EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments

Wenchao Ding, Lu Zhang, *Student Member, IEEE*, Jing Chen, and Shaojie Shen, *Member, IEEE*

***Abstract*—In this article, we present an efficient planning system for automated vehicles in highly interactive environments (EPSILON). EPSILON is an efficient interaction-aware planning system for automated driving, and is extensively validated in both simulation and real-world dense city traffic. It follows a hierarchical structure with an interactive behavior planning layer and an optimization-based motion planning layer. The behavior planning is formulated from a partially observable Markov decision process (POMDP), but is much more efficient than naively applying a POMDP to the decision-making problem. The key to efficiency is guided branching in both the action space and observation space, which decomposes the original problem into a limited number of closed-loop policy evaluations. Moreover, we introduce a new driver model with a safety mechanism to overcome the risk induced by the potential imperfectness of prior knowledge. For motion planning, we employ a spatio-temporal semantic corridor (SSC) to model the constraints posed by complex driving environments in a unified way. Based on the SSC, a safe and smooth trajectory is optimized, complying with the decision provided by the behavior planner. We validate our planning system in both simulations and real-world dense traffic, and the experimental results show that our EPSILON achieves human-like driving behaviors in highly interactive traffic flow smoothly and safely without being overconservative compared to the existing planning methods.**

***Index Terms*—Autonomous vehicle navigation, decision-making for automated driving, intelligent transportation systems, motion and path planning.**

I. Introduction

AUTONOMOUS driving is an emerging topic, both in industry and in the academic community. Planning, as one of the core components of autonomous driving systems, largely determines the intelligence and user experience of the system. Despite there being many industrial demos illustrating promising autonomy, few methodological details are provided to show how pain points in planning are systematically dealt with. On the other hand, the academic community has presented various planning systems which cover several pain points, such as uncertainty and interaction modeling [1]–[9]. However, most of these methods are only validated through simulation or well-annotated datasets, leaving a question as to whether they can work on a real autonomous vehicle. Planning on a real vehicle with closed-loop execution is far more challenging. Onboard planning requires dealing with an imperfect world and interacting with other traffic participants naturally. The imperfectness and uncertainty come from a wide range of aspects, such as occlusion in sensing, detection, and tracking noises, and unavoidable stochastic behaviors of other traffic participants, which are hard to reproduce in simulation. In this article, we aim at building a robust and socially compliant planning system which can handle the imperfect real world.

In this article, we present an efficient planning system for autonomous vehicles in highly interactive environments (EPSILON). We systematically investigate several pain points of planning for autonomous driving, such as the modeling of uncertainty and multiagent interaction, and attempt to achieve one small but concrete step toward automated driving in the real world. We validate the performance of EPSILON by conducting long-term closed-loop autonomous driving in complex driving environments. EPSILON follows a hierarchical structure which consists of a behavior planning layer and a motion planning layer, as in many previous methods [7], [9], [10]. In EPSILON, the role of behavior planning is to generate a preliminary decision which is represented by a sequence of states covering the planning horizon, while the role of motion planning is to wrap the sequence of states to a safe and smooth trajectory for closed-loop execution.

There is an extensive literature on behavior planning for automated vehicles. Previous works [7]–[14] focused on a pure geometry perspective, and adopted various rule-based or search-based techniques, which typically assume a behavior/trajectory prediction module is equipped on the system. The prediction is conducted independent of planning, and when the prediction is provided, the planner plans a trajectory with a sufficiently large safety margin. However, it is impossible to yield a perfect

Manuscript received December 4, 2020; revised May 1, 2021; accepted July 11, 2021. Date of publication September 1, 2021; date of current version April 5, 2022. This work was supported in part by the Hong Kong Ph.D. Fellowship Scheme, in part by the HKUST-DJI Joint Innovation Laboratory, and in part by the HKUST Institutional Fund. This paper was recommended for publication by Associate Editor R. Vasudevan and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Wenchao Ding and Lu Zhang contributed equally to this work.) (Corresponding author: Lu Zhang.)*

Wenchao Ding is with the Huawei Technology Company Ltd., Shanghai 200122, China, Hong Kong University of Science and Technology, Hong Kong (e-mail: wdingae@connect.ust.hk).

Lu Zhang and Shaojie Shen are with the Department of Electronic, and Computer Engineering, also with the Hong Kong University of Science and Technology, Hong Kong (e-mail: lzhangbz@connect.ust.hk; eeshaojie@ust.hk).

Jing Chen is with the DJI Technology Company Ltd., Shenzhen 510810, China (e-mail: jchenbr@connect.ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3104254.

Digital Object Identifier 10.1109/TRO.2021.3104254

prediction due to the stochastic nature of other traffic participants and onboard perception noise. Moreover, with such independent prediction, the mutual influence of the ego vehicle's and other vehicles' future motion cannot be modeled. In cooperative and interactive scenarios, such as gap merging, where this mutual influence is essential, coupled prediction, and planning is more promising [15]. Partially observable Markov decision process (POMDP) [16] provides a mathematically rigorous form of modeling uncertainties and multiagent interactions, while suffering from prohibitively high computational complexity. Several attempts [17], [18] have been made to accelerate the problem solving but are still not efficient enough for driving in complex environments [19], [20]. In this article, we propose a guided branching technique to focus the exploration using domain knowledge, which is much more efficient and can work in highly dynamic city traffic.

Motion planning aims at generating a safe and smooth trajectory which faithfully follows the decision provided by the behavior layer. Our motion planning layer is adapted from our previous work [1], which follows an optimization-based scheme. All the constraints posed by complex semantics are encoded in a unified way using a spatio-temporal semantic corridor (SSC). The piecewise Bézier curve is adopted as trajectory parameterization for its convex hull and hodograph properties which can enforce safety and dynamical feasibility for the entire trajectory. Benefiting from our optimization formulation, our motion planner can generate a safe and smooth trajectory as well as fit the preliminary behavior plan closely, significantly enhances the consistency of EPSILON.

The behavior planning layer of EPSILON was originally presented in our previous research [21]. In [21], although the interaction among traffic participants is captured using multiagent forward simulation, it generally assumes other participants are *rational*. The rationality is reflected in the predefined multiagent integration model. However, in real-world city driving, we find traffic participants are often *noisily rational*, especially in cities where the driving style is always aggressive. In this article, we extend our previous work by integrating a more flexible interactive forward simulation model with a safety mechanism, which better guarantees safety, even when encountering overaggressive traffic participants. The motion planning module of EPSILON was originally proposed in [1]. We advance the motion planner to incorporate multiagent interaction and unify behavior planning and motion planning. Moreover, we bring the newly designed system, EPSILON, into real-world dense traffic, while [1] and [21] are limited to only the simulation environment. We summarize our contributions as follows.

1) We present an efficient planning system by extending and tightly integrating our previous behavior planner [21] and motion planner [1]. The proposed new system deals with several pain points of automated driving systematically, including efficient interaction and uncertainty handling, improved consistency, and real-time implementation.
2) We enhance the robustness of the behavior planning layer by introducing a new forward simulation model, which better guarantees safety, even when encountering overaggressive or uncooperative traffic participants.
3) Moving beyond validation using only simulation and datasets, we extensively validate our system on a real automated vehicle in dense city traffic without an HD-map and purely relying on onboard sensor suites.
4) We release the complete decision-making and motion planning systems as open-source packages[1] to the research community.

The remainder of this article is organized as follows. The relevant literature is discussed in Section II. An overview of EPSILON is provided in Section III. The problem is formulated in Section IV. The behavior planning module is elaborated in Section V while the motion planning method is presented in Section VI. Implementation details are given in Section VII. Systematic comparisons and real-world experiments are illustrated in Section VIII. Finally, Section IX concludes this article.

II. Related Work

Behavior Planning for Automated Vehicles: A significant number of works on behavior planning[2] for automated vehicles have been published in recent years [22], and many of them consider the problem from a geometric perspective, namely, finding preliminary geometric collision-free paths/trajectories for the controlled vehicle. State-machine-based [10]–[12] behavior planning, which employs handcrafted rules for different driving conditions, was popular at the early stage. In this approach, based on the output state and action, reference paths can be extracted and fed to the motion planning layer. However, due to the complexity of real-world driving, continuous engineering, and tuning efforts are required to maintain the state machine. For this reason, later methods turned to a more generic formulation for behavior planning, such as search-based methods [7]–[9], [13], [14]. For example, Hubmann *et al.* [13] used A* graph search on a state lattice with static and dynamic events encoded as a cost map. The methods which adopt a purely geometric reasoning typically assume a deterministic trajectory prediction of other traffic participants is provided for the purpose of collision-checking. However, in real-world driving, prediction should be modeled from a probabilistic perspective due to the multimodal and uncertain nature of the prediction problem. Moreover, different future actions of the controlled vehicle may result in different future situations, which is also not modeled under purely geometric reasoning. In this article, we tackle the behavior planning problem from an interaction-aware perspective, which fundamentally addresses the above issues.

There is extensive literature on behavior planning from an interactive multiagent perspective, and many of the methods are formulated as POMDPs [16]. The POMDP is mathematically rigorous and outlines a principled way to solve the problem of planning under uncertainty. Due to the *curse of dimensionality*, the POMDP quickly becomes computationally intractable when the problem size scales. To overcome this issue, many online POMDP solvers have been proposed to accelerate the problem solving, such as those in [17], [18], [23], and [24]. Leveraging the

[1] https://github.com/HKUST-Aerial-Robotics/EPSILON

[2] In this article, the terms "behavior planning" and "decision-making" are used interchangeably.

latest advances in POMDP solvers, several works [2], [19], [20], [25] have applied POMDPs to behavior planning for automated vehicles. However, most conduct validation only in simulation with particular scenarios, such as merging or intersections, except [20], which conducts onboard experiments in a crowd. Nevertheless, the authors in [20] only deals with a tailored 1D problem, namely, the speed optimization problem under a given path but yielding a limited efficiency (around 3 to 5 Hz), which may be inadequate for solving a full decision-making problem in highly dynamic driving environments. One solution to handle the computational issue is to simplify the original POMDP using domain knowledge. One representative method, multiple policy decision-making (MPDM), was proposed in [26] and [27]. It first designs a set of semantic-level policies such as lane-nominal and yield, then conducts closed-loop forward simulation to evaluate the policies. MPDM converts the original problem into selecting the best policy with highest reward in a limited number of policies which is computationally efficient and easy to deploy. However, MPDM oversimplifies the action for the ego vehicle: Each candidate policy contains only one semantic-level action with a rather long duration, making MPDM a single-stage MDP, which restricts the manoeuvrability and the level of "intelligence" of the decision. In this article, we retain the multiple layer structure of the POMDP and utilize domain knowledge as well as guided branching to achieve higher efficiency while retaining flexibility. Moreover, our method is validated against real-world environments with rich interactions and potentially stochastic traffic participants.

Motion Planning for Automated Vehicles: Motion planning for automated vehicles is relatively mature compared to behavior planning [28], [29]. Typical routines for motion planning include search-based methods based on the state lattice [30]–[32] or motion primitives [33], optimization-based methods [3]–[5], [34], [35], or the combination of these two [6], [36]. Besides, works focus on higher safety guarantee that utilizes the reachability analysis to generate a trajectory under the constraints of reachable set that considers the uncertainties of the vehicle models [37], [38], limited sensing [39], and interactions [40]. Our previous method [1] belongs to the category of optimization-based methods. The key feature is that it proposes a unified input representation, namely, the SSC structure, to wrap complex constraints posed by various semantic elements. This input representation saves considerable effort in engineering and tuning the constraints. However, like most motion planning methods, our previous approach still requires a trajectory prediction module and does not capture the multiagent interaction, which poses inconsistency when being used with interaction-aware behavior planning methods. In this article, since future anticipation is coupled inside our behavior planning, we can directly feed the decision together with the conditional anticipation of other traffic participants to the motion planning layer, which makes the motion planner in [1] amenable to modeling interaction.

