492

RESEARCH ARTICLE

Planning for Autonomous Driving via Interaction-Aware Probabilistic Action Policies

SALAR ARBABI[1], DAVIDE TAVERNINI[1], SABER FALLAH[1], AND RICHARD BOWDEN[2], (Senior Member, IEEE)
[1]Centre for Automotive Engineering, University of Surrey, Guildford, GU2 7XH, U.K.
[2]Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford GU2 7XH, U.K.

Corresponding author: Salar Arbabi (s.arbabi@surrey.ac.uk)

This work was supported in part by the Jaguar Land Rover, and in part by the U.K.-Engineering and Physical Sciences Research Council (EPSRC) jointly funded by Toward Autonomy: Smart and Connected Control (TASCC) Program under Grant EP/N01300X/1.

ABSTRACT Devising planning algorithms for autonomous driving is non-trivial due to the presence of complex and uncertain interaction dynamics between road users. In this paper, we introduce a planning framework encompassing multiple action policies that are learned jointly from episodes of human-human interactions in naturalistic driving. The policy model is composed of encoder-decoder recurrent neural networks for modeling the sequential nature of interactions and mixture density networks for characterizing the probability distributions over driver actions. The model is used to simultaneously generate a finite set of context-dependent candidate plans for an autonomous car and to anticipate the probable future plans of human drivers. This is followed by an evaluation stage to select the plan with the highest expected utility for execution. Our approach leverages rapid sampling of action distributions in parallel on a graphic processing unit, offering fast computation even when modeling the interactions among multiple vehicles and over several time steps. We present ablation experiments and comparison with two existing baseline methods to highlight several design choices that we found to be essential to our model's success. We test the proposed planning approach in a simulated highway driving environment, showing that by using the model, the autonomous car can plan actions that mimic the interactive behavior of humans.

INDEX TERMS Autonomous vehicle, autonomous driving, driver modeling, human-robot interaction, interaction-aware motion prediction.

I. INTRODUCTION

Considering that an estimated 94% of traffic accidents can be attributed to driving errors [1], there is a significant potential to improve traffic safety by automating the driving task. Despite the recent advances in sensing and control technologies, which have led to the successful demonstration of autonomous driving on public roads [2], [3], one key open research challenge lies in enabling self-driving vehicles to navigate interactive scenarios involving other drivers. An example scenario where such interactions are prevalent arises when performing lane-change maneuvers in congested traffic conditions.

Given the safety-critical nature of driving, many existing works have primarily focused on safety [4], treating vehicles in the vicinity of the autonomous vehicle as bounded disturbances [5], [6] and obstacles moving with constant velocity [3]. Alternatively, the methods involve predicting driver trajectories first and using these predictions to plan collision-free trajectories in response [2], [7]. The main limitation of these studies is that they do not consider interactions with human drivers. In contrast, humans are often aware of how their actions influence others—an autonomous vehicle that fails to recognize its coupled interactions with human drivers would potentially cause disruptions to the traffic flow and create dangerous situations [8], [9].

To consider interactions, some works have integrated models of human driving behavior in their planning frameworks. An intuitive way to construct a driver model is to learn the

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

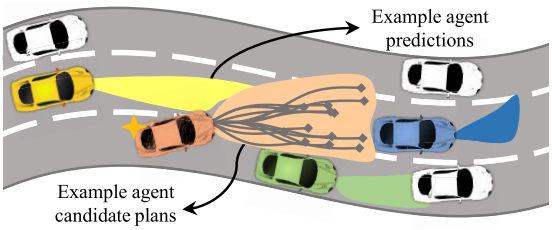


FIGURE 1. Example of an interaction scenario. The agent (orange) may fail to successfully change its lane if it cannot anticipate the human driver (yellow) yielding.

model directly from driving data [10]–[14]. We follow this paradigm to learn a parametric model of interaction dynamics from many exemplars of human-human interactions in naturalistic driving. The model consists of encoder-decoder recurrent neural networks (which we will refer to as an RNN encoder-decoder) and mixture density networks (MDNs), which are particularly well suited for modeling sequential tasks [15] and stochastic processes [16], respectively. In our approach, we use the model to both anticipate the future actions of human drivers and to generate candidate action plans for the autonomous agent. Subsequently, a planning module scores each candidate plan according to an evaluation function, and the agent plan with the highest score gets executed in a receding horizon fashion.

In this work, we are particularly interested in lane-change scenarios, similar to that shown in Fig. 1, where it may not be possible to merge successfully without interacting with other drivers. The scenario has three characteristics that motivate our approach. First, given the dense traffic condition, there are several vehicles that can influence each other. As such, interactions among multiple vehicles have to be considered. Second, the agent has to take actions while lacking knowledge of all the factors that may influence the driving behavior of humans. Such factors include varying driver preferences and the position of vehicles occluded from the agent's perceptual field. Third, the driving behavior of humans, in most cases, is inconsistent across different instants and scenarios. Thus, the agent must plan accordingly by considering the inherent uncertainty in drivers' behavior.

This paper presents the following main contributions:

1) We use deep neural networks to learn a predictive model of the non-deterministic driver behavior in multi-vehicle scenarios (Section IV-B).
2) We propose a novel training strategy suited for continuous vehicle trajectories that improves the model's prediction accuracy (Section V-B).
3) We leverage the model to efficiently plan actions that mimic the driving behavior of humans while fulfilling the agent's driving objective (Section VI-B).
4) We perform an ablation study to justify our model design choices (Section VI-D).

II. RELATED WORK

While planning in interactive traffic scenarios remains an unsolved problem, it has attracted considerable interest in the recent years. In this section, we first provide a brief overview of the existing research on driver modeling and discuss their suitability for integration in planning frameworks. We then present a review of several studies that have used driver models for planning in interactive scenarios. At last, we provide a summary of our approach and contribution in the context of former research.

A. DRIVER BEHAVIOR MODELING

Driver behavior modeling has long been a subject of study for the application of traffic flow analysis and infrastructure-based traffic management [17], [18]. The developed models determine driver actions (e.g., acceleration and turn rate) as a function of time or the traffic state. Despite being suitable for large-scale traffic simulations, these models rely on many strong inbuilt assumptions and do not account for the stochastic nature of human driving behavior.

To capture the more subtle nuances of driver behavior, several studies have proposed fitting probabilistic models to naturalistic driving data [11], [12]. The traffic state can then be propagated forward by iteratively drawing samples from each driver's action distribution, and using a vehicle dynamics model to estimate the next traffic state. For highway driving, Lenz *et al.* [11] proposed fully connected Deep Neural Networks (DNNs) to parameterize the distributions over driver actions. The traffic state from the viewpoint of each driver was used as input to the DNNs. As noted by the authors of [11], requiring access to the traffic state from the standpoint of a driver may limit the model's usage in practice due to the limited range of autonomous vehicles' onboard sensors. Schulz *et al.* [12] proposed a similar approach, but besides traffic states, they also conditioned their model on drivers' route intentions and local maps to make their model applicable to urban traffic scenarios. Perhaps surprisingly, results from both [11] and [12] suggest that a model that is only conditioned on the current state of traffic outperforms one conditioned on the history of past traffic states.

Lee *et al.* [19] proposed DESIRE, an end-to-end deep learning-based framework for tackling interaction-aware motion prediction. DESIRE consists of multiple modules, including a trajectory sample generator using a conditional variational auto-encoder, a trajectory ranking and refinement module based on inverse reinforcement learning, and a module for aggregating scene context and interactions between multiple drivers. Although the computational demand of their framework has not reported, such a complex architecture may present challenges for fast runtime inference, which is of practical importance for sampling-based planning approaches such as ours.

B. PLANNING APPROACHES

The efficacy of several planning algorithms based on the Markov decision process formulation has been demonstrated through both simulation-based [20] and real-world experiments [21]. These methods can handle various sources of uncertainty and allow for planning at multiple levels of

abstraction, e.g., high-level tactical decisions and low-level actions that directly influence the agent's dynamics. The primary weakness of the aforementioned literature is the use of hand-engineered driver models, as there may be a large mismatch between the driver models used by the autonomous vehicle for planning and the actual driving behavior of humans [22].

In [23], planning is formulated as a game whereby drivers are treated as rational agents attempting to maximize a defined objective. The value of the states vehicles can occupy is determined based on their collision risk and how far the vehicles are from reaching their desired velocity. While the action space of the agent in [23] is a set of discrete lateral and longitudinal behaviors, we are interested in planning over a continuous action space, which would incur a prohibitively high computational cost if the interactive planning problem is treated as a game. Furthermore, the employed tree-type structure in [23] grows exponentially with the number of interacting vehicles, limiting the approach's real-time applicability in congested traffic scenarios.

Assuming access to the objective that governs driver behavior, planning can be cast as a dynamical system and solved via optimization. One option is to hand-code the objective using a combination of cost terms that reflect a typical driver's goal of driving safely and efficiently [24]–[26]. To not rely on a hand-coded objective, another option is to fit a family of cost functions to data using Inverse Reinforcement Learning (IRL) [27]. One commonly used variant of IRL employs the maximum entropy principle [28], where the aim is to maximize the likelihood of expert actions under a parameterized objective. Using this principle in the context of planning, Sadigh *et al.* [13] demonstrated that by exploiting a learned driver objective, an autonomous vehicle is able to both anticipate and influence the future action responses of a human driver. To render optimization tractable, it is assumed that the human has access to the planned actions of the autonomous vehicle and responds optimally according to his or her objective. This formulation is known as a Stackelberg game, which offers a lower computational load by avoiding the recursive turn-taking of the full game-theoretic setting. Nevertheless, the learned objective in [13] could only express the single most probable human response. In contrast, in this work we aim to capture a diverse range of human responses.