III. System Overview

The structure of the proposed planning system is shown in Fig. 1. The planning system is built on top of the environment understanding and closed-loop execution modules. There

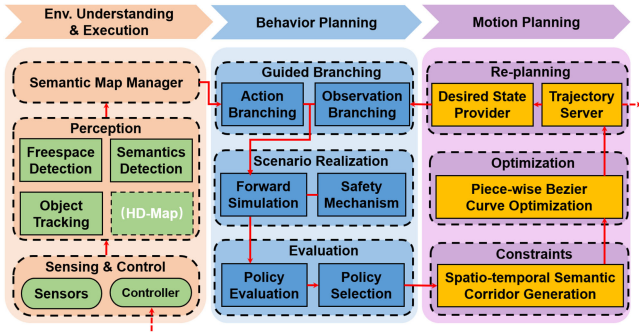


Fig. 1. Diagram of EPSILON together with environmental understanding and execution modules. HD-map is optional for EPSILON.

are several modules in the perception pipeline: Freespace detection [41], semantics detection (e.g., lane detection, traffic light detection, etc.), object detection, and tracking. Note that HD-map is *optional* for our planning system. For onboard experiments on a real vehicle, we do not use HD-map and purely rely on online lane detections. The output of perception is synchronized and fed to a semantic map manager module which is responsible for organizing the data structures and providing querying interfaces for planning modules. The semantic map is updated at 20 Hz and includes an occupancy grid for static obstacles, multiple tracklets for dynamic obstacles, and road structures for lanes and traffic semantics.

As mentioned previously, EPSILON follows a two-layer hierarchical structure, comprising a behavior planning layer and a motion planning layer. Note that there is no additional trajectory prediction module like in [1] since the trajectory predictor is coupled inside the behavior planner. At a high level, our behavior planner consists of three processes, namely, guided branching, scenario realization, and evaluation. Essentially, guided branching is responsible for expanding action sequences according to predefined policies for the controlled vehicle and reasoning about the possible intentions of other traffic participants. By incorporating a particular ego action sequence with a particular intention combination of other traffic participants, we form a *scenario*. During scenario realization, we use closed-loop multiagent forward simulation to realize the scenario step by step. Different from our previous work [21], we extend the forward simulation model into a more flexible driving model to achieve safe and human-like driving behavior, even in complex real-world traffic with quantities of noisily rational participants.

The motion planning layer basically follows our previous work [1]. First, static, dynamic obstacles, and constraints posed by environment semantics are modeled using a SSC around the initial guess provided by the behavioral layer. Then a piece-wise Bèzier curve is optimized with respect to the corridor. Using its convex hull property and hodograph property, the *entire* trajectory can be guaranteed to be safe and dynamically feasible. The generated trajectory is fed to a trajectory server for replanning scheduling, and the desired state for replanning can be queried from the trajectory server. The trajectory server is also responsible for sending out control commands to a vehicle controller to close the execution loop.

The design frequency for both layers is 20 Hz. In practice, the two layers can be assembled as a pipeline, which will increase the throughput of the whole planning system.

IV. Problem Formulation

We first clarify the problem setup of the decision-making and motion planning process of the ego agent. Let $\mathcal{E}_t$ denote the ego-centric local environment at time t, including road structures, traffic signals, and the occupancy grid for static obstacles. Define x_t^i as the state of the vehicle $i \in V$ at time t, and without loss of generality, $i = 0$ is reserved for the ego vehicle. As a notational convenience, subscript absence denotes *all* time steps, and superscript absence denotes all agents. For example, x_t denotes the states of all vehicles at time t, while x^i denotes the states at all time instances for vehicle i. For the planning cycle at time t, the ego agent receives the observation z_t, and uses it to estimate the real state x_t for planning. We define the input of the behavior planning layer as $\langle z_t, \mathcal{E}_t \rangle$, and the output is a *decision* $\mathcal{D}_t$, which is parameterized by a sequence of discrete states $\mathcal{D}_t := [x_{t+1}, x_{t+2}, \ldots, x_{t+H}]$ for all agents, where H denotes the planning horizon. The input to the motion planning layer is $\langle \mathcal{D}_t, \mathcal{E}_t \rangle$, while the output is a smooth and safe trajectory (typically parameterized by splines) for controlling the ego vehicle. The reason that we decouple the behavior planning and motion planning layer is for higher efficiency. After decoupling, the behavior planner only needs to reason about the future scenario at a relatively coarse resolution, while the motion planning layer works in the local solution space given the decision $\mathcal{D}_t$.

As mentioned in Section I, in the real world, the planning system always suffers from unknown interactive patterns among traffic participants. To address this issue, we model the uncertainty-aware behavior planning problem in the form of the POMDP, after which we formulate it into a multiagent setting. A POMDP model can be defined as a tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{Z}, T, O, R \rangle$, where $\mathcal{X}$, $\mathcal{A}$, and $\mathcal{Z}$ are the state space, action space, and observation space, respectively. The function $T(x_{t-1}, a_t, x_t) = p(x_t | x_{t-1}, a_t)$ is the probabilistic state transition model, while $O(x_t, z_t) = p(z_t | x_t)$ is the observation model. These two functions reflect the stochastic property of the motion model and uncertain sensing. $R(x_{t-1}, a_t)$ is a real-valued reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ for the agent taking action $a_t \in \mathcal{A}$ in state $x_{t-1} \in \mathcal{X}$. Since some states in real-world applications cannot be directly observed (e.g., hidden intentions or noisy measurements), the POMDP maintains the *belief* $b \in \mathcal{B}$, which is the probability distribution over $\mathcal{X}$. The belief can be updated after the agent leaves an initial belief b_{t-1}, and then takes an action a_t and receives an observation z_t. The resulting belief state can be inferred using Bayes' rule

$$\begin{aligned} b_t(x_t) &= p(x_t | z_t, a_t, b_{t-1}) \\ &= \eta O(x_t, z_t) \int_{x_{t-1} \in \mathcal{X}} T(x_{t-1}, a_t, x_t) b_{t-1}(x_{t-1}) dx_{t-1} \end{aligned} \tag{1}$$

where η is a normalizing factor. The POMDP is to find an optimal *policy* π^* which maps the belief state to an action $\pi : \mathcal{B} \to \mathcal{A}$, maximizing the expected total discounted reward over the planning horizon

$$\pi^* := \arg\max_{\pi} \mathbb{E}\left[\sum_{t=t_0}^{t_H} \gamma^{t-t_0} R(x_t, \pi(b_t)) \big| b_{t_0} \right]$$

where t_0 is the current planning time and $\gamma \in [0, 1]$ is a discount factor. For online POMDPs, starting from an initial belief b_{t_0} and expanding in action space $\mathcal{A}$ as well as observation space $\mathcal{Z}$ until a certain planning horizon t_H, a belief tree is built node by node. Then, an optimal policy is found by applying the Bellman equation on each internal node

$$\begin{aligned} V^*(b) &= \max_{a \in \mathcal{A}} Q^*(b, a) \\ &= \max_{a \in \mathcal{A}} \left\{ \int_{x \in \mathcal{X}} b(x) R(x, a) dx \right. \\ &\qquad \left. + \gamma \int_{z \in \mathcal{Z}} p(z | b, a) V^*\left(\tau(b, a, z)\right) dz \right\} \end{aligned}$$

where $V^*(b) = \int_{s \in \mathcal{S}} V^*(s) b(s) ds$ is the optimal utility function for the belief state, and $Q^*(b, a)$ describes the optimal value of the belief-action pair. For more details about solving POMDPs, we refer interested readers to [16] and [23].

Different from previous works [20], [25] that only use the single optimal action on the initial belief node as the final output, here, we extract a complete trace $\mathcal{S}_t = [b_t^*, a_{t+1}^*, z_{t+1}^*, b_{t+1}^*, \ldots, b_{t+H}^*]$ on the belief tree by applying

$$a_t^* = \arg\max_{a_t \in \mathcal{A}} Q^*(b_{t-1}, a_t)$$

$$z_t^* = \arg\max_{z_t \in \mathcal{Z}} p(z_t | b_{t-1}, a_t^*)$$

on action branches and observation branches recurrently. The resulting trace contains the optimal action a_t^* on each belief node b_{t-1}^*, and the most likely received observation z_t^* given a_t^*. The final decision $\mathcal{D}_t$ can be generated by simply applying $x_t = \arg\max_x b_t^*(x)$ for each step in $\mathcal{S}_t$. Compared to a single optimal action, $\mathcal{D}_t$ contains much more holistic information about the environment and future forecasting, which are essential for the following motion planner.

Considering the driving scenario at time t with N agents, the full state can be written as $x_t = \{x_t^0, x_t^1, \ldots, x_t^N\} \in \mathcal{X}$, wherein $x_t^i \in \mathcal{X}^i$ is the ith vehicle's state containing its position, velocity, acceleration, heading, and steering angle. Note that for surrounding vehicles, hidden states such as driving *intention* or *aggressiveness*, which cannot be directly observed, are also included in $x_t^i \forall i \neq 0$. In the driving scenario, the only thing we can control is the input signal (e.g., throttle/brake and steering) of the ego vehicle; in other words, we cannot directly determine the surrounding agents' actions. Therefore, we point out that the action of the original problem can be written as $a_t = a_t^0$, similarly, the action space $\mathcal{A}$ is equivalent to $\mathcal{A}^0$. The complete state transition model T can be represented as a joint distribution of the full state $p(x_t | x_{t-1}, a_t) = p(x_t^0, \ldots, x_t^N | x_{t-1}^0, \ldots, x_{t-1}^N, a_t^0)$ which is nontrivial to directly model. However, since most of the agents in real traffic, such as vehicle and bicycle, follow common traffic rules and physical principles (e.g., kinematics or dynamics), we

can simplify the formulation by making reasonable assumptions for the surrounding agents and converting it into a multiagent interactive model. We assign a probabilistic transition model with an action $a_t^i \in \mathcal{A}^i \forall i \neq 0$ for each surrounding agent and assume that the instantaneous transitions of each vehicle are independent, resulting in

$$p(x_t|x_{t-1}, a_t) \approx \underbrace{p(x_t^0|x_{t-1}^0, a_t^0)}_{\text{ego transition}} \prod_{i=1}^{N} \int_{\mathcal{A}^i} \underbrace{p(x_t^i|x_{t-1}^i, a_t^i)}_{i\text{-th agent's transition}} \underbrace{p(a_t^i|x_{t-1})}_{\text{driver model}} da_t^i \quad (2)$$

which separates the controlled vehicle from the surrounding agents. In the formulation, $p(x_t^i|x_{t-1}^i, a_t^i)$ is the *assumed* transition model of the other agents that reflects the agent's low-level kinematics and $p(a_t^i|x_{t-1})$ is the *assumed* driver model representing the high-level decision process of the other agents that provides appropriate control signals according to the driving context. In practice, the driver model can be user-defined or learned by data-driven approaches, and can be controlled by some latent states, such as intentions or aggressiveness, achieving diverse *maneuvers* or *driving styles*. Note that the ego action a_t^0 is generated by the predefined ego policy. The ego observation $z_t^0 = z_t$ contains the estimated position and velocities of other vehicles generated by the perception module. For the surrounding agents, we use the agents' poses as the origin and transform the ego observation z_t^0 into their coordinate frames as observation z_t^i. Since z_t^i is fully determined by the ego observation, we can get $p(z_t|x_t) = \prod_{i=0}^{N} p(z_t^i|x_t^i)$ assuming the observation processes are independent. Therefore, (1) can be factorized as

$$b_t(x_t) = \eta \cdot \overbrace{p(z_t^0|x_t^i) \int_{\mathcal{X}^0} p(x_t^0|x_{t-1}^0, a_t^0) b_{t-1}^0(x_{t-1}^0) dx_{t-1}^0}^{\text{belief update for ego agent}}$$

$$\overbrace{\prod_{i=1}^{N} p(z_t^i|x_t^i) \iint_{\mathcal{X}^i \mathcal{A}^i} p(x_t^i|x_{t-1}^i, a_t^i) p(a_t^i|x_{t-1}) b_{t-1}^i(x_{t-1}^i) da_t^i dx_{t-1}^i}^{\text{belief update for other agents}}. \quad (3)$$

We find that although the state transitions of each agent in each step are independent, the assumed driver models $p(a_t^i|x_{t-1})$ and observation models $p(z_t^i|x_t^i)$ leverage all agents' states and observations, making the belief update an interactive process. Note that the hidden states are estimated step by step during the belief update using observations, which is exactly the role of behavior prediction. As a consequence, the POMDP implicitly contains the prediction, indicating that planning and prediction are naturally coupled. It is also notable that methods which decouple prediction and planning are essentially simplifications of the original POMDP formulation.

By applying belief update recurrently, we can build the belief tree starting from the initial belief node and extract the final decision $\mathcal{D}_t$. However, the scale of the belief tree grows exponentially w.r.t. the tree depth, which is computationally intractable for real-time applications. To overcome this issue, in this article, we apply domain knowledge into the formulation to further simplify the problem of achieving real-time decision-making in a fast-changing driving environment while preserving the ability to handle interactions and uncertainties. Given the output of behavior planner $\mathcal{D}_t$, the motion planning layer aims to generate a safe and smooth trajectory to realize the high-level decision in a fine-grained fashion. Note that since our behavior layer is formulated in a multiagent setting, it naturally reasons about the future states for *all* the agents. Therefore, the role of the motion planning boils down to a local trajectory optimization problem, as shown in Fig. 2.

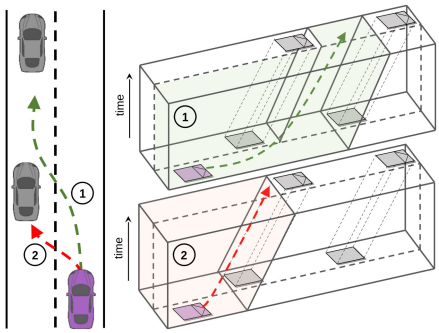


Fig. 2. Illustration of the relationship between behavior planning and motion planning defined in this article. The controlled vehicle (marked in *purple*) can accelerate and insert between two nearby vehicles (maneuver ①), or yield and follow the nearest vehicle (maneuver ②). Visualized in the spatio-temporal domain, the two maneuvers belong to two different homotopy classes. The behavior planning targets at discovering diverse tactical maneuvers, while the motion planning is for generating a local smooth and safe trajectory within the corresponding spatio-temporal space.

V. Efficient Behavior Planning

A. Motivating Example

It is notable that human drivers do not bear a fine-grained lattice in their mind. Instead, they tend to reason about *only a few* long-term *semantic-level actions* given common driving knowledge (e.g., lane keeping, yielding, overtaking, etc.) which makes the decision-making process extremely efficient. By utilizing semantic-level actions, we can directly sample long-term high-likelihood trajectories for action branching, as shown in Fig. 3. Another key issue of POMDPs is that the intention of other agents is not directly observable. The POMDP requires sampling over the intentions of other agents, which we refer to branching in the observation space. The number of required samples scales exponentially with respect to the number of agents, which is another source of the efficiency problem.

The motivation of guided branching is to mimic the decision-making process of human drivers and utilize domain knowledge for efficient exploration. In this article, we borrow the idea from MPDM [26], [27] and push our behavior planner one step forward in the following three aspects.

1) *Flexible Driving Policy*: We reduce the decision space of the original POMDP by guided branching in the action space. Compared to MPDM, the proposed method considers *multiple future decision points* rather than using

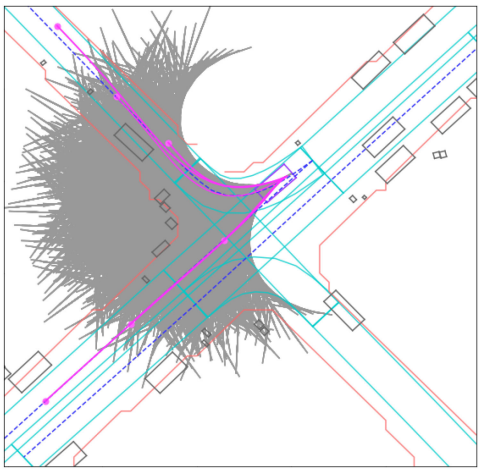


Fig. 3. Illustration of using semantic-level action to guide the exploration of the action space. Even with coarsely discretized control actions (e.g., three discretized longitudinal velocities and three lateral accelerations), the action space to be explored (*gray*) of a POMDP-based planner is huge during a multistep look-ahead search, while most of the space explored is of low likelihood. The situation is much worse when considering the multiagent setting. After incorporating semantic-level action (e.g., pursuing center-lines with different aggressiveness) using predefined controllers, the exploration (*purple*) can be guided using domain knowledge. Although much of the exploration is pruned, the resulting exploration still faithfully captures the potential high-level decisions.

a single semantic action in the whole planning horizon, thereby obtaining more flexible motions.

2) *Efficient Policy Evaluation*: Instead of directly sampling all possible combinations of agent intentions, which leads to exponential complexity, we propose a mechanism to pick out the potentially risky scenarios conditioned on the ego policy to reduce computational expense during the policy evaluation.
3) *Enhanced Risk Handling in Real Traffic*: The proposed planner provides an implementation of a new forward simulation model with a safety mechanism, which can reduce the risks caused by the inconsistency between real traffic participants and assumed models.

B. Guided Action Branching

To guide the exploration in the large space, we introduce semantic-level actions. An exemplary demonstration is provided in Fig. 3. Essentially, the semantic action is a high-level action which can be directly interpreted using human senses, such as *lane-change* and *deceleration*. The semantic action contains multiple small steps and can be executed in a closed-loop manner, it generates a primitive action (i.e., steering angle and longitudinal acceleration) at each step according to a predefined controller. Compared to the primitive action, employing semantic actions constrains the exploration within a higher likelihood region and can generate human-like motion naturally. Moreover, the duration of semantic actions can be much longer (up to several seconds), which can effectively reduce the height of the belief tree to a relatively small number while obtaining a large enough planning horizon.

Mathematically, the assumption above introduces an additional variable $\phi_t^i \in \Phi^i$ into the driver model, which represents the semantic action for agent i at time t, and Φ^i denotes a set of discretized predefined semantic-level actions for agent i. Note that the semantic action set can be different depending on the agent type. For example, the ego semantic set Φ^0 can be larger since more actions lead to more diverse and flexible driving behavior, while the size of $\Phi^{i\neq 0}$ for the other agents is designed to be relatively small due to the complexity concern. Moreover, even $\Phi^{i\neq 0}$ for different types of road-user (e.g., pedestrians, cyclists, and vehicles) can be different as well. At each step, a semantic action ϕ_t^i produces a primitive action a_t^i based on the previous *joint* state x_{t-1} according to a predefined controller $p(a_t^i|x_{t-1}, \phi_t^i)$. This reflects the fact that the mapping from semantic actions to primitive actions takes the surrounding driving context into account, which preserves the closed-loop property of the forward simulation.

To incorporate the semantic action, the joint belief update (1) can be represented as $b_t(x_t) = p(x_t|z_t, \phi_t, b_{t-1})$ by substituting the primitive action with the semantic action, and the joint state transition becomes $T(x_{t-1}, \phi_t, x_t)$. Here, we point out again that the only controllable element is the ego semantic action ($\phi_t = \phi_t^0$), then, we can expand it similar to (2) and obtain the new joint state transition

$$p(x_t|x_{t-1}, \phi_t) \approx \int_{\mathcal{A}^0} \underbrace{p(x_t^0|x_{t-1}^0, a_t^0)}_{\text{ego state transition}} \underbrace{p(a_t^0|x_{t-1}, \phi_t^0)}_{\text{ego controller}} da_t^0 \prod_{i=1}^{N} \int_{\mathcal{A}^i} \underbrace{p(x_t^i|x_{t-1}^i, a_t^i)}_{i\text{th agent's transition}} \underbrace{p(a_t^i|x_{t-1})}_{\text{driver model}} da_t^i. \quad (4)$$

Note that during the planning process, the ego semantic action ϕ_t^0 is specified by the policy that the planner is designed to explore. Similarly, we introduce semantic actions for other agents $\phi_t^i \in \Phi^i$ and reformulate the driver model in (4) as

$$p(a_t^i|x_{t-1}) = \sum_{\phi_t^i \in \Phi^i} \underbrace{p(a_t^i|x_{t-1}, \phi_t^i)}_{\text{agent controller}} \underbrace{p(\phi_t^i|x_{t-1})}_{\text{new driver model}} \quad (5)$$

wherein $p(\phi_t^i|x_{t-1})$ reflects the decision process of the the ith agent, which maps the driving context to its high-level action.

Similar to MPDM, we prune the belief tree in the action space by directly predefining the decision space within a limited number of policies. One step further, we extend the decision point in the time domain by introducing *domain-specific closed-loop policy tree* (DCP-tree). The nodes of the DCP-tree are predefined semantic actions associated with a certain time duration and the directed edges of the tree represent the execution order in time. The DCP-Tree starts from an *ongoing* action $\hat{\phi}$, which is the executing action from the best policy in the last planning cycle. Every time we enter a new planning episode, the DCP-tree is rebuilt by setting $\hat{\phi}$ to the root node. Note that the number of possible policy sequences scales exponentially w.r.t. the depth of the tree. To overcome this, the DCP-tree is expanded according to a predefined strategy, which comes from the observation that, as human drivers, we typically do not change the driving policy back and forth in a single decision cycle. Therefore,

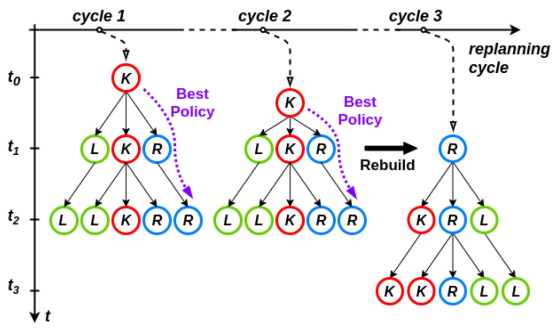


Fig. 4. Illustration of the proposed DCP-tree and the rebuilding process after its *ongoing* action changes. Suppose there are three discrete semantic-level actions $\{K, L, R\}$, denoting "keep lane," "left lane change," and "right lane change," respectively. The *ongoing* action for the left and middle tree is K, and the best policy is the dashed *purple* path. After executing K, the *ongoing* action switches to R, and the DCP-tree is updated to the right one. Each path only contains one change of action.

from the *ongoing* action, each policy sequence will contain at most one change of action in one planning cycle, as shown in Fig. 4, while the back-and-forth behavior is achieved by replanning. Note that the size of the leaf nodes in the DCP-tree is $O[(|A|-1)(h-1)+1]\ \forall h>1$, which grows linearly with respect to the tree height h. For more details about the DCP-tree, we refer readers to [21].