Schmerling *et al.* [14] presented an approach similar to ours, whereby a data-driven model of interaction dynamics is used for interactive planning. Importantly, their model can capture multimodal human action distributions, and their employed receding horizon formulation yields fast responses to human actions. While we use naturalistic vehicle trajectories as training data, in [14], training data were collected using a simulator, which may not elicit the same driving behaviors that occur when driving in real traffic with high-risk consequences [29]. Furthermore, the agent's action policy is constructed by evaluating a large number of context-independent candidate plans. We instead aim to learn the agent's action policy, allowing for the efficient exploration of the space of context-dependent candidate plans.

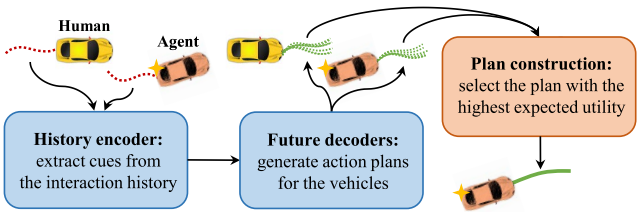


FIGURE 2. Proposed action planning cycle. The plan construction module takes as input the candidate plans generated by the decoders, evaluates them, and selects the plan with the highest expected utility for execution.

C. OUR CONTRIBUTIONS

Thus, there is a great deal of work remaining in enabling self-driving vehicles to navigate scenarios involving human-robot interaction. Our ultimate goal is to enable an autonomous vehicle to drive in congested traffic, which exemplifies many scenarios involving driver interactions. Our central idea is to model the interactions among several human drivers and the autonomous agent via learned action policies, for the purpose of interactive planning. In contrast to other approaches in the literature that only predict the most likely human actions [13] or learn a driver model solely for making predictions [14], in this work we use the policies for the probabilistic motion prediction of multiple vehicles and for generating candidate agent plans. By doing so, the agent can learn to emulate the interactive driving behavior of humans with all its richness and complexities, which is naturally hard to achieve using hand-coded strategies.

III. PROBLEM STATEMENT

The planning cycle for the proposed framework in this paper is shown in Fig. 2. We seek to devise a planner that, given the surrounding vehicles' past motion history, produces an action plan which is applied by the agent to move from its current position towards a goal position, subject to safety and interactivity. To evaluate the agent's choices of actions, the planner relies on the anticipation of how the neighboring drivers might act in the future. Specifically, the safety of the agent's planned paths with respect to the movement of other traffic participants has to be considered. To this end, a state transition function is used, which specifies how the traffic state evolves over time as a function of the agent's and the other vehicles' actions.

A. STATE TRANSITION FUNCTION

To formalize the state transition function, we follow multi-policy decision-making (MPDM), a theoretically grounded formulation of planning for interactive, multi-vehicle scenarios originally presented in [21]. We briefly describe the MPDM formulation and how it has been adapted for our planning framework; for complete details, we guide the readers to [21].

The notation used throughout the paper is as follows. Each vehicle $v \in V$ can take an action $a_v^t \in \mathcal{A}_v$ at any given time

t to transition from a state $x_v^t \in \mathcal{X}_v$ to x_v^{t+1}. An action a_v^t is a tuple of a vehicle's continuous longitudinal acceleration and lateral speed, and state x_v^t contains the continuous vehicle position and speed. Let x^t be a joint state, which contains the state of the agent and human-driven vehicles. Let e denote the agent and z_e^t denote the agent's observation of surrounding vehicles' state. Similarly, each vehicle can access the state of its neighboring vehicles through an observation $z_v^t \in \mathcal{Z}_v$. Assuming independence between the instantaneous actions of each vehicle, the transition probability is governed by,

$$p(x^{t+1}) \approx \int_{\mathcal{X}_e}\int_{\mathcal{Z}_e} p_e(x_e^t, x_e^{t+1}, z_e^t, a_e^t)\, dz_e^t\, dx_e^t \times \prod_{v\in V|v\neq e} \int_{\mathcal{X}_v}\int_{\mathcal{Z}_v}\int_{\mathcal{A}_v} p_v(x_v^t, x_v^{t+1}, z_v^t, a_v^t)\, da_v^t\, dz_v^t\, dx_v^t \tag{1}$$

where $p_v(x_v^t, x_v^{t+1}, z_v^t, a_v^t)$ is the joint density for vehicle v. MPDM assumes that each vehicle follows a high-level decision policy $\pi_v^t \in \Pi$ (e.g., keeping lane or changing lane), where a decision policy is a mapping $\pi_v : \mathcal{Z}_v \times \mathcal{X}_v \to \mathcal{A}_v$. As such, the joint distribution factors as follows:

$$p_v(x_v^t, x_v^{t+1}, z_v^t, a_v^t) = p(x_v^t)p(z_v^t|x_v^t)p(x_v^{t+1}|x_v^t, a_v^t) \times \underbrace{p(\pi_v^t|x_v^t, z_v^{0:t})}_{\text{decision policy}} \underbrace{p(a_v^t|x_v^t, z_v^t, \pi_v^t)}_{\text{action policy}} \tag{2}$$

In [21], the primary focus was on characterizing the decision policies, with the action policies being approximated via hand-engineered, deterministic functions of the traffic state. We instead focus on learning the action policies and exploiting them for interactive planning. We simplify the problem by assuming that an upstream decision-making module (similar to the module in [21]) has already determined whether to remain on the current lane or perform an autonomous lane change to one of the adjacent lanes.

Due to the large state, observation and action spaces, obtaining an exact solution to the integrals in (1) is computationally intractable. We can instead sample state transitions rather than considering all possible state transitions. We estimate the next state of a vehicle by drawing samples from its action distributions. Then, given the vehicle's sampled action at each step of the planning horizon, we use the following discrete-time equations to propagate the vehicle's state forward in time,

$$\mathrm{x}_v^{t+1} \approx \mathrm{x}_v^t + \dot{\mathrm{x}}_v^t \Delta t + \frac{1}{2}\ddot{\mathrm{x}}_v^t \Delta t^2 \tag{3}$$

$$\dot{\mathrm{x}}_v^{t+1} \approx \dot{\mathrm{x}}_v^t + \ddot{\mathrm{x}}_v^t \Delta t \tag{4}$$

$$\mathrm{y}_v^{t+1} \approx \mathrm{y}_v^t + \dot{\mathrm{y}}_v^t \Delta t \tag{5}$$

where x_v^t and y_v^t are the vehicle's longitudinal and lateral positions, respectively, $\ddot{\mathrm{x}}_v^t$ and $\dot{\mathrm{y}}_v^t$ are the longitudinal and lateral components of the vehicle's action, respectively, $\dot{\mathrm{x}}_v^t$ is the vehicle's longitudinal speed and Δt is the time step size.

B. ACTION POLICY

The action policy in (2) is referred to as a stochastic policy [30], which is a mapping to a distribution over driver action a_v^t. A stochastic policy is suited for representing the driving behavior of humans since there is not a "correct" way to act at any given moment, but rather there is a range of plausible, context-dependent actions a driver can take. We use deep neural networks parameterized by a parameter vector θ to simultaneously learn the action policy of the agent and all other drivers whose future actions we aim to predict.

Ideally, we would like to consider vehicles' past action, state, and observation history, as past motion often provides useful cues that can inform future predictions. Furthermore, changes in driver actions from one step to the next are at times insignificant or random, making it more difficult to learn the potential causes of actions without any temporal context. In Section VI-D, we assess the utility of motion history to policy performance with respect to prediction accuracy.

We also prefer a policy that characterizes the distributions over drivers' action plans as opposed to their instantaneous actions. The rationale for this is twofold. First, the instantaneous driver actions often express the drivers' short-term, reactive responses, while a plan can capture the anticipatory aspects of their behavior (i.e., the immediate driver actions reflecting their expectation of future). Second, conditioning the policy on drivers' interaction history while training it to predict the instantaneous driver actions is likely to yield a policy that performs poorly in practice; the policy would become prone to merely extrapolating between consecutive actions instead of attempting to learn their underlying cause.[1] A multi-step prediction method by contrast would force the policy to find correlation patterns in data that are useful for the prediction task rather than exploiting local, spurious data correlations.

Let $X_v = (\tilde{a}_v^{0:t}, x_v^{0:t}, z_v^{0:t})$ denote the motion history from the view point of vehicle v, where $\tilde{a}_v^{0:t}$ is the joint action history of the vehicle v and all other vehicles observed by v. Our goal is to obtain the conditional distribution of an action plan τ_v,

$$p(\tau_v|X_v;\theta) = \prod_{i=1}^{T} p(a_v^{t+i}|\tilde{a}_v^{t:t+i-1}, X_v;\theta) \tag{6}$$

where T is the number of time steps in the look-ahead horizon. Besides $\tilde{a}_v^{0:t}$, we follow [14] in conditioning the policy on the joint future actions $\tilde{a}_v^{t:t+i-1}$ to account for the potential presence of response dynamics, i.e., one vehicle's predicted response to the other vehicles' predicted actions. In the rest of the paper, we refer to the conditioning terms in (6) as contextual information and denote it as $c_v^{t+i} = (\tilde{a}_v^{t:t+i-1}, X_v)$. For notational simplicity, we drop the subscript v and the time superscript when not reasoning about a specific vehicle or time step.