Without loss of generality, we define the policy $\pi \in \hat{\Pi}$ of our behavior planner as a sequence of semantic-level actions which are executed one by one. Once the DCP-tree is built, the whole decision space $\hat{\Pi}$ of the ego agent is determined. Candidate policies can be obtained by traversing all paths on the DCP-tree from the root node to all leaf nodes. Therefore, the behavior planning is simplified as selecting the best policy with maximum utility from the candidate policy set. Note that since the evaluations of each policy are independent of each other, we can implement the planner in a parallel fashion without additional effort.

C. Guided Observation Branching

To evaluate the utility of the policy, a common practice is to use the Monte Carlo-based method. The basic idea is to sample the starting state in the initial belief and use a black-box simulator which mimics the transitions and observations to generate the following states. Then, the belief of each node can be approximated using a particle filter as long as we have enough samples [23]. However, sampling over a high-dimensional space lacks efficiency, and modeling randomness in the simulation model makes it even worse. In this work, aiming at real-time applications, we follow the paradigm of black-box simulation and do further simplification for the belief update process. We assume the ego state x_t^0 is fully observable and the observations z_t (i.e., tracklets) are noise-free. This is a fair assumption since the ego motion and moving objects information can be estimated with high precision using state-of-the-art methods [42], [43]. Then, we let the agent policy $p(\phi_t^i|x_{t-1})$ in (5), i.e., the mapping from joint state to agent's semantic action, be deterministic by assuming the semantic action is fully determined by the hidden states, such as intention, and thereby the randomness of the agents' actions are converted into the belief state. As for the the transition model $p(x_t^i|x_{t-1}^i, a_t^i)$, we also use a deterministic kinematic model while the control noises can be injected to reflect the inaccuracy of the assumed model if the computational budget is sufficient. After the simplifications above, the only source of uncertainty is the hidden states of the other agent and their update processes after receiving observations.

Based on the understanding of how action branching is guided and implemented, the remaining problem is to estimate the semantic actions $\phi_t^{i\neq 0}$, or going one step further, the hidden states of other agents, which are partially modeled in the observation model $p(z_t^i|x_t^i)$. Different from our previous work [21], for other traffic participants, we adopt the assumption that the hidden states of other agents are not updated in the forward simulation, which means that the semantic actions of other agents are fixed. The reason is that once we apply the belief update in the simulation, the policy evaluation will suffer from heavy computation with exponential complexity ($O(|\mathbb{Z}|^d)$), which is unfavorable in real-time applications. And, in practice, we find the assumption for other traffic participants leads to satisfactory future anticipation, as shown in Section VIII. Finally, we estimate the belief of hidden states of other agents in the beginning, and the prediction result is used as the initial belief. We obtain a scenario by sampling over the initial belief and realize it by conducting the deterministic forward simulation

$$b_{t_H}(x_{t_H}) = b_{t_0}(x_{t_0}) \prod_{t=t_0}^{t_H} \left\{ \underbrace{p(x_t^0|x_{t-1}^0, a_t^0) p(a_t^0|x_{t-1}, \phi_t^0)}_{\text{ego state propagation}} \right. \\ \left. \prod_{i=1}^{N} \underbrace{p(x_t^i|x_{t-1}^i, a_t^i) p(a_t^i|x_{t-1}, \phi_0^i)}_{\text{other agents' state propagation}} \right\}$$

where the ego semantic action ϕ_t^0 is provided by the current policy. $\phi_t^{i\neq 0}$ only depends on the initial belief b_{t_0} and remains fixed during the forward simulation. Note that although we employ multiple simplifications, the forward simulation is still a closed-loop process since actions of all agents depends on the simulated state of the previous step x_{t-1}, retaining the interaction-awareness. As a result, the total utility of the policy is the weighted sum of the sampled scenarios, while the weights are exactly the value of the initial belief.

In this work, the semantic action for the other agent is represented by the action of pursuing a particular center-line. Therefore, estimating $\phi_t^{i\neq 0}$ is equivalent to estimating the probability distribution over the candidate center-lines that a certain agent is pursuing. In highway environments, this task is straightforward due to the simple lane structure, and we adopt a rule-based predictor in our previous work [21]. In urban environments, this task is more challenging due to more complex road structures. Essentially, in urban environments we are facing a varying number of candidate center-lines depending on the scene. To deal with this, we propose a lightweight classification network for estimating the probabilities while supporting a varying number

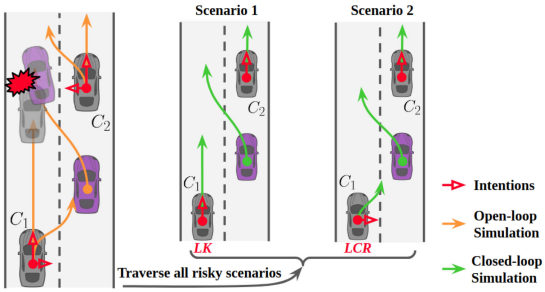


Fig. 5. Illustration of the CFB mechanism. Considering a case that the ego vehicle (purple) is considering a left lane-change. We use the open-loop simulation to get all possible future trajectories for all agents. Then, for each surrounding agent, we check whether there would be potential risks according to the simulated trajectories. For the critical agent (C_1), we unfold all combinations of the possible intentions, and for the vehicle with low risk (C_2), we directly employ the intention with maximum probability. As a result, the policy evaluation turns into an evaluation of the two scenarios. Interested readers may refer to [21] for more details.

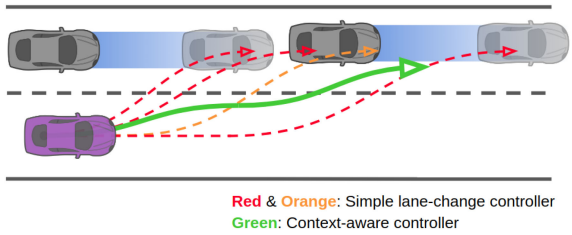


Fig. 6. Illustration of the difference between the simple controller and *context-aware* controller in the action branching. The lane-changing manoeuvres launched at different timestamps are realized by a simple pure pursuit controller (dashed curves). However, the simple controller does not *actively* pursue lane-changing, resulting in failure cases (red) or risky cases (orange), which is very inefficient. We find that it is beneficial to utilize the *context-aware* controller to achieve an advanced driving style, e.g., gap finding and velocity adaption. The resulting motion automatically adapts to different traffic configurations (green) with a higher success rate.

of classes. The details of the proposed learning-based intention estimator are elaborated in Section VII-C.

After obtaining the initial belief, the remaining problem is to determine $\phi_t^{i\neq 0}$ based on the estimated probabilities. Since each traffic participant potentially has multiple choices (e.g., multiple candidate center lines), we define one combination of intentions for all the traffic participants as one *scenario*. Intuitively, there is an exponential number of scenarios, and the probability for each scenario can be derived from the joint distribution of the estimated probabilities for each participants.[3] It is too expensive to examine all the possible scenarios when the agent number is large. In [21], we propose using conditional focused branching (CFB) for this task. Essentially, the goal of CFB is to find the intentions of nearby vehicles that may lead to risky outcomes with as few branches as possible. The term "conditional" means conditioning on the ego policies. The motivation comes from the observation that the attention of the human driver for nearby vehicles is biased differently when intending to conduct different maneuvers. For example, a driver will pay more attention to the situation in the left-hand lane rather than the right-hand one when he intends to make a left lane change. As a consequence, by conditioning on the ego policy sequence, we can pick out a set of relevant vehicles w.r.t. the ego policy. A toy example is provided in Fig. 5. The selection process is currently based on the rule-based safety assessment, which is identical to that in [21]. We point out that the critical agent selection can be implemented using learning-based methods based on real-world data. However, in practice, we find the proposed method can identify many risky cases despite its simple design.

D. Multiagent Forward Simulation

As mentioned above, a *scenario* is realized by the forward simulation, which propagates all agents in the environment simultaneously in an interactive manner. During the multiagent forward propagation, all trajectories of the ego vehicle and other agents are generated. It is worth noting that in many existing works (e.g., [7]–[14]), this type of task is conducted in an independent trajectory prediction module. However, after decoupling prediction and planning, it is problematic to account for the impact of the ego vehicle's future motion on the other agents. By contrast, our method couples the prediction inside the behavior planning, where each potential scenario considered by the planner has its own corresponding future anticipation. Essentially, at each time step in the planning horizon, each agent observes the latest states of other agents and obtains model-based control using a predefined driver model $p(a_t^i|x_{t-1})$. Based on the control, the agent's state is pushed one step forward following the state transition $p(x_t^i|x_{t-1}^i, a_t^i)$ of the agent.

In [21], we designed a *simple model-based* driver model by using the intelligent driver model (IDM) [44] for velocity control and pure pursuit [45] for steering control. Different from our previous work, we differentiate the controlled vehicle from other traffic participants. For the controlled vehicle, we design a *context-aware* controller for flexible and safe ego behaviors, while for other traffic participants we stick to a *simple model-based* controller. The proposed *context-aware* controller also contains a velocity controller and a steering controller, however, compared to the previous *simple model-based* approach, the new controller can generate more realistic maneuvers, especially in the lane change process. The reason that we utilize the *context-aware* controller is that there is considerable potential in the policy design for realizing the semantic-level actions, which can facilitate enforcing advanced driving styles during action branching. An intuitive example is provided in Fig. 6, in which we show how lane changing is conducted. By introducing the new controller, the solution space is enlarged and the resulting decision can adapt to the traffic configuration in a more flexible way. Benefiting from the *context-aware* controller, the number of action sequences needed is reduced significantly, and the success rate of the forward simulation is greatly improved, which further enhances the performance and efficiency of branching. Actually, the implementation of such controllers has been well studied in the traffic simulation literature and we detail our implementation in Section V-D.

[3]We assume each traffic participant makes decisions independently at each time instance.

Although the forward simulation model seems quite straightforward, we find it is actually accurate enough for anticipating the future. To our surprise, our lightweight design outperforms the two state-of-the-art trajectory prediction methods listed in Section VIII in terms of multimodal prediction. Moreover, our design provides the possibility of coupling prediction and planning, which cannot be achieved in traditional trajectory prediction methods. From the perspective of prediction, our multiagent forward simulation is like a "conditional" prediction, which conditions the prediction process on the decision of the controlled vehicle. The authors in [46] and [47] show that the predictor can achieve much better performance by considering the potential future plan of the agent.

E. Safety Mechanism

The foregoing discussion does not answer one important question: What if the real-world driver deviates from the predefined model? In practice, we observe that this issue often leads to underestimation of risk in decision-making, since real-world drivers are often stochastic and noisily rational. In traditional methods, this issue is dealt with by leaving sufficiently large safety margins [15], [33], which restricts the flexibility in highly interactive environments. To deal with this issue, we propose a safety mechanism with two levels.

First, we enhance the safety of the forward simulation model by embedding a safety module in the *context-aware controller* which can automatically ensure the control output is safe or *no-fault* to some extent. In terms of responsibility and safety, Shai *et al.* [48] defines the responsibility-sensitive safety (RSS), which provides a mathematical model for safety lateral and longitudinal distances in various driving situations and the *proper response* in dangerous situations (i.e., when safety distances are not satisfied). For each simulation step, we check whether the simulated state for the controlled vehicle is RSS-safe. If not, we check whether the control signal provided by the *context-aware* controller follows the criterion of *proper response* defined by the RSS model. If still not, we generate a safe control signal obeying the proper response and override the control output. We refer interested readers to [48] for a detailed description of RSS.

Second, we strengthen the robustness of the decision layer by setting the safety criterion in the policy selection. We are motivated by the reason why human drivers may perform an aggressive cut-in maneuver: For one thing, human drivers are aware of the possibility that other drivers may react to their insertion; for another, experienced human drivers have a safety evaluation process in their mind, namely, even in the case that other drivers are not cooperative, they still have the maneuverability to cancel their lane-change if they cannot complete it. Benefiting from our policy design, this feature can be achieved via the so called *backup plan*, which is naturally possessed by our behavior planner. Since our behavior planner evaluates multiple policies in one planning cycle, we can take any other policy that successfully fulfils the forward simulation and the safety check as a backup plan. In order to be more targeted, we handcraft a priority for the backup policy selection. For example, the backup plan for the "lane-change" policy should be "lane change cancelled." We can find the corresponding backup policies and check whether there is at least one feasible plan. If so, the planned policy can be executed. If there are no safe policies at all, meaning that the behavior planning has failed, a warning signal will be displayed, and the low-level active protection, such as autonomous emergency braking, will be triggered.

Algorithm 1: Process of Behavior Planning Layer.

1 Inputs: Current states of ego and other vehicles x; *Ongoing action* $\hat{\phi}$; Pre-defined semantic action set Φ; Planning horizon H;
2 $\mathfrak{R} \leftarrow \emptyset$; // set of rewards for each policy;
3 $\Psi \leftarrow \texttt{UpdateDCPTree}(\Phi, \hat{\phi})$; // DCP-Tree Ψ;
4 $\hat{\Pi} \leftarrow \texttt{ExtractPolicySequences}(\Psi)$;
5 **foreach** $\pi \in \hat{\Pi}$ **do**
6 $\quad \Gamma^{\pi} \leftarrow \emptyset$; // set of simulated trajectories;
7 $\quad \Omega \leftarrow \texttt{CFB}(x, \pi)$; // set of critical scenarios;
8 $\quad$ **foreach** $\omega \in \Omega$ **do**
9 $\quad\quad \Gamma^{\pi} \leftarrow \Gamma^{\pi} \cup \texttt{SimulateForward}(\omega, \pi, H)$;
10 $\quad$ **end**
11 $\quad \mathfrak{R} \leftarrow \mathfrak{R} \cup \texttt{EvaluatePolicy}(\pi, \Gamma^{\pi})$;
12 **end**
13 $\pi^*, \hat{\phi} \leftarrow \texttt{SelectPolicy}(\mathfrak{R})$;

F. Policy Selection

After the guided branching in both the action space and observation space, as well as the approximation of the transitions and observations, the behaviour planner can be simplified into a limited number of policy evaluation problems. For each policy, we calculate the weighted sum of the reward of each scenario by evaluating the planned behavior and the simulated trajectory. The reward function consists of three parts, namely, efficiency, safety, and navigation, which are elaborated in Section VII-D. Finally, the flow of our behavior planner is described in Algorithm 1. Evaluation for each policy sequence can be carried out in parallel (Line 5 to 11). Each critical scenario selected by CFB is examined by closed-loop forward simulation (Line 8 to 10) in parallel, and each policy is evaluated (Line 11) using the reward function, with the best policy selected (Line 13).

VI. Trajectory Generation With Spatio-Temporal Semantic Corridor

As stated in Section V, the generated final decision $\mathcal{D}_t = [x_{t+1}, x_{t+2}, \ldots, x_{t+H}]$ from the behavior planning layer contains a sequence of states for both the ego vehicle and the other agents. The remaining problem is to generate a smooth, safe, and dynamically feasible trajectory which faithfully follows the decision $\mathcal{D}_t$. In this article, we follow the formulation of our previous work [1] that characterizes the local solution space around $\mathcal{D}_t$ using the SSC, and then use an optimization-based formulation for trajectory generation within the SSC. For simplicity, we omit the detailed formulations here, and we refer interested readers to [1].

Due to the hierarchical structure of our planning system, the formulations and objectives of the behavior and motion layer are different, resulting in the potential discrepancy between the two layers and leading to inconsistent plan. Different from [1], in addition to minimizing the jerk along the trajectory, we incorporate the mean squared error of the trajectory to the reference state at the corresponding time, which represents the *similarity* between the trajectory and the ego decision in $\mathcal{D}_t$. Specifically, the cost J_j^σ of the jth segment on dimension σ can be written as

$$J_j^\sigma = w_s^\sigma \cdot \int_{t_{j-1}}^{t_j} \left(\frac{d^3 f_j^\sigma(t)}{dt^3} \right)^2 dt + w_f^\sigma \cdot \frac{1}{n_j} \sum_{k=0}^{n_j} \left(f_j^\sigma(t_k) - r_{jk}^\sigma \right)^2 \tag{6}$$

where w_s^σ and w_f^σ denote the weight for the *smoothness* cost and the *similarity* cost, respectively. r_{jk}^σ denotes the kth reference state on dimension σ generated by the ego-simulated states corresponding to the jth segment, and t_k is the timestamp of r_{jk}^σ. The number of reference states n_j in the jth segment is determined by the result of cube inflation. Note that we can further simplify the cost function and rewrite it in a quadratic formulation (ignoring constant terms)

$$\begin{aligned} J_j^\sigma &= \mathbf{p}_j^{\mathrm{T}}(w_s^\sigma \mathbf{Q}_s + w_f^\sigma \mathbf{Q}_f)\mathbf{p}_j + w_f^\sigma \mathbf{c}^{\mathrm{T}}\mathbf{p}_j \\ &= \frac{1}{2}\mathbf{p}_j^{\mathrm{T}}\hat{\mathbf{Q}}\mathbf{p}_j + \hat{\mathbf{c}}^{\mathrm{T}}\mathbf{p}_j \end{aligned}$$

where $\mathbf{Q}_s$ and $\mathbf{Q}_f$ are real symmetric matrices related to the smoothness term and similarity term, respectively, and $\mathbf{c}$ is a real vector. The objective function is simple and invariant given different combinations of semantic elements thanks to the SSC, which allows the formulation to easily adapt to different traffic configurations. Finally, the overall formulation can be written as a quadratic programming (QP), which can be solved efficiently using off-the-shelf solvers (such as OOQP [49]). In the case that no feasible solution can be found, the error is fed back to the behavior layer for further reaction.

VII. Implementation Details

A. Semantic-Level Actions and DCP-Tree

We consider both lateral and longitudinal actions to obtain a diverse driving policy. For both highway and urban environments, the lateral action set can be defined as the target center-line c_i of the adjacent lanes, since the most common lateral movement for a vehicle is interlane-changing among available lanes. Note that the number of lateral actions N_a^{lat} in the set is conditioned on the driving context since the adjacent lanes may be unavailable (due to traffic rules) or just not exist. For longitudinal motion, in order to ensure the continuity and safety, we use a predefined velocity controller rather than directly applying longitudinal acceleration signals. Thus, we define the semantic actions as a set of parameters of the velocity controller obtaining different longitudinal maneuvers. For example, in the IDM, we can achieve a more aggressive driving style by increasing the desired velocity while decreasing the time headway and minimum spacing. Here, we define the longitudinal semantic action set with $N_a^{\text{lon}} = 3$ items: {*Aggressive, Moderate, Conservative*}. Finally, we can get $N_a^{\text{lon}} \times N_a^{\text{lat}}$ semantic-level actions in total by combining the lateral and longitudinal actions. As for the DCP-tree, we set the depth of the tree as 5. We assign a time duration of 1 s for each semantic-level action, while the closed-loop simulation is carried out with a 0.2-s resolution to preserve the fidelity. Therefore, the planning horizon of the proposed behavior planner is 5 s.

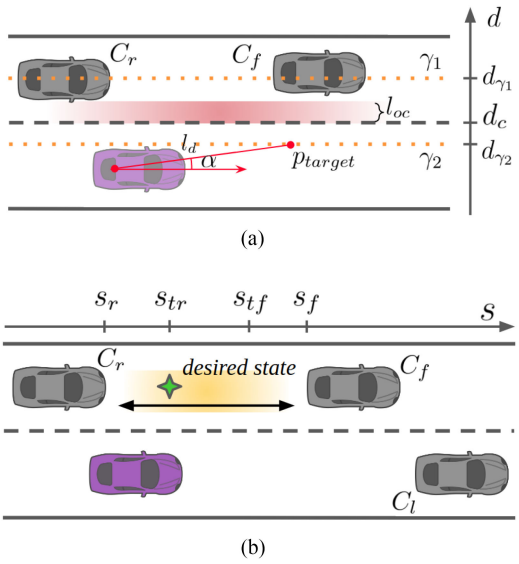


Fig. 7. Illustration of the proposed forward simulation model. The ego vehicle is shown in purple while the other agents are in gray. (a) Reactive lateral motion controller for lane-changing with obstacles in the target lane. The origin and alternate reference path are shown in orange, and the minimum clearance between lane markings and obstacles is represented by the red zone. The look-ahead distance l_d and heading error α can be quickly obtained once the target point p_{target} is determined. (b) Gap-informed longitudinal motion controller for lane-changing action. We consider three vehicles during lane-changing maneuvers, which are the current leading vehicle C_l and the new leader and follower C_f and C_r. The target gap formed between C_f and C_r is shown in yellow, while the desired longitudinal state is marked by the green star.

B. Context-Aware Forward Simulation Model

The goal of the closed-loop simulation is to push the state of the multiagent system forward while considering the potential interaction. For the proposed *context-aware* driver model, we utilize different lateral and longitudinal controllers for different combinations of semantic actions. Specifically, for lane keeping, the lateral controller follows the pure pursuit controller, while the longitudinal motion is governed by the IDM. For lane changing maneuvers, the longitudinal and lateral control is coupled since all the vehicles in the current and adjacent lanes need to be considered. We extend the lateral controller as the reactive pure pursuit, and the longitudinal controller as a *gap-informed* velocity controller.

To better illustrate the proposed forward simulation model, we provide two exemplar scenarios shown in Fig. 7. Note that the driving scenarios are presented in the curvilinear coordinate frame, where $s_{(\cdot)}$ and $d_{(\cdot)}$ denote the longitudinal and lateral positions in the path-relative coordinate frame. We brief the

TABLE I
DESCRIPTION OF THE RELATED VARIABLES OF THE PROPOSED FORWARD SIMULATION MODEL

Variables	Description
C_l, C_r, C_f	Current leading vehicle, the expected new leader and follower of the ego vehicle after the lane-changing
W_{ego}, L	Width and the wheelbase of the ego vehicle
γ_1, d_{γ_1}	Target center-line and its lateral position
γ_2, d_{γ_2}	Alternate reference path and its lateral position
d_c	Lateral position of the lane marking between the target and current lanes
l_{oc}	Observed minimum distance between the obstacles and the lane marking
l_{safe}	User-preferred minimum lateral clearance between the ego vehicle and obstacles
p_{target}	Look-ahead point for the pure-pursuit controller
l_d, α	Look-ahead distance and the heading error
s_{des}, $\dot{s}_{des}$	Desired longitudinal position and velocity for the ego vehicle while merging
s_{ego}, $\dot{s}_{ego}$	Ego longitudinal position and velocity
$\dot{s}_{pref}$	Preferred longitudinal velocity assigned by the user and traffic rules
s_r	Longitudinal position of the front bumper for the expected rear vehicle C_r
s_f	Longitudinal position of the rear bumper for the expected leading vehicle C_f
$\dot{s}_r$, $\dot{s}_f$	Observed longitudinal velocities of C_r and C_f
s_{tr}, s_{tf}	Threshold longitudinal positions derived from the safe time headway
l_{min}, T_{safe}	Minimum spacing and the safe time headway in the longitudinal direction

related variables in Table I, while the important ones are further introduced in the following contents.