[1] Previous studies have faced this issue, whereby the neural-network-based policy learned to "cheat" by extrapolating from the past rather than learning the underlying causes of driver behavior [31], [32].

C. CONTEXTUAL INFORMATION

The contextual information carries the available cues about how a driver might act in the future. It is common to assume access to the local observation z_v and the joint vehicle action $\tilde{a}_v$ when modeling the interactions between one autonomous agent and one human driver [13], [14], i.e., the idea that the agent can "see" the environment from the viewpoint of the other driver. However, we argue that unless vehicles can explicitly communicate their current state and plan with each other, this assumption does not hold in multi-vehicle, interactive scenarios. Consider the example of inferring $p(a_{v3}^{t+i}|c_{v3}^{t+i})$ in the scenario depicted in Fig. 3. The behavior of v_3 may have been influenced by a preceding vehicle, whose behavior may in turn have been influenced by another vehicle. In congested traffic, this recursive dependence can repeat for an arbitrary number of vehicles. Apart from the computational challenge of reasoning about such chains of influence, the agent cannot access the necessary information due to occlusions and its limited perceptual field. To this end, we consider a more realistic z_v and $\tilde{a}_v$ from the agent's perspective:

1) The agent can observe the state of vehicles in its lane and its adjacent lanes that are within a ± 70 m perception range.
2) Vehicles of interest are the immediate vehicles that are considered relevant for planning (for example see vehicles v_1, v_2 and v_3 in Fig. 3).[2]
3) The agent e can only access its own observation z_e, and $\tilde{a}_v$ only contains the joint action of vehicles observed by e. In other words, we do not assume that the agent can perceive the traffic environment from the standpoint of another vehicle.

Inevitably, the defined z_v and $\tilde{a}_v$ lead to a loss of contextual information; we aim to enrich the accessible context by taking into account the vehicles' past motion in our modeling approach. Furthermore, in real-world driving, sensor error and occlusions must also be accounted for in the planning framework. Future research may consider a high-fidelity perception model that more accurately reflects the perceptual field of the agent and other neighboring drivers from the agent's standpoint [33].

IV. METHODOLOGY

A. PLAN CONSTRUCTION

1) SPLINE-BASED PLAN REPRESENTATION

A drawback of a stochastic policy is that trajectories formed from sequences of sampled actions are characterized by jaggedness, potentially resulting in sharp action responses from the agent. To form smooth trajectories, we use the actions sampled at fixed time intervals of δ_t as control points, and interpolate between the points via cubic splines. Each spline segment

$$s_n : [t_n, t_{n+1}] \mapsto \tau^{t_n:t_{n+1}} \tag{7}$$

[2]In this work, we assume the number of vehicles V relevant for planning is known, for simplicity—we leave the consideration of an arbitrary number of vehicles to future work.

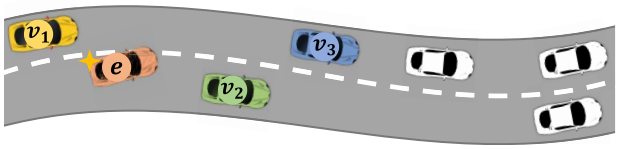


FIGURE 3. Example scenario showing the available information to the agent when performing a lane-change maneuver. The agent is denoted as e and the relevant vehicles whose future motion we want to predict are considered to be v_1, v_2 and v_3. Other vehicles (white) are not considered for prediction.

is a cubic polynomial representing a vehicle plan for the time interval $[t_n, t_{n+1}]$, where $n \in [0, N)$ for a spline with N segments. The control points at times t_n and t_{n+1} fully characterize the spline segment s_n. Since two adjacent spline segments share the same control points, transitions between them are smooth and continuous. Additionally, when computing the polynomial coefficients, we use the time derivative of the agent's current plan at the first time step t_0 as a constraint to enforce smooth transitions between the actions of consecutive planning iterations. We note that trajectories may still violate dynamic constraints such as friction, bounded acceleration, or steering angle. The development of an algorithm for the online feasibility assessment and refinement of the generated trajectories is however beyond the scope of this paper.

2) PLAN EVALUATION

We seek a mathematical representation of the agent's driving goal for evaluating the agent's candidate plans. Naively selecting the plan with the highest likelihood according to the learned policy is not always the best choice; the most likely plan may direct the agent to an unsafe state in the case that critical contextual information go uncaptured [8]. As such, at every planning iteration, we select the plan τ_e^* that is an approximate solution to the following maximization problem:

$$\arg\max_{\tau_e \in \mathbb{R}^{2\times T}} \mathbb{E}\left[w_l \left(\prod_{n=1}^{N} p(\hat{a}_e^{t_n}|c_e^{t_n}) \right) - \sum_{i=1}^{T} \gamma^i J(x^{t+i}, a_e^{t+i}) \right] \tag{8}$$

where γ is the discount factor and $\hat{a}_e^{t_n}$ is an action sample drawn at time t_n. The former term is the plan likelihood, which is obtained by querying the probability density function over the action sequence. The latter term is a cost function to favor plans with the lowest safety and comfort cost, where the weight parameter w_l can be tuned to make a trade-off between the desire to select high-likelihood plans vs choosing plans that incur the lowest cost. In our implementation, the agent applies the first δ_t seconds of its chosen plan τ_e^* to move to the next state. Similar to other model predictive control methods, new plans are computed at the next planning iteration to respond to the changes in the dynamic environment. We describe our implementation in greater detail later in Section VI-A.

B. MODEL DESCRIPTION

The model architecture, depicted in Fig. 4, is composed of a history encoder and four autoregressive sequence

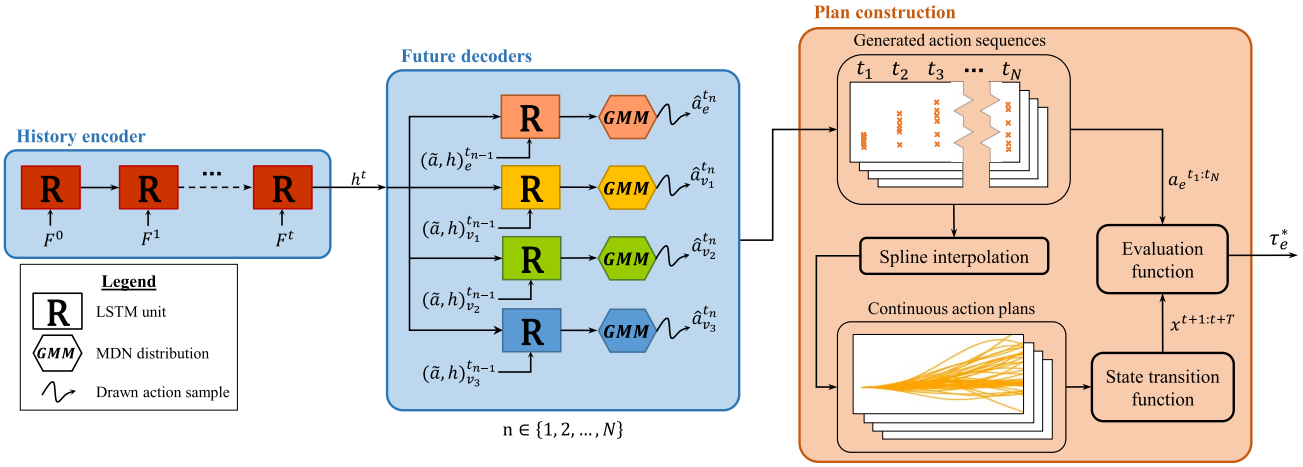


FIGURE 4. Overview of the proposed planning approach. The history encoder extracts cues from the vehicles' interaction history. Each decoder LSTM is followed by an MDN, which outputs a vector consisting of M_k, μ_k, ρ_k and σ_k^2 governing a mixture of K Gaussian action distributions. Subsequently, a set of samples are drawn from the action distributions, aggregated, and transformed into spline-based plans. Next, the vehicle states are propagated forward in time in order to obtain space-time trajectories over the prediction horizon. The trajectories are then used to estimate the expected utility of the agent's candidate plans. At last, the plan with the highest expected utility is selected for execution. Note: plan construction is only used at test time and not during model training.

decoders that facilitate the generation of action sequences. In this section, we describe the components of the model in greater detail.

1) MIXTURE DENSITY NETWORKS

We model driver actions via a mixture of Gaussian distributions,

$$p(a|c) = \sum_{k=1}^{K} M_k \, \mathcal{N}(a \mid \mu_k, \, \Sigma_k) \tag{9}$$

where M_k, μ_k and Σ_k are the context-dependent mixing coefficients, means, and covariance matrices, respectively, of the corresponding K mixture components. The neural network policy outputs the parameters of the distribution over a vehicle's longitudinal acceleration and lateral speed. To characterize this distribution, we have a mean vector and a covariance matrix,

$$\mathcal{N}\left(\begin{bmatrix}\mu_{\text{lon},k}\\ \mu_{\text{lat},k}\end{bmatrix}, \begin{bmatrix}\sigma^2_{\text{lon},k} & \rho_k\sigma_{\text{lon},k}\sigma_{\text{lat},k}\\ \rho_k\sigma_{\text{lon},k}\sigma_{\text{lat},k} & \sigma^2_{\text{lat},k}\end{bmatrix}\right) \tag{10}$$

where σ_k^2 and ρ_k denote the variance and the correlation parameters, respectively. We note that the mixing coefficients must sum to 1, the variances must remain positive and the correlation parameter is a value between -1 and 1. As in [11], we ensure that the parameter values remain valid by applying a softmax activation function to M_k, an exponential operator to σ_k^2 and a tanh activation function to ρ_k. The parameters of the *ith* mixture component can thus be obtained by,

$$M_i = \frac{\exp(\hat{M}_i)}{\sum_{k=1}^{K}\exp(\hat{M}_k)}, \quad M_i \in (0,1), \; \sum_{k=1}^{K} M_k = 1 \tag{11}$$

$$\sigma_i^2 = \exp(\hat{\sigma}_i^2), \quad \sigma_i^2 > 0 \tag{12}$$

$$\rho_i = \tanh(\hat{\rho}_i), \quad \rho_i \in (-1,1) \tag{13}$$

where the parameter values specified by the neural network are indexed by a hat ($\hat{\square}$).