For the lateral steering control, we first check whether the target lane has enough space for merging. If the adjacent lane is free, the center-line of the target lane γ_1 is assigned as the reference path of the pure pursuit controller, otherwise [Fig. 7(a)], we use an alternate collision-free path γ_2 as the reference. The lateral position of γ_2 can be calculated using

$$d_{\gamma_2} = d_c - \frac{W_{\text{ego}}}{2} - \max(0, l_{\text{safe}} - l_{oc})$$

wherein l_{oc} is the minimum distance between obstacles and the lane marking, and l_{safe} denotes the user-preferred minimum lateral clearance between the ego vehicle and obstacles [see Fig. 7(a)]. By tracking the alternate reference path γ_2, the controlled vehicle can slightly move toward the target lane without crossing the lane marking while keeping an appropriate distance from obstacles. The corresponding steering angle can be calculated using the pure pursuit controller $\delta_{\text{ctrl}} = \arctan \frac{2L \sin \alpha}{l_d}$.

For longitudinal control, a gap in the target lane is selected, and a desired longitudinal state is generated according to the states of the new leader C_f and follower C_r. The desired longitudinal state $[s_{\text{des}}, \dot{s}_{\text{des}}]$ can be calculated as where

$$s_{des} = \min(\max(s_{tr}, s_{\text{ego}}), s_{tf})$$
$$\dot{s}_{des} = \min(\max(\dot{s}_r, \dot{s}_{\text{pref}}), \dot{s}_f),$$

$[\dot{s}_r, \dot{s}_f]$ is the observed longitudinal velocity of C_r and C_f, $\dot{s}_{\text{pref}}$ is the preferred longitudinal cruise velocity assigned by user and traffic rules, and s_{tr} and s_{tf} are the threshold positions derived from the time headway corresponding to C_r and C_f

$$\begin{bmatrix} s_{tr} \\ s_{tf} \end{bmatrix} = \begin{bmatrix} s_r + l_{\text{min}} + T_{\text{safe}} \dot{s}_r \\ s_f - l_{\text{min}} + T_{\text{safe}} \dot{s}_{\text{ego}} \end{bmatrix}$$

where l_{min} and T_{safe} are the minimum spacing and safe time headway. The presented desired state provides us a target for the longitudinal control during the lane-changing maneuver, making the whole process comfortable and safe. Note that even when the bumper-to-bumper distance is small, the desired state still exists and the ego longitudinal state will gradually converge to the target and align with the gap. Furthermore, we point out that the desired state calculation still works when either C_f or C_r is absent by just simply eliminating the related terms in the equations above.

Once the desired longitudinal state is generated, a simple feedback controller is applied to push the ego vehicle toward the desired state. The output of the controller is the longitudinal acceleration a_{track} with the form of

$$a_{\text{track}} = K_v(\dot{s}_{\text{des}} + K_s(s_{\text{des}} - s_{\text{ego}}) - \dot{s}_{\text{ego}})$$

where K_v and K_s are the feedback gain of the controller. Note that during the lane-changing process, we also need to consider the current leading vehicle C_l to ensure safety. Here we use the ordinary IDM to get another acceleration a_{idm} for ego-lane distance keeping. Therefore, the output of the velocity controller is defined as $a_{\text{ctrl}} := \min(a_{\text{track}}, a_{\text{idm}})$.

As stated in Section V-D, to achieve a higher safety requirement, we double-check the lateral and longitudinal control command using the RSS-safety criterion. Once the ego is RSS-dangerous at the current simulation step, i.e., the situation is not RSS-safe according to the rules introduced in [48], feasible longitudinal and lateral limits on acceleration are provided. We compare the control constraints with the output signal (δ_{ctrl} and a_{ctrl}) and check whether the output satisfies the *proper response*. If not, we generate a safe control signal induced by the proper response and override the output.

For the vehicle model, i.e., the transition model $p(x_t^i | x_{t-1}^i, a_t^i)$, all agents are described using the kinematic bicycle model [50] for balancing computational efficiency and simulation fidelity.

C. Intention Estimator

To obtain better prediction performance, we present a neural network-based method to estimate the intention of the surrounding agents using the observation history and environmental information. The structure of the network is depicted in Fig. 8(a). At a high level, the network takes position samples from candidate center-lines and the history trajectory of the target vehicle as input features and uses an encoder–decoder structure to predict a scalar score for each candidate center-line. The score is then normalized by applying a softmax function. Since the network predicts the scalar score for each center-line instance and is not dependent on the number of center-lines n_c in the scene, the network can work in various urban environments, as shown in Fig. 9.

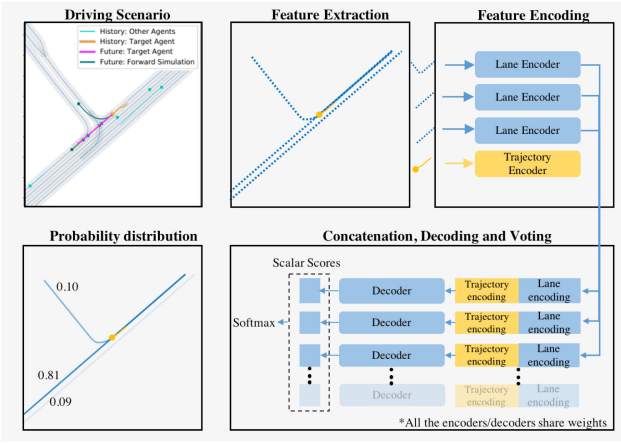


Fig. 8. Illustration of the intention prediction network. (a) Illustration of our lightweight network for estimating the semantic action (i.e., center-line) of other traffic participants. (b) Illustration of the network structure. In the trajectory encoder, h_0 is the initial hidden vector, which is a zero vector. In the lane encoder, the size of the input and output channel of the 1D convolution is presented. k denotes the kernel size, n_e is the size of the latent channel assigned beforehand, and n_l is the length of the embedding, in this case, $n_l = n_{cp} - 8$.

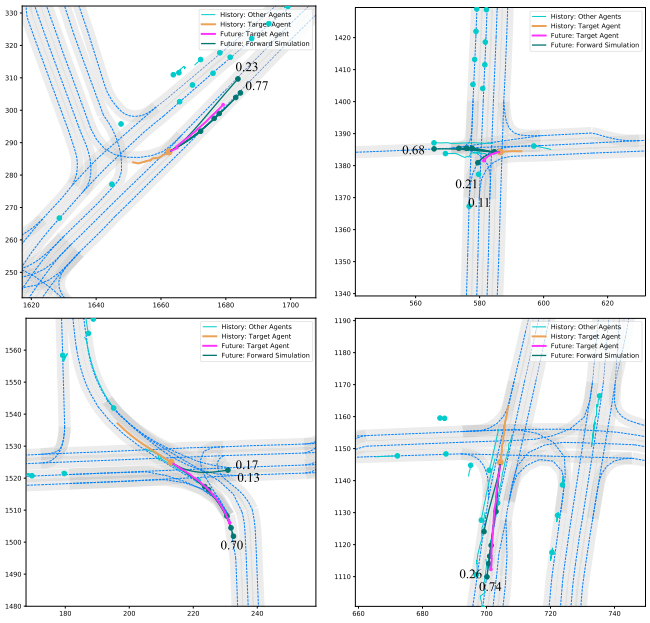


Fig. 9. Intention estimation and forward simulation on Argoverse validation set.

As shown in Fig. 8, three major modules are incorporated into the intention prediction network. The lane encoder takes the center lines of the surrounding lanes of the target vehicle as input. We take the sampled points on the center lines and use the position coordinates as input. For each center line, we sample $n_{cp} = 70$ points in total. The feature of the center-line is first encoded using multiple 1D convolutional layers with kernel size equal to 3. Then we employ an average pooling in the tail. For the trajectory of the target vehicle, we use an LSTM to encode the 2D positions recurrently and produce a latent vector, which is a compact representation of the motion history. We set the input size of the LSTM (after input embedding) to $n_x = 4$ and the length of the hidden state in the LSTM to $n_h = 64$. After processing the lane and trajectory information, we concatenate the trajectory embedding with every lane embeddings, then use a multilayer perceptron (MLP) with a ReLU activation function to estimate the score of the candidate lane. The final probability distribution is obtained via the softmax function. Fig. 8(b) shows the structure of the modules introduced above. Since the intention prediction is essentially a classification problem, we use the negative log-likelihood as the loss to minimize.

D. Policy Evaluation

The overall reward for a policy is calculated by the weighted sum of the reward for each CFB-selected scenario. The reward function of each scenario consists of a linear combination of multiple user-defined metrics, efficiency cost F_e, safety cost F_s, and navigation cost F_n

$$F_{\text{total}} = \lambda_1 F_e + \lambda_2 F_s + \lambda_3 F_n$$

where λ_1, λ_2, and λ_3 are weights to balance the cost terms, and F_{total} is the total cost (negative reward, i.e., $F_{\text{total}} = -R$).

To evaluate the efficiency cost, two sources are considered after a semantic action has finished. One item is the velocity difference between the current simulated velocity and the preferred velocity for the ego vehicle $\Delta v_p = |v_{\text{ego}} - v_{\text{pref}}|$. The other item partially represents the efficiency of the target lane by checking the velocity overshoot w.r.t. the leading vehicle $\Delta v_o = \max(v_{\text{ego}} - v_{\text{lead}}, 0)$, and calculating the velocity difference between the leading vehicle and the ego's preferred velocity $\Delta v_l = |v_{\text{lead}} - v_{\text{pref}}|$. Therefore, the efficiency of the policy can be written as

$$F_e = \sum_{i=0}^{N_a} F_e^i = \sum_{i=0}^{N_a} (\lambda_e^p \Delta v_p + \lambda_e^o \Delta v_o + \lambda_e^l \Delta v_l)$$

where λ_e^p, λ_e^o, and λ_e^l are weights and N_a is the number of semantic actions in a policy.

For the safety cost, we evaluate all the discretized states of the ego-simulated trajectory together with all other trajectories nearby. If a collision occurs, the corresponding scenario will be directly marked as failed and punished with a great cost. In addition, we also check whether the ego trajectory contains an RSS-dangerous state. If the state is RSS-dangerous, we obtain the safe velocity interval $[v_{\text{rss}}^{lb}, v_{\text{rss}}^{ub}]$ given the current situation, and punish the state by the difference from the safe velocity

interval. The safety cost is written as

$$F_s = \lambda_s^c b_{\mathrm{c}} + \sum_{i=0}^{N_s} b_{\mathrm{r}} F_s^i$$

$$= \lambda_s^c b_{\mathrm{c}} + \sum_{i=0}^{N_s} b_{\mathrm{r}} v_{\mathrm{ego}} \lambda_s^{r1} e^{\lambda_s^{r2} |v_{\mathrm{ego}} - \min(\max(v_{\mathrm{ego}}, v_{\mathrm{rss}}^{lb}), v_{\mathrm{rss}}^{ub})|}$$

where b_{c} and b_{r} are the Boolean values denoting the collision and the violation of RSS-safety for the ego simulated state, respectively. λ_s^c is the penalty for collision. N_s is the total number of the simulated states in the ego trajectory. The cost for RSS-dangerous is designed in an exponential form to emphasize the degree of the RSS-safety violation while λ_s^{r1} and λ_s^{r2} are tunable parameters.

The navigation cost is determined by the user preference and decision context. We enforce a reward (negative cost) for the policies that match the user's command provided by the human–machine interface, which can realize driver-triggered lane-change. Moreover, to improve the decision consistency, we also reward the policies with a similar meaning to the decision made by the last planning cycle. For example, if the optimal plan for the last cycle is "changing to the left lane," we will assign a reward to the policies that achieve the same maneuver. Therefore, the cost term is represented as

$$F_n = \lambda_n^{\mathrm{user}} b_{\mathrm{navi}} + \lambda^{\mathrm{consist}} b_{\mathrm{consist}}$$

where b_{navi} and b_{consist} denote the flags of the policy that matches the navigation goal and decision history, respectively.

Note that the safety cost F_s and efficiency cost F_e are generated w.r.t. each semantic-level action. A discount factor γ is introduced to adjust the weight for rewards in the distant future, and we set it to $\gamma = 0.7$.

VIII. Experimental Results

In this section, we present the quantitative and qualitative analysis. The quantitative validations are based on the dataset and simulation, while the qualitative experiments are conducted on a real vehicle in dense real-world traffic.

A. Quantitative Analysis

We verify the effectiveness of the key features of EPSILON, namely, semantic-level action, interaction-awareness, and the safety mechanism. We begin with a quantitative analysis on semantic-level action and forward simulation, since they determine how each scenario is realized. To verify the contributions of interaction-awareness and the safety mechanism, we conduct ablative studies for each component.

1) Semantic-Level Action and Forward Simulation: As introduced in Section V-D, the semantic-level action and forward simulation are highly related to behavior/trajectory prediction techniques in the literature. The difference is that traditional prediction is often independent of planning, while our semantic-level action and forward simulation are coupled inside our behavior planner.

The reason why the semantic-level action and forward simulation can be coupled into planning is due to the simplification induced by the semantic-level actions. Specifically, our behavior planning reasons about a limited set of predefined closed-loop policies which allows for efficient and tight integration. However, it remains questionable whether this setup oversimplifies the problem and results in significant loss in the fidelity of future anticipation. To answer this question, we compare our intention estimation and forward simulation pipeline with two state-of-the-art learning-based trajectory prediction techniques, namely, TPNet [51] and UST [52]. Both methods [51], [52] can generate multimodal trajectory predictions with state-of-the-art accuracy.

Dataset We adopt the Argoverse [53] motion forecasting dataset for comparisons, which is a large-scale autonomous driving dataset with rich map information. It provides a semantic graph which supports lane-level query. The dataset is collected in two different cities, Pittsburgh and Miami, including 205 942 sequences for training, 39 472 sequences for validation, and 78 143 sequences for testing. These three subsets are split with no geographical overlap. Both TPNet [51] and UST [52] have been also validated on this dataset. We use 2-s past observations and predict spatial locations for the target vehicle in the future 3 s, which is the common setup used in [51]–[53].

Training We implement the network using Pytorch [54], and train the network on a single NVIDIA GTX 1080 GPU. The network is trained in an end-to-end fashion using Adam optimizer [55] with a batch size of 128. We initiate the learning rate to 1e-3 and decay to 1e-4 after 35 epochs. The total time cost is around 3 h for 50 epochs.

Metrics We follow the official metrics of Argoverse to benchmark multiple predictions on this dataset. Average displacement error (ADE) and final displacement error (FDE) are the most used metrics in motion prediction. In order to evaluate the capability for modeling multimodality, Argoverse [53] also uses minimum over N (MoN) metrics such as minADE and minFDE. Specifically, minFDE represents the distance between the endpoint of the best forecasted trajectory (i.e., the trajectory with the minimum endpoint error) and the ground truth, while minADE represents the average distance between the best forecasted trajectory and the ground truth. Following the setup of TPNet [51] and UST [52], we evaluate minADE and minFDE with six multimodal trajectory predictions.

Baselines All the baseline methods are compared on the Argoverse test set. Depending on whether map information or prior knowledge is used, TPNet [51] has several variants. Apart from TPNet [51] and UST [52], we also include another two baselines, namely, NN and LSTM ED, which are commonly used in comparisons.

1) *Nearest Neighbor (NN)* [53]: Weighted nearest neighbor regression using top-N (N=6) hypothesized center-lines.
2) *LSTM ED* [53]: LSTM encoder–decoder network with road map information as input.
3) *TPNet* [51]: TPNet with only 2-s past observations and without road map information.
4) *TPNet-Map*: TPNet with road map information as input.

TABLE II
COMPARISON WITH BASELINE METHODS ON ARGOVERSE TEST SET

Methods	ADE (m)	FDE @ 3s (m)	minADE (m), N=6	minFDE (m), N=6
NN [53]	3.45	7.88	1.71	3.29
LSTM ED [53]	2.96	6.81	2.34	5.44
TPNet [51]	2.33	5.29	2.08	4.69
TPNet-map [51]	2.23	4.71	2.04	4.23
TPNet-map-safe [51]	2.23	4.70	2.03	4.22
TPNet-map-mm [51]	**2.23**	**4.70**	1.61	3.28
UST [52]	-	-	1.47	2.94
Our method	2.58	5.70	**1.41**	**2.48**

TABLE III
ABLATIVE STUDY ON THE IMPACT OF INTENTION ESTIMATION ACCURACY ON ARGOVERSE VALIDATION SET

Methods	ADE m	FDE @ 3s	minADE (m), N=6	minFDE (m), N=6
Without intention estimation	3.94	9.13	2.07	4.16
With intention estimation	2.19	4.80	1.20	2.05
Oracle intention estimation	2.12	4.58	1.13	1.83

5) *TPNet-Map-Safe*: TPNet with prior knowledge for constraining proposals.
6) *TPNet-Map-mm*: TPNet with prior knowledge for generating multiple proposals.
7) *UST* [52]: Unified spatio-temporal representation for learning-based trajectory prediction.

Results As listed in Table II, for prediction accuracy with only one trajectory hypothesis, our method achieves 2.58 for ADE and 5.7 for FDE, outperforming NN and LSTM ED baselines, but there is still a gap w.r.t. to the two state-of-the-art baselines. When considering multiple prediction hypotheses, which is the common case in real-world driving, our method outperforms all the baseline methods (incl. the two state-of-the-art methods) significantly, achieving 1.41 for minADE and 2.48 for minFDE. Note that the motivation for introducing the semantic-level action and forward simulation is not targeting "prediction accuracy." Instead, it aims at providing a lightweight querying interface which can be coupled inside planning for realizing diverse future scenarios conditioned on the ego decision. To our surprise, despite our lightweight design, our method still achieves satisfactory accuracy, which supports our claim that simplified predefined controllers are sufficient for faithfully capturing the future scenarios.

To further study the impact of the accuracy of intention estimation on the scenario realization, we conduct an ablative study in which we replace the intention estimation network with a dummy predictor which generates a uniform probability distribution on all candidate center lines. On the other hand, we set up an oracle intention estimator which uses the future ground truth center line for the forward simulation. We compare these two alternatives with our method on the Argoverse validation set, as shown in Table III. We find that without an intention estimation module, the accuracy of forward simulation drops significantly in terms of ADE and FDE, and also in terms of minADE and minFDE, which indicates that it is important

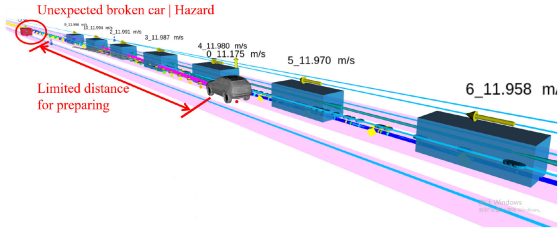


Fig. 10. Illustration of the benchmark scenario.

to include an intention estimation module (even a lightweight one) in complex environments. On the other hand, with an oracle intention predictor, the accuracy is improved but the gap is relatively small. Our conjecture is that since the forward simulation is based on predefined controllers, when the intention estimation becomes perfectly accurate, the accuracy bottleneck is on the predefined models. In the future, we may explore using learning-based controllers for the forward simulation, as we previously attempted in [47]. However, we think only marginal improvements can be made, considering that the accuracy of the existing method (a lightweight intention estimator with a model-based forward simulator) is already high (since it outperforms the two state-of-the-art methods).

2) Impact of Coupling Prediction and Planning: To quantitatively examine the impact of coupling prediction and planning, we present a benchmark scenario for testing. The setup of the benchmark scenario follows a typical driving configuration which we often encounter in city driving, as shown in Fig. 10. In this scenario, there is an unexpected broken-down car ahead, and the controlled vehicle should react in a limited preparation distance. If the controlled vehicle fails to grasp the opportunity to change lanes, it is forced to slow down, which not only affects its travel efficiency but also leads it into a hazard as it needs to wait indefinitely for the incoming traffic to clear. However, changing lanes may also be risky. On the one hand, the controlled vehicle should not accelerate too much, considering the risk of collision in the case of sudden braking by the leading vehicle in the nearby lane, while on the other, it should complete the lane change in a timely way to be safe (in the sense of responsibility) from a rear collision with the following vehicle. To achieve good performance in this benchmark, the planner is required to conduct counterfactual reasoning, namely, examining how different future decisions of the ego vehicle affect its surrounding traffic participants.

Note that in the simulation, all other traffic participants (AI agents for short) are controlled by their own policy according to the perception information they received. The policy for AI agents is a simple adaptive cruising policy adapted from the IDM [44] that mimics a real traffic queue. We define three aggressiveness levels by adjusting the *desired time headway* that directly determines the longitudinal distance to the leading vehicle. We also define the cooperative range to describe the friendliness of the AI agents to the queue-jumper vehicle, namely, the AI agents will yield and keep distance to the cut-in vehicle when the lateral distance between the vehicle centroid and the current center-line is smaller than the cooperative range. To make a

TABLE IV
COMPARISON WITH DECOUPLED PREDICTION AND PLANNING SYSTEM AND EPSILON W/O SAFETY MECHANISM

Methods	Aggress. level	Cooperative range (m)	Travel effi. (m/s)	Safety cost
Decoupled	1 (2.0 s)	2.55	11.1 (✓)	4.5
EPSILON w/o safety	1 (2.0 s)	2.55	11.1 (✓)	3.85
EPSILON	1 (2.0 s)	2.55	11.1 (✓)	3.05
Decoupled	2 (1.5 s)	2.00	10.6 (✓)	81.4
EPSILON w/o safety	2 (1.5 s)	2.00	10.5 (✓)	65.9
EPSILON	2 (1.5 s)	2.00	10.6 (✓)	13.7
Decoupled	3 (1.0 s)	1.75	7.3 (✗)	42.3
EPSILON w/o safety	3 (1.0 s)	1.75	10.7 (✓)	147.2
EPSILON	3 (1.0 s)	1.75	10.8 (✓)	16.8

[3]The time in the brackets denotes the value of *desired time headway.*

fair experiment, the settings such as the aggressiveness level and cooperative range of the AI agents is unknown to the ego planning system.

Metrics In the benchmark scenario, *safety* and *travel efficiency* are two representative metrics. Successful merging leads to higher travel efficiency, since failure of lane change results in waiting indefinitely in the hazard. However, higher efficiency should not come at the price of sacrificing safety. The travel efficiency metric can be simply represented by the average velocity, while the safety metric can be represented by RSS. Specifically, we evaluate the safety cost of the chosen decision (i.e., a sequence of states) for every planning cycle and calculate the average. We quantitatively evaluate the average travel efficiency of the trace of the ego vehicle and the average safety cost of each decision for the first 300 planning cycles (around 15 s) which are the time needed to safely pass the hazard. We vary the aggressiveness and cooperative range of the traffic participants in the nearby lane to adjust the level of difficulty.

Baseline The baseline for the comparison is the traditional decoupled prediction and planning pipeline. Specifically, we use an independent trajectory prediction module which is derived from multiagent forward simulation, but exclude the decision of the controlled vehicle. The resulting prediction is unaware of the impact of the ego's future action, while retaining the interaction among other traffic participants, which maintains the exact functionality of the popular trajectory prediction methods [51], [52]. After obtaining the predicted trajectory, we generate multiple action sequences in the same way as EPSILON and pick out the best action sequence using the same evaluation pipeline as for EPSILON.