2) ENCODER-DECODER RECURRENT NEURAL NETWORKS

The basic building blocks of the history encoder and the future decoders are the RNN variant Long Short-Term Memory (LSTM) [34], which have previously shown great success in various sequence learning tasks [15], [16], [35]. The input to the history encoder, as illustrated in Fig. 4, is a sequence of feature vectors $F = \{F^0, F^1, \ldots, F^t\}$ that together capture the motion history of the relevant vehicles from the agent's perspective (discussed later in Section V-C3). The encoder LSTMs encode the motion history into a fixed length state representation h^t.

Once the encoder has read the entire sequence, the future decoders take over to generate a set of action plans for the vehicles. At the first time step, each decoder LSTM receives h^t and the vehicles' joint action. Each decoder is then followed by an MDN to parameterize $p(a_v^{t_1}|c_v^{t_1})$. At a subsequent time step t_n, each decoder receives h^t, as well as the decoder's state $h_v^{t_{n-1}}$ and the joint action $\tilde{a}_v^{t_{n-1}}$ from the previous time step as input. We assign a separate decoder to each vehicle; however, we experiment with both a single decoder (i.e., all the vehicles share the same decoder) and the proposed multi-decoder architectural variant and provide empirical results in Section VI.

V. MODEL TRAINING

A. OPTIMIZATION PROCEDURE

Every pair of history X^d and action sequence $Y^d = a^{t_1:t_N}$ in the driving dataset provides a training example, where $Y^d = \{Y_v^d\}_{v\in V}$. The log probability $p(Y_v^d|X^d;\theta)$ can be computed as,

$$log\; p(Y_v^d|X^d;\theta) = \sum_{n=1}^{N} log\; p(a_v^{t_n}|\tilde{a}_v^{t_0:t_{n-1}}, X^d;\theta) \tag{14}$$

As mentioned in the preceding section, the latter terms in the above expression are represented by the LSTM state, i.e.,

$$log\ p(a_v^{t_n}|\tilde{a}_v^{t_0:t_{n-1}}, X^d; \theta) = log\ p(a_v^{t_n}|c_v^{t_n}; \theta) \tag{15}$$

where $c_v^{t_n} = (h^t, h_v^{t_{n-1}}, \tilde{a}_v^{t_{n-1}})$. The training procedure for optimizing the model parameters involves maximizing the log-likelihood of generating the correct action sequence for all the training pairs (X^d, Y^d):

$$\theta^* = \arg\max_{\theta} \sum_{(X^d,Y^d)} \sum_{v \in V} log\ p(Y_v^d|X^d; \theta) \tag{16}$$

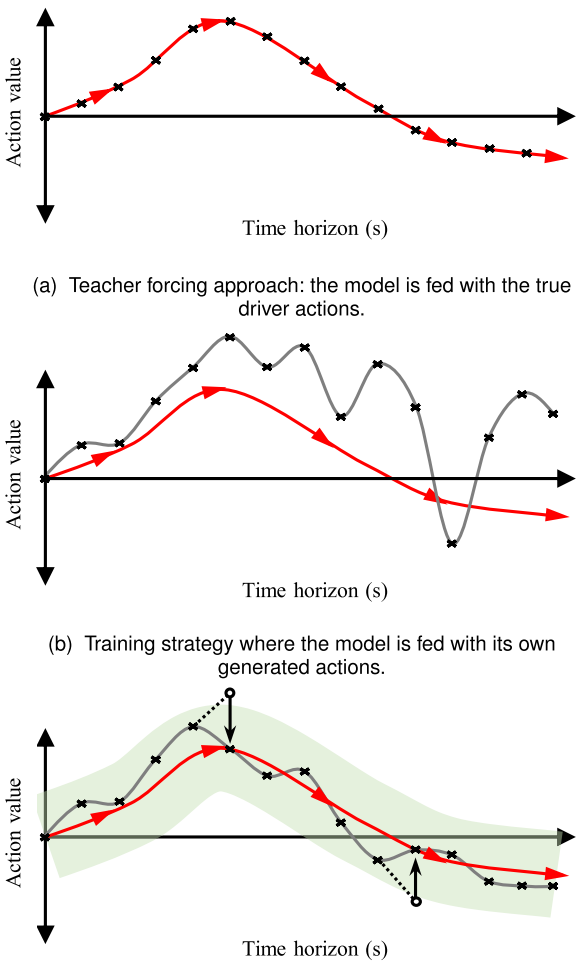
In practice, we use the mini-batch stochastic gradient descent algorithm with a batch size of 512 to train the model. We have implemented the model architecture shown in Fig. 4 using the TensorFlow library [36]. The architecture consists of a two-layer encoder and four decoder layers, each with 128 LSTM units. We ran experiments with 5 and 10 mixture MDNs, and found that for our dataset, both provide comparable validation error upon convergence. We use a total of 5 mixtures for the rest of our analyses. For training, we used the Adam optimizer [37] with default parameters and a learning rate of 0.001. In addition to hyperparameter annealing and regularization, we found that standardization of both input and target values lead to a faster convergence. Model training takes approximately 3 hours on an NVIDIA GeForce GTX 1050 and an 3.2 GHz Intel i5-6500 CPU.

B. MODEL TRAINING STRATEGY

The continuous nature of driving trajectories brings unique challenges to learning a model that yields a high prediction accuracy at test time. To illustrate, consider the task of computing $p(a_v^{t_n}|c_v^{t_n})$ when training the policy model, where $c_v^{t_n}$ contains the true driver action $a_v^{t_{n-1}}$. The subtle difference between $a_v^{t_n}$ and $a_v^{t_{n-1}}$ would result in the model paying a disproportionate attention to $a_v^{t_{n-1}}$, essentially failing to learn an informative history representation and pick up cues from $\tilde{a}_v^{t_1:t_{n-1}}$ for reasoning about response dynamics. The over-reliance of the model on $a_v^{t_{n-1}}$ would result in poor predictions at test time when the model no longer has access to true driver actions. One solution is immediately apparent: we can encourage the model to learn a useful history representation by feeding it with its own generated action $\hat{a}_v^{t_{n-1}}$ instead of the true driver action. However, feeding the model with $\hat{a}_v^{t_{n-1}}$ has its downside; the model would end up largely ignoring $\hat{a}_v^{t_{n-1}}$, particularly towards the end of the sequence, when errors in the generated actions accumulate so much that $\hat{a}_v^{t_{n-1}}$ conveys little information. An illustration of this phenomena is provided in Fig. 5b. Feeding the model with highly noisy actions amounts to assuming that a driver's sequence of actions are independent from each other:

$$p(a_v^{t_n}|c_v^{t_n}) \approx p(a_v^{t_n}|h^t, h_v^{t_{n-1}}) \tag{17}$$

The undue deviation from $p(a_v^{t_n}|c_v^{t_n})$ has two adverse effects on the model's performance. First, the stochastic nature of the policy coupled with the growing variance of $p(a_v^{t_n}|h^t, h_v^{t_{n-1}})$ with n would result in the model generating highly oscillating plans that do not replicate the characteristics of true driver trajectories. Second, the model would fail to produce an adequately diverse—possibly multimodal—set of prediction hypotheses. This is because unlike $c_v^{t_n}$, both h^t and $h_v^{t_{n-1}}$, as well as the internal structure of the LSTMs are entirely deterministic. A major source of randomness is found in the action distribution at one time step being conditioned on every previously model-generated action.



(a) Teacher forcing approach: the model is fed with the true driver actions.

(b) Training strategy where the model is fed with its own generated actions.

(c) Guided learning approach: the model is fed with the true driver actions only if its generated actions lie outside an error bound (shown in green).

FIGURE 5. Comparison of different training strategies. The red plot corresponds to the true driver plan and the crosses correspond to the actions that are sequentially fed to the model.

To summarize, we require that the trained model can at once: 1) discover useful history representations through the encoder; 2) learn to pick up any cues present in $\tilde{a}_v^{t_1:t_{n-1}}$ for reasoning about response dynamics; and 3) capture multimodality in driver action distributions. We next describe two techniques for addressing these competing requirements.

1) GUIDED LEARNING

A commonly used model training strategy, referred to as teacher forcing, involves feeding the model with ground truth

inputs throughout the duration of model training. As we discussed earlier in this section, always feeding the model with ground truth inputs carries several downsides that can hurt model performance. More advanced strategies for training RNN-based natural language models were proposed in [35]. Although we explored these techniques in our preliminary experiments, we found them to be ineffective for our task, yielding a model with a relatively poor predictive accuracy at test time.

We instead propose a simple approach for training multi-step time series models such as ours trained on continuous data. During training, we allow the model to use its own generated actions (i.e., longitudinal acceleration and lateral speed) as long as the action values are within an error bound of E,[3]

$$a_{\text{fed}} = \begin{cases} a_{\text{gen}}, & \text{if } |a_{\text{gen}} - a_{\text{true}}| \leq E \\ a_{\text{true}}, & \text{otherwise} \end{cases} \tag{18}$$

where a_{fed} is the action fed to the model, a_{true} is the ground truth action, a_{gen} is the action generated by the model and the error bound has the same unit as driver action. As illustrated in Fig. 5c, every time a generated action exits outside the error bound, the model is fed with the true action instead. This way, the model is forced to extract more useful cues from the interaction history and learn the action dependence without becoming over-reliant on having access to the true driver actions. We call this technique guided learning and provide empirical results in Table 2.

2) ACTION SUB-SAMPLING

The driving data we use to train the model comes at a recording frequency of 10 Hz. One way to increase the difference between each consecutive pair of actions in a target sequence is to sub-sample the sequence, which would increase the step size δ_t. In Section VI, we present the results of experiments performed to assess the influence of varying δ_t on the model's prediction accuracy.

C. TRAINING DATA

1) DATA PREPROCESSING

To train the model, we use the publicly available Next Generation Simulation (NGSIM) [38] dataset. The vehicle trajectories in NGSIM were recorded from CCTV cameras mounted at two highway locations: US 101 in Los Angeles, California, and I-80 interstate in the San Francisco Bay Area, California. Since the raw data are noisy, we have used the extended Kalman filter and the exponential smoothing algorithm from [39] to obtain better estimates of vehicle positions and velocities.

2) DATA EXTRACTION

A model trained on the entire dataset is prone to estimating concentrated probability densities at close to zero lateral speed as most highway driving involves drivers maintaining their lane. We overcome this problem by extracting an equal number of lane-change and lane-keeping episodes and training the model on the partitioned dataset instead. The method we use to estimate the beginning and the end of each lane-change episode is as follows. For each driver performing a lane-change maneuver (e.g., see Fig. 6), the initiation instant is considered to be the time at which the magnitude of the vehicle's lateral speed exceeds 0.1 m s^{-1}. The exact instant at which a driver decides to change lane is unknown and cannot be specified from the NGSIM data (we note that in general, the drivers' state of mind stays hidden). As an estimate, we take the start of an episode to be 2 s before the initiation instant, and the end of an episode to be the time at which the driver arrives at his or her desired lane. We extract a total of 6000 episodes, corresponding to roughly six hours of driving data.

A subset of the extracted trajectories of the drivers whose behaviors we want the agent learn to imitate (i.e., ego drivers) is visualized in Fig. 7. The occurrence frequencies of the ego vehicles' longitudinal speed and headway distance between the ego vehicles and the first vehicles preceding them in their target lane are shown in Fig. 8. We note that although we have partitioned the dataset to overcome the large imbalances among different behavior classes (i.e., keeping lane and

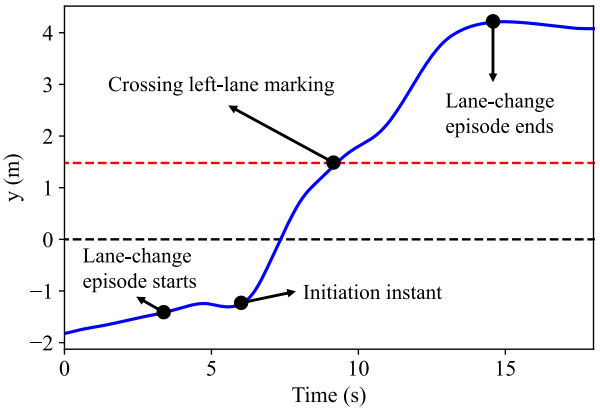


FIGURE 6. Example of a lateral motion profile of a driver changing lane from the NGSIM dataset. As an estimate, we take the start of lane-change episodes to be 2 s before their initiation instant.

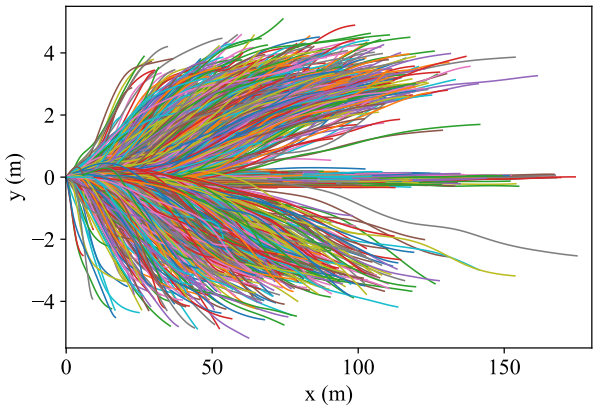


FIGURE 7. Visualization of 1800 driver trajectories extracted from the NGSIM dataset.

[3]Note that when $E = 0$ is equivalent to training a model via the teacher forcing strategy.

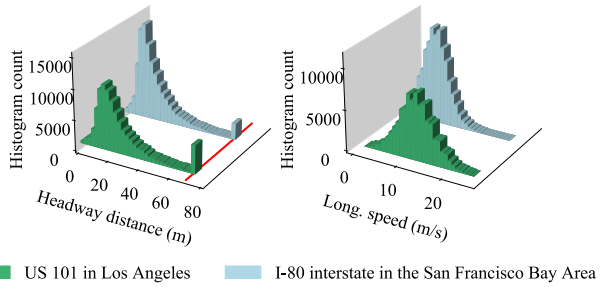


FIGURE 8. Left: Occurrence frequency of headway distance between the ego vehicles and the first vehicles preceding them in their target lane; the red line marks the agent's 70 m perception range. Right: Occurrence frequency of the ego vehicles' longitudinal speed. We use this dataset for learning the action policies, which are evaluated on a randomly selected, withheld (i.e., not used for model training) subset of the dataset. Data source: NGSIM [38].

TABLE 1. Extracted features.

Features	Description
Δx_v	longitudinal distance between vehicle $v \neq e$ and e
$\dot{x}_v$	longitudinal speed of vehicle v
$\ddot{x}_v$	longitudinal acceleration of vehicle v
$\dot{y}_v$	lateral speed of vehicle v, with positive values to the left and negative values to the right
Δy_e	lateral position of ego vehicle, with 0 m at the center of the lane, positive values to the left and negative values to the right
l_g	ego driver's target lane
f_b	Boolean indicating which surrounding vehicles are present

changing lane), the data remains highly diverse, containing driving conditions with a large range of vehicle speeds and headway distances. This is also reflected in the varied nature of the extracted driver trajectories.

3) TRAINING FEATURES

The motion history fed to the history encoder constitutes a sequence of feature vectors that represent the local driving context of ego vehicles. As well as the vehicles' joint action, state, and observation history, the feature vector contains two additional features. We add a Boolean feature f_b to indicate which surrounding vehicles are present within a ± 70 m range of ego vehicles. We set the state and action values of the missing vehicles to dummy values equal to their mean value across all the training examples. We also include the ego drivers' target lane, denoted as l_g, in their corresponding feature vectors,

$$l_g := \{-1,\ 0,\ 1\} \triangleq \{\text{right lane},\ \text{current lane},\ \text{left lane}\} \tag{19}$$

Together with l_g and f_b, each feature vector contains the 20 features shown in Table 1, where $v \in \{e, v_1, v_2, v_3\}$.

VI. EXPERIMENTS

We conduct experiments to: 1) examine the performance of the planner in simulation; 2) compare the prediction accuracy of the RNN encoder-decoder to two existing baselines; and 3) assess the influence of several key design choices on the model's performance.

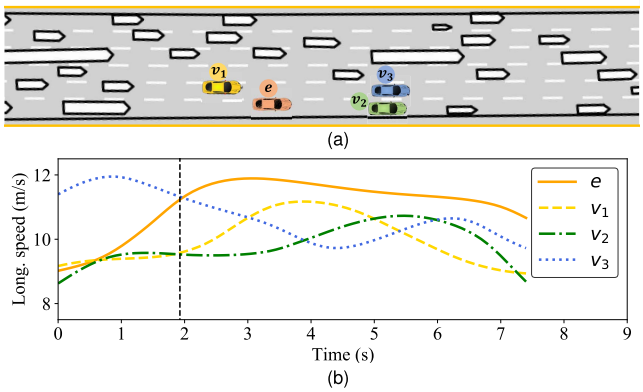


FIGURE 9. Lane-change scenario. In (a), the initial traffic condition is shown, where the traffic state is extracted from the NGSIM dataset. In (b), the speed profiles for the vehicles of interest are shown, with the dashed vertical line marking the instant at which we begin the simulation.

The method we use to evaluate the performance of the planner in simulation proceeds as follows:

1) For a given scenario, the road is populated with vehicles which are assigned with their true initial state from the NGSIM dataset.
2) The planner receives 2 s of the vehicles' past motion history to produce an agent plan for execution.
3) The first 0.3 s of the chosen plan is applied by the agent to move from its current state to the next state. For the human drivers, we instead apply their true actions from the NGSIM dataset.
4) A feature vector is obtained (see Section V-C3 for details) from the new traffic state to update the history with the latest contextual information.
5) Steps 2, 3 and 4 are repeated to propagate the traffic state forward in time.

It is important to note that while the state of human-driven vehicles are replays from the NGSIM dataset, the agent's states arise from its chosen control actions over time. This evaluation method aims at answering one particular question: *How would have the agent responded if it encountered a real traffic scenario?* We apply the method mentioned above to 600 randomly extracted scenarios from the test dataset, collecting statistics on the agent's visited states and collisions between the simulated agent and its neighboring vehicles.

We compare the prediction accuracy of the RNN encoder-decoder to a multilayer perceptron (MLP) policy [11] and an LSTM policy [12]. Both the LSTM and the MLP-based policies follow a step-wise prediction strategy, predicting driver actions for one step at a time as opposed to our method of generating action plans. The MLP is a five layer, fully connected neural network architecture that only receives the current traffic state as input to parameterize the distributions over driver actions. The LSTM is similar to the MLP in parameterizing action distributions, however, it additionally maintains an internal state that conditions the policy on 2 s of interaction history (i.e., same as the RNN encoder-decoder).

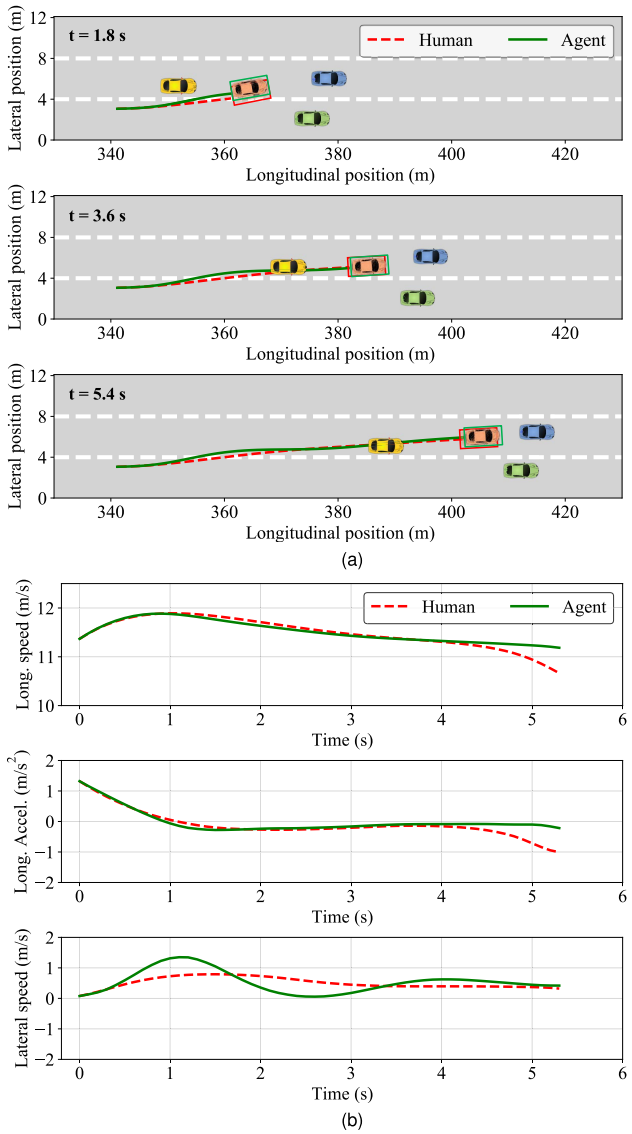


FIGURE 10. A simulated lane-change maneuver. (a) Time-lapse for the lane-change maneuver. The trajectories correspond to the paths taken by the human (red) and the agent (green), where the vehicles' positions are from the current instant. (b) Action and speed profiles belonging to the agent and the human.

A. AGENT PLAN SELECTION

Through the multi-stage process depicted in Fig. 4, the agent plan with the highest expected utility is selected and applied in a receding horizon fashion. In our experiments, the agent has a set of 100 candidate plans to choose from. For each candidate plan, 50 anticipated human plans are considered when approximating the expectation in (8).[4] In designing the agent's driving objective, our aim is to assign more value to high-likelihood plans while discouraging actions that result in discomfort and/or lead the agent to dangerous traffic states. Similar to [14], the cost term $J(x^{t+i}, a_e^{t+i}) = J_{c,v} + J_a$ consists of the running costs $J_{c,v}$ and J_a corresponding to collision avoidance and comfort,

$$J_{c,v} = 700 \cdot 1_{\{|\Delta x_v|<3 \wedge |\Delta y_v|<1\}} \cdot (3.5 - \sqrt{\Delta x_v^2 + \Delta y_v^2}) \quad (20)$$

$$J_a = \ddot{x}_e^2 + \dot{y}_e^2 \quad (21)$$

where $|\Delta x_v|$ and $|\Delta y_v|$ denote the longitudinal and lateral distance, respectively, between the agent and another vehicle v. The plan likelihoods are normalized across all the agent's candidate plans to values between [0, 1], making it easier to tune the weight parameter w_l in (8). In our experiments, we use a discount factor $\gamma = 0.9$ and a weight parameter $w_l = 4$.

Two aspects related to the agent's driving objective should be taken into account. First, since the action policy has already been conditioned on the agent's target lane l_g, there is no need to incorporate l_g in the objective. Second, the objective is a design choice—it is possible to alter the agent's behavior by accommodating other cost terms and adjusting their allocated weight. Admittedly though, designing a suitable objective is a challenging task for the system's designers that reflects potentially competing trade-offs. Unfortunately, there is currently no clear understanding of how to obtain a quantitative measure when evaluating plan quality in scenarios involving nuanced human-robot interactions. For instance, when performing a lane-change maneuver, the agent can maximize efficiency (potentially at a cost to traffic safety) by cutting in front of the preceding vehicle in the adjacent lane. Another option is to drive cautiously, although this may result in unnecessary delays. Alternatively, the agent can act in a more socially compliant manner, having learned such tendencies from human drivers.

B. QUALITATIVE EVALUATION

We next give a qualitative impression of the agent's behavior in a simulated lane-change scenario. The scenario, shown in Fig. 9a, involves merging into the gap between two vehicles in relatively congested traffic, with a starting headway distance of 4 m between the agent and the preceding vehicle in the agent's target lane. Fig. 9b shows the speed profiles belonging to the human drivers, with the dashed vertical line marking the instant at which we begin the simulation.

Fig. 10a visualizes the agent's trajectory as it performs the lane-change maneuver, which takes approximately 5.5 s to complete. The corresponding speed and action profiles of the agent and the lane changing human are shown in Fig. 10b. We note that the agent's emergent trajectory exhibits similar characteristics to the human trajectory. This mimicking behavior of the agent arises as a result of its objective assigning more value to plans with a higher relative likelihood.

A subset of the generated action plans for each vehicle is shown in Fig. 11, where the vertical blue lines mark the instant at which action generation begins and each column corresponds to the action plans for each vehicle. Overall, the model generated plans resemble those of the humans. For instance, in Fig. 11a, the model rightly anticipates that after about one second, vehicle v_1 (third column) is likely

[4]Our implementation has an average execution time of 465 ms per planning iteration on an NVIDIA GeForce GTX 1050 and an 3.2 GHz Intel i5-6500 CPU.

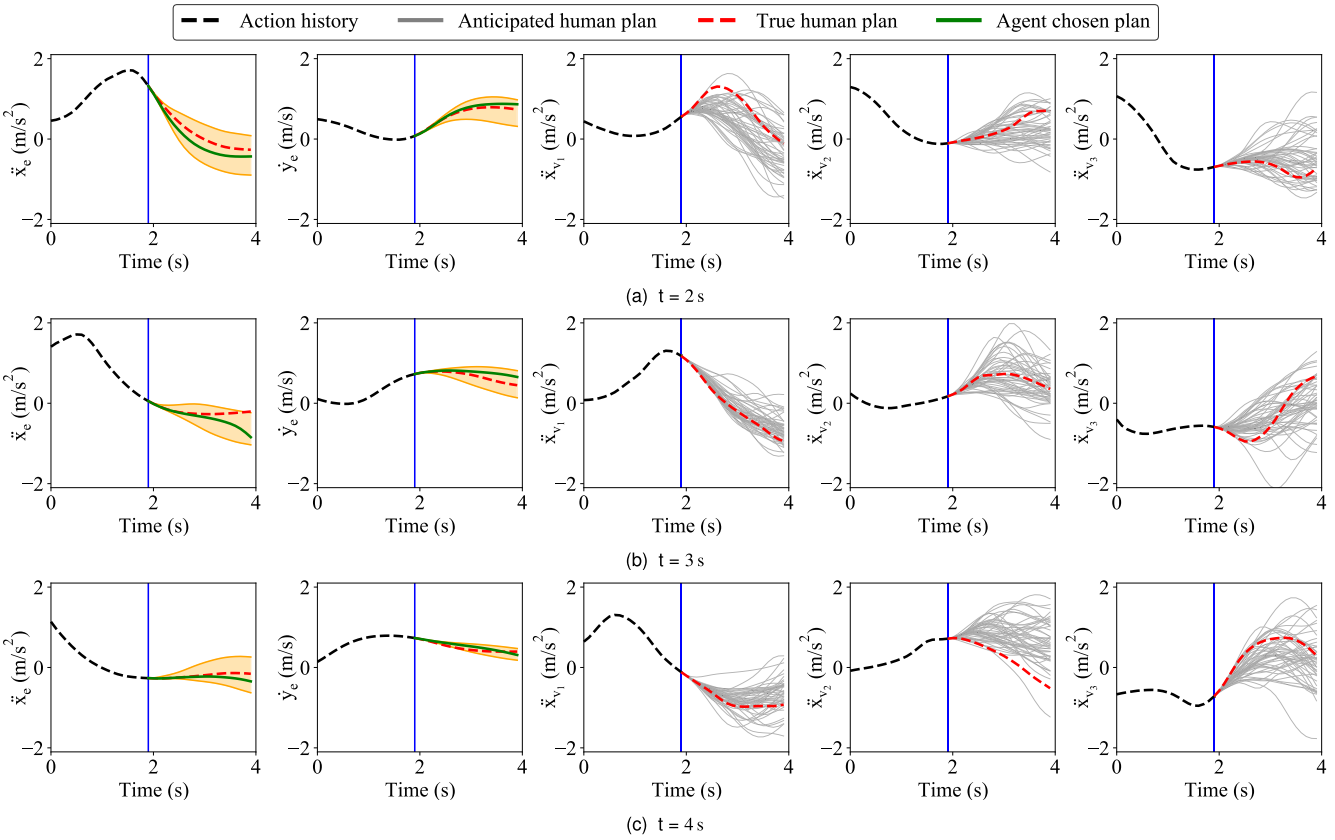


FIGURE 11. Plans (50 samples) for the time-lapse of an example lane-change maneuver. The columns show the plans for each vehicle and the rows correspond to snapshots at different instants. The agent's chosen plan is shown in green; the standard deviation corresponds to the range of candidate plans considered. The vertical blue lines mark the instant at which action generation begins.

to have a lower acceleration. This example is particularly interesting as the past actions of v_1 alone are not indicative of its future actions, suggesting that the model has used other traffic features to inform its predictions.

The model is also at times prone to making mistakes. An example of a relatively poor prediction can be seen in Fig. 11c, where for vehicle v_2, the estimated probability mass by the model does not closely align with the true driver plan. Although the performance of data-driven models such as ours often improve with more training examples, it should be noted that predictive accuracy may still suffer in situations where the accessible contextual information does not sufficiently inform future predictions. In practice, the continuous replanning step of the planner ensures that the agent can respond reasonably well under the uncertainty in model predictions. However, a more sophisticated approach would identify and handle the operating conditions in which the neural-network-based model becomes prone to erroneous predictions. Addressing this problem is currently an active research topic [40] but it is beyond the scope of this paper.

C. QUANTITATIVE EVALUATION

Fig. 12 shows two histograms that represent the occurrences of Euclidean distances between the vehicles that arise from the true human plans and those that emerge from the agent's planned actions. We find that the two distributions strongly

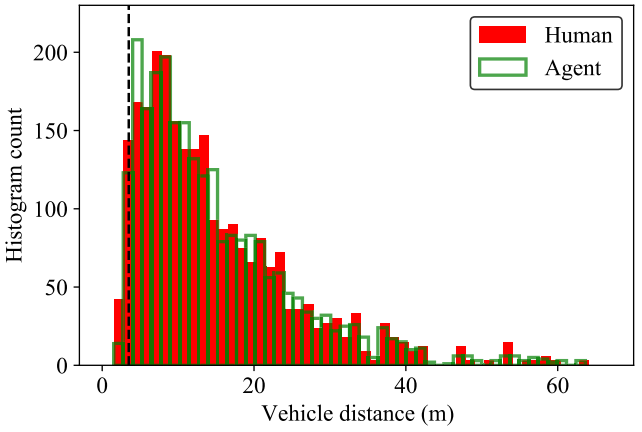


FIGURE 12. Empirical distribution over the occurrences of Euclidean distances between the agent and other vehicles, with the dashed vertical line marking the 3.5 m distance below which the agent's candidate plans incur a large negative cost.

overlap, implying that the agent exhibits a short-term driving behavior similar to human drivers. The closest distance between the agent and another vehicle is 1.53 m. Importantly, there were no collisions in any of the 600 scenarios extracted from the test dataset. This is to be expected since the agent's driving objective assigns a large collision avoidance penalty to plans that lead the agent to near-collision situations.

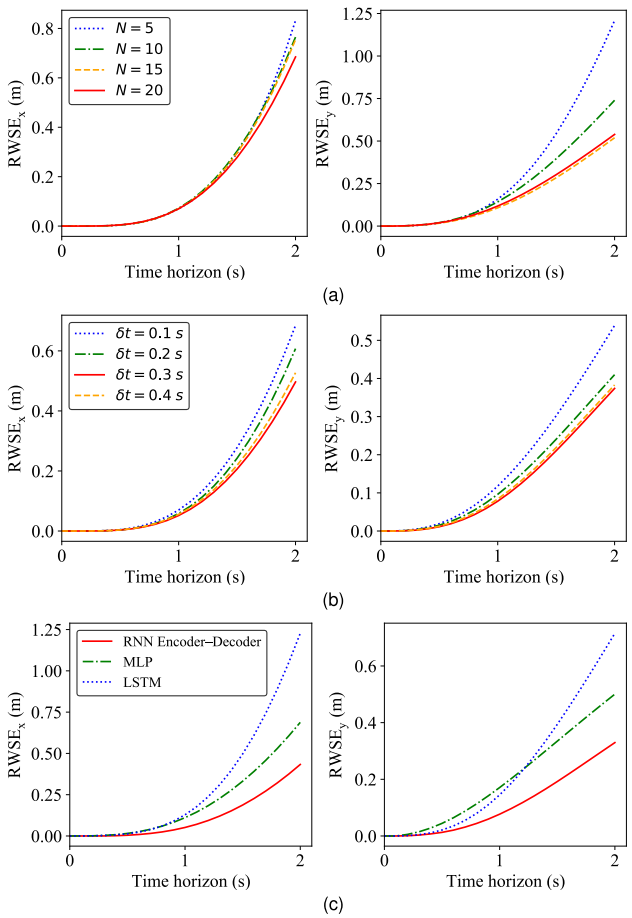


FIGURE 13. Comparison of prediction accuracy for (a) sequence lengths N (b) varying step sizes δ_t and (c) different baselines. The models shown in (a) were all trained with a step size of $\delta_t = 0.1$ s.

D. ABLATIONS

1) MODEL EVALUATION METRIC

As a model evaluation metric, we seek a similarity measure between the true human trajectories and the distribution of generated trajectories. We note that data likelihood, which is a measure for the goodness-of-fit of the model to data, is not a sufficient metric to assess model performance, as it only captures similarity at the level of individual actions. Following [10]–[12], we use the Root-Weighted Square Error (RWSE) as the evaluation metric, which captures the deviation of the model's probability mass from ground truth trajectories. The RWSE for m true trajectories and for the predicted variable r is:

$$\text{RWSE}^t = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\int_r p(r^t).(r^t_{(i)} - r^t)^2\, dr} \qquad (22)$$

where $r^t_{(i)}$ is the variable's true value at time t and $p(r^t)$ is the modeled density. Since the above expression cannot be evaluated analytically, it is approximated using Monte Carlo integration with n model generated trajectories for each of the m true trajectories,

$$\text{RWSE}^t = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(r^t_{(i)} - \hat{r}^t_{(i,j)})^2} \qquad (23)$$

where $\hat{r}^t_{(i,j)}$ is the predicted variable at time t and under the generated sample j. We use $m = 600$ randomly extracted trajectories from the test dataset and produce $n = 50$ generated trajectories.

2) ABLATIVE ANALYSIS

The RWSE values for the models trained with varying sequence lengths and step sizes are shown in Fig. 13a and Fig. 13b, respectively. In all these experiments, we have fixed the look-ahead horizon to $T = 20$ (with $\Delta t = 0.1s$), regardless of the action sequence length used for training. It is clear from both figures that increasing N and δ_t on average results in a lower prediction error, up to a limit, where there is no further improvement. This result demonstrates the benefits of learning a policy that directly maps to a sequence of actions and applying action sub-sampling as a model training technique.

Fig. 13c shows the RWSE values for the RNN encoder-decoder and the baselines. The RNN encoder-decoder achieves the best performance according to this metric. The MLP accumulates less error over time than the LSTM, indicating that a model fed with the current scene state as input outperforms one that receives vehicles' motion history. In several existing studies [10]–[12], it was also observed that neural-network-based models that are fed with motion history often perform worse than variants that only consider the current scene state for making predictions. This is perhaps surprising since intuitively, vehicles' past motion should enrich the available contextual information. The major difference between our RNN encoder-decoder and the LSTM is that, unlike the RNN encoder-decoder, the LSTM is trained to predict actions for one step at a time. For the reasons discussed in Section III-B, this step-wise prediction strategy carries several downsides which can hurt model performance.

The performance results for the model variants and the baselines are listed in Table 2. The RNN encoder-decoder models are all trained with $N = 7$ and $\delta_t = 0.3$ s. We note that a multi-decoder setup consistently outperforms a single decoder. This gain in prediction accuracy may be attributed to the improved representation power of a model with multiple decoders. The results also demonstrate that, as a training strategy, guided learning is effective in further lowering the model's prediction error. Comparing the RNN encoder-decoder model variants, it appears that modeling response dynamics yields a marginal performance gain. This suggests that for highway driving, the short-term future is reasonably predictable from motion history alone. Given that drivers are known to express anticipatory behaviors [41], one plausible explanation is that the history encoder has learned to pick up cues related to future interactions, making the consideration of response dynamics less valuable to the model.

TABLE 2. The prediction error for the presented model variants.

Model variants					RWSE_{xy} at 2 s							
Model	Action response	Guided learning	Error bound (E)	Decoder setup	x_e	y_e	x_{v_1}	y_{v_1}	x_{v_2}	y_{v_2}	x_{v_3}	y_{v_3}
MLP [11]	✗	✗	-	-	0.69	0.50	-	-	-	-	-	-
LSTM [12]	✗	✗	-	-	1.23	0.72	-	-	-	-	-	-
RNN encoder-decoder	✓	✗	-	single-decoder	0.57	0.38	0.56	0.23	0.61	0.24	0.61	0.29
RNN encoder-decoder	✗	✗	-	multi-decoder	0.52	0.38	0.49	0.22	0.56	0.23	0.56	0.27
RNN encoder-decoder	✓	✗	-	multi-decoder	0.50	0.37	0.48	0.22	0.55	0.23	0.54	0.26
RNN encoder-decoder	✓	✓	0.2	multi-decoder	0.50	0.34	0.48	0.21	0.56	0.24	0.53	0.25
RNN encoder-decoder	✓	✓	0.4	multi-decoder	**0.43**	**0.33**	**0.43**	0.21	**0.52**	**0.22**	**0.50**	**0.24**

Another contributing factor could be the nature of our training examples: they all come from scenarios in which lane-change maneuvers were successfully performed by human drivers. Consideration of response dynamics may play a more important role in scenarios involving a higher degree of ambiguity (e.g., negotiation of the right of way at complicated urban junctions).

VII. CONCLUSION

We have presented a planning framework for autonomous driving in scenarios involving interactions between an autonomous car and other human drivers. The planner takes as input the autonomous car's target lane and the surrounding vehicles' past motion history to output a locally optimal plan for execution. The key component of the planner is a predictive model of interaction dynamics learned from a dataset of many real driver interactions on US highways. Advantages of the proposed approach include (1) computationally inexpensive modeling of interactions among multiple vehicles, (2) the ability to capture the uncertainty in drivers' predicted motion, and (3) the generation of plans that mimic the driving behavior of humans. Additionally, we have explored several design choices and model learning strategies to produce a model that outperforms two existing methods in terms of the commonly used RWSE evaluation metric.

There are many potential avenues for extending this work. Before deploying the planner onto real-world roads, we will need to put more emphasis on its quantitative safety validation through high-fidelity traffic simulators [42] and scale our experiments to much larger and more diverse datasets. In addition, an important next step is to consider not only the uncertainty associated with the driving behavior of humans but also the uncertainty that emanates from the model's exposure to inputs that are underrepresented in the training data. Finally, a promising future advancement to our framework is to employ a more systematic approach for selecting the autonomous car's neighboring vehicles that are relevant for planning. This would allow the planner to scale efficiently to scenarios with an arbitrary number of interacting vehicles. The code associated with this paper is available at https://github.com/saArbabi/InteractivePlanning.

REFERENCES

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep. DOT HS 812 506, 2015.

[2] M. Ardelt, C. Coester, and N. Kaempchen, "Highly automated driving on freeways in real traffic using a probabilistic framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1576–1585, Dec. 2012.

[3] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, and T. Strauss, "Making Bertha drive—An autonomous journey on a historic route," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, Summer 2014.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Jun. 2016.

[5] A. Gray, Y. Gao, J. K. Hedrick, and F. Borrelli, "Robust predictive control for semi-autonomous vehicles with an uncertain driver model," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2013, pp. 208–213.

[6] S. Arbabi, S. Dixit, Z. Zheng, D. Oxtoby, A. Mouzakitis, and S. Fallah, "Lane-change initiation and planning approach for highly automated driving on freeways," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Nov. 2020, pp. 1–6.

[7] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, "A prediction-based reactive driving strategy for highly automated driving function on freeways," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 400–406.

[8] L. Fletcher, S. Teller, E. Olson, and D. Moore, "The MIT—Cornell collision and why it happened," *J. Field Robot.*, vol. 25, no. 10, pp. 775–807, 2008.

[9] DMV. (2020). *2020 Disengagement Report*. [Online]. Available: https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/

[10] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1289–1298, May 2017.

[11] D. Lenz, F. Diehl, M. T. Le, and A. Knoll, "Deep neural networks for Markovian interactive scene prediction in highway scenarios," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 685–692.

[12] J. Schulz, C. Hubmann, N. Morin, J. Lochner, and D. Burschka, "Learning interaction-aware probabilistic driver behavior models from urban scenarios," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1326–1333.

[13] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016.

[14] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human–robot interaction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 3104–3112.

[16] A. Graves, "Generating sequences with recurrent neural networks," 2013, *arXiv:1308.0850*.

[17] Q. Yang and H. N. Koutsopoulos, "A microscopic traffic simulator for evaluation of dynamic traffic management systems," *Transp. Res. C, Emerg. Technol.*, vol. 4, no. 3, pp. 113–129, Jun. 1996.

[18] M. Treiber and A. Kesting, "Car-following models based on driving strategies," in *Traffic Flow Dynamics*. Berlin, Germany: Springer, 2013, ch. 11, pp. 187–189.

[19] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant future prediction in dynamic scenes with interacting agents," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 336–345.

[20] Z. Sunberg and M. Kochenderfer, "Improving automated driving through POMDP planning with human internal states," 2020, *arXiv:2005.14549*.

[21] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Auton. Robots*, vol. 41, no. 6, pp. 1367–1382, 2017.

[22] A. Kesting and M. Treiber, "Calibrating car-following models by using trajectory data: Methodological study," *Transp. Res. Rec.*, vol. 2088, no. 1, pp. 148–156, Mar. 2008.
[23] M. Bahram, A. Lawitzky, J. Friedrichs, M. Aeberhard, and D. Wollherr, "A game-theoretic approach to replanning-aware interactive scene prediction and planning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3981–3992, Jun. 2016.
[24] J. Schulz, K. Hirsenkorn, J. Lochner, M. Werling, and D. Burschka, "Estimation of collective maneuvers through cooperative multi-agent planning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 624–631.
[25] 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.
[26] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2034–2039.
[27] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, vol. 1, 2000, p. 2.
[28] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," 2012, *arXiv:1206.4617*.
[29] N. Mullen, J. Charlton, A. Devlin, and M. Bedard, "Simulator validity: Behaviours observed on the simulator and on the road," in *Handbook of Driving Simulation for Engineering, Medicine and Psychology*. Boca Raton, FL, USA: CRC Press, 2011, pp. 1–18.
[30] T. Osa, J. Pajarinen, G. Neumann, J. Andrew Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," 2018, *arXiv:1811.06711*.
[31] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," 2018, *arXiv:1812.03079*.
[32] P. de Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11693–11704.
[33] A. Saha, O. Mendez, C. Russell, and R. Bowden, "Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5133–5139.
[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[35] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," 2015, *arXiv:1506.03099*.
[36] M. Abadi, P. Barham, J. Chen, Z. Chen, and A. Davis, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[38] (2017). *Next Generation Simulation (NGSIM) Traffic Analysis Tools*. [Online]. Available: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm
[39] T. Wheeler. (2018). *A Julia Package for Handling the Next Generation Simulation (NGSIM) Traffic Dataset*. [Online]. Available: https://github.com/sisl/NGSIM.jl
[40] J. Postels, F. Ferroni, H. Coskun, N. Navab, and F. Tombari, "Sampling-free epistemic uncertainty estimation using approximated variance propagation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2931–2940.
[41] M. Treiber, A. Kesting, and D. Helbing, "Delays, inaccuracies and anticipation in microscopic traffic models," *Phys. A, Statist. Mech. Appl.*, vol. 360, no. 1, pp. 71–88, 2004.
[42] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to simulate realistic multi-agent behaviors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10400–10409.

SALAR ARBABI received the M.Eng. degree in mechanical engineering from the University of Surrey, U.K., in 2018, where he is currently pursuing the Ph.D. degree with the Centre for Automotive Engineering. His research interest includes developing models of human behavior for creating more interactive decision-making agents that can act in human-dominated environments.

DAVIDE TAVERNINI received the M.Sc. degree in mechanical engineering and the Ph.D. degree in dynamics and design of mechanical systems from the University of Padova, Padua, Italy, in 2010 and 2014, respectively. During his Ph.D. degree, he was part of the Motorcycle Dynamics Research Group. He is currently a Lecturer in advanced vehicle engineering with the University of Surrey. His research interests include vehicle dynamics modeling, control and state estimation, and mostly applied to electric vehicles with multiple motors.

SABER FALLAH is currently the Director of the Connected Autonomous Vehicles Laboratory (CAV Laboratory), Department of Mechanical Engineering Sciences, University of Surrey, where he leads several research activities funded by the U.K. and European governments (e.g., EPSRC, Innovate U.K., H2020, and KTP) in collaboration with major companies active in autonomous vehicle and robot technologies. The CAV Laboratory provides a unique laboratory to design, develop and test the next generation of robotics and autonomous systems used for remote assembly and manufacture, highly automated transportation systems, and missions in hazardous environments including space. The CAV Laboratory also provides expertise in distributed control systems, AI and machine learning, and predictive optimization techniques. Prior to joining the University of Surrey, he was part of the Cross-Disciplinary Team in Green Intelligent Transportation Systems Research (University of Waterloo, Canada). His research interests include deep reinforcement learning, advanced control and prediction, and their application to autonomous robot systems.

RICHARD BOWDEN (Senior Member, IEEE) is currently a Professor in computer vision and machine learning with the University of Surrey, where he leads the Centre for Vision, Speech and Signal Processing, Cognitive Vision Group. His research interests include the use of computer vision to locate, track, and understand humans. He is a fellow of the Higher Education Academy and the International Association of Pattern Recognition (IAPR). He has previously held a Royal Society Leverhulme Trust Senior Research Fellowship and served as an Associate Editor for the *Image and Vision Computing* journal and IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.