Results As shown in Table IV, we find that, in less aggressive and cooperative traffic (aggressiveness 1 with 2.0-s headway time and 2.55-m cooperative range), the performance of the decoupled planning system is close to that of EPSILON. Both methods can conduct lane change successfully with high travel efficiency while also staying safe. However, when increasing the aggressiveness of other traffic participants to 3 with 1.0-s headway time, and narrowing down the cooperative range to 1.75 m, decoupled prediction-planning method cannot find a gap in which to merge in the preparation range (marked by ✗), resulting in waiting in the hazard. Note that the time of interest is set to the first 300 planning cycles, and when considering a longer horizon for calculating statistics, the average velocity of the decoupled method can be even lower due to waiting for the clearance of all the traffic. By contrast, EPSILON can successfully achieve safe lane change with high travel efficiency (10.8 m/s) and a small safety cost 16.8. The reason is that the ego vehicle can implicitly negotiate with the traffic participants in the nearby lane by considering the impact of different future decisions. To summarize, the performance gain from using the coupled prediction-planning framework appears in dense and aggressive traffic.

3) Impact of the Safety Mechanism: As introduced in Section V-E, both our previous method [21] and MPDM [26] suffer from the problem that the predefined policies may be overconfident, which results in overaggressive maneuvers. To overcome this, we introduce a safety mechanism in Section V-E. To verify its effectiveness, we set up a baseline which excludes the safety mechanism of EPSILON ("EPSILON w/o safety" in Table IV), which essentially represents the performance of our previous method [21]. The benchmark scenario and metrics are the same as those in the previous section.

As shown in Table IV, we find that the performance gap between "EPSILON w/o safety" and EPSILON is quite small in loose and cooperative traffic. However, in aggressive and less cooperative traffic, for example, when the aggressiveness level is 3 with a 1.75-m cooperative range, there is a huge gap in terms of the safety cost of resulting decisions. Although EPSILON w/o safety can conduct the lane change and achieve good travel efficiency with an average speed of 10.7 m/s, it is overconfident about other traffic participants' cooperativeness, which incurs an average safety cost of 147.2, which is 8.7 times that of EPSILON.

A typical example is provided in Fig. 11 to illustrate how the difference arises. As shown in Fig. 11(a), EPSILON w/o safety always assumes the following vehicle in the nearby lane will yield according to the predefined model (e.g., IDM). However, when the following vehicle is not as cooperative as the planner preassumes, the controlled vehicle will pick out a decision which is overaggressive. In this case, EPSILON w/o safety leads to a near-collision scenario. In contrast, thanks to the safety mechanism, when the controlled vehicle tries to change lanes, EPSILON incorporate the lane change with an backup maneuver. With this maneuver, the controlled vehicle does not need to "estimate" the cooperativeness of the following vehicle and can always assume it is not cooperative. However, this conservative assumption does not prevent the controlled vehicle from conducting the lane change due to the existence of the backup plan. The resulting situation becomes that the ego vehicle will first adjust itself to a position which is safe, and merges when the following vehicle responds. This "push-wait-merge" style of merging is automatically achieved by the safety mechanism instead of using complex handcrafted logics.

4) Computation Time Cost: To verify whether the proposed planning system meets the requirement of real-time applications, we analyze the computational time cost of both the behavior and motion layer in the simulation on a consumer-level desktop (Intel i7-8700 with 32 GB RAM). By changing the number of vehicles considered, we can obtain the computational time cost from simple to complex scenarios, and for each situation, we

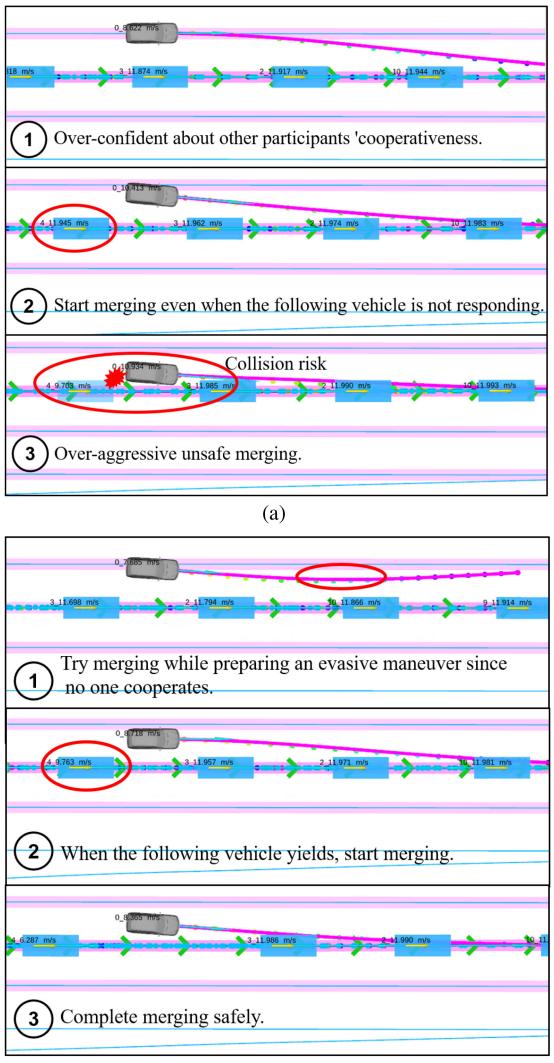


Fig. 11. Illustration of merging into dense traffic w/ and w/o safety mechanism. Note that the decoupled prediction-planning method is excluded here since it fails to conduct the lane change and gets trapped in the hazard. The best decision is visualized by the *rainbow* dots, while the corresponding trajectory is marked in *purple*. (a) Merging w/o safety mechanism. (b) Merging w/ safety mechanism.

collect over 250 frames for statistical analysis. As shown in Fig. 12, the time cost of the behavior planner increases w.r.t. the number of surrounding vehicles, while the time cost of the motion planner is basically constant. We can find that the time cost of these two modules is all restricted within 50 ms, and by leveraging the pipeline structure mentioned in Section III, our EPSILON can run stably at 20 Hz.

B. Qualitative Analysis

In this section, we present the onboard experimental results in the real-world traffic. Our experiment platform is a prototype vehicle refitted from a passenger-SUV, which is equipped with a sensing suite including two sets of forward-facing stereo cameras, four fish-eye cameras, four millimeter-wave radars, six ultrasonic sensor, and a low-cost INS. Neither Lidar nor high-precision positioning system is applied in this platform.

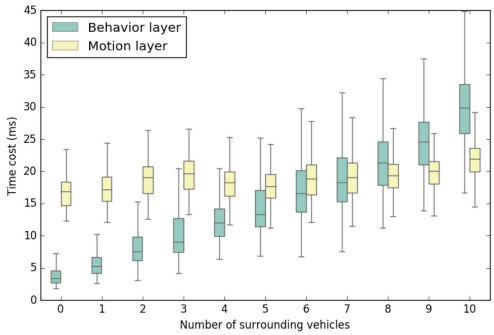


Fig. 12. Illustration of the computational time required by the proposed planning system w.r.t. the number of vehicles considered. Results for both the behavior layer and motion layer are presented.

Fig. 13. Illustration of the test road. The length of the test route is approximately 12.5 km and contains up to four lanes in each direction. Unlike other intercity expressway, there is an entrance or exit every 500 m on average, so vehicle interactions on this road are very common. (Satellite imagery credit: Google). (a) Test route is marked in yellow. (b) Street view of the test road.

Note that the HD-map is excluded during the real-world road tests, both localization information and road structure are inferred online. And the other traffic participants (surrounding vehicles) are detected and tracked by the perception module. Our planning system receives the data from these upper-level modules and sends the planned trajectory to the control system. The vehicle control system is based on the model predictive control which achieves accurate tracking performance in both low and high-speed driving. For onboard testing, our planning system is implemented using C++ and deployed on a single NVIDIA Xavier with an 8-core ARM-base CPU.

We validate our system in a busy urban expressway shown in Fig. 13. The test road is a four-lane dual carriageway with a speed limit of 80 km/h and the total length of the test route is about 12.5 km. We conduct four test drives on this same route. For the first two trials, we disable the active lane-changing option and we encourage the human driver to trigger lane change command as he wants. For the other two trials, we enable the active lane change and try to rely only on the lane change decision proposed by the planner.

The statistics of the evaluation metrics are shown in Table V. As we can see, all the maximum longitudinal and lateral accelerations are limited in an appropriate range resulting in a comfortable ride (also see Fig. 14). Due to the presence of many ramp entrances and exits, the surrounding traffic participants will frequently change lanes and merge in front of the ego vehicle, causing plenty of interactions. Among the four test drives, we achieve a reasonable overall success rate for lane changing (over 80%) in such highly interactive environments. Note that

TABLE V
EVALUATION METRICS OF THE FOUR TEST DRIVES

	Trial 1	Trial 2	Trial 3	Trial 4	Sum
Avg vel (km/h)	56.2	61.0	30.8	59.3	-
Min vel (km/h)	24.2	14.2	0.0	10.7	-
Max vel (km/h)	72.4	90.9	74.9	77.5	-
Max lon acc (m/s^2)	1.83	2.00	2.02	1.65	-
Max lon dec (m/s^2)	-2.07	-3.24	-4.88	-2.08	-
Max lat acc (m/s^2)	1.35	1.60	0.99	1.29	-
Time duration	$14'42''$	$12'35''$	$25'16''$	$12'14''$	-
Lane change triggered	14	17	8	10	49
Lane change success	12	15	5	8	40
Backup plan triggered	3	2	3	2	10
Backup plan success	2	1	1	0	4
Human take-over for unsafe lane change	1	1	2	2	6
Human take-over for unsafe distance	0	0	1	0	1

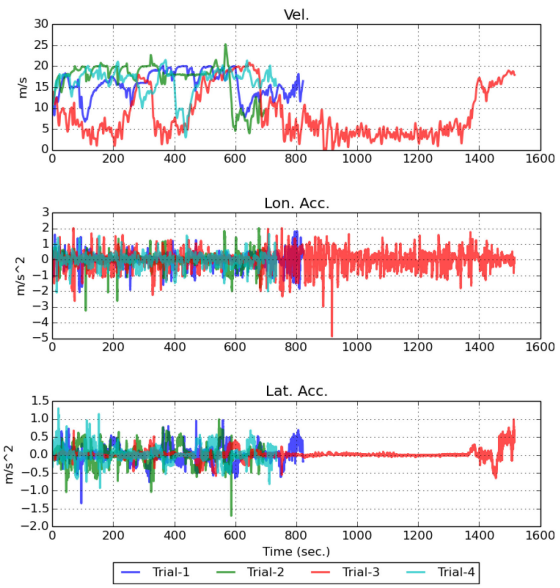


Fig. 14. Dynamic profiles for the four test drives. The time duration of the third trial is much longer than the others since the road is quite congested during the experiment. With the exception of a few rare cases, the vehicle's dynamics were kept within a comfortable range. For Trial-2 at around 590th second, a lane-cancel behavior was executed in a risky situation, causing a relatively large lateral acceleration up to 1.60 m/s^2 (actually still acceptable). At around 920th s of Trial-3, the controlled vehicle made a hard brake (-4.88 m/s^2) because of a drastic deceleration of the leading vehicle.

lane change cancellations handled by the backup plans are not recorded as success cases. Besides, we observe that human-like maneuvers such as "find gap then merge" and cut-in handling can be achieved automatically. Despite risky scenarios are inevitable in such interactive driving environments, our planner can handle them appropriately by leveraging the proposed backup plans and reduce the number of human interventions. We find that the driving time duration for the third trial is much longer than others because the test vehicle encounters traffic jams during the experiment, leading to a much lower average velocity and more challenging driving scenarios. As a result, the success rate of lane change drops a lot in the third trial (62.5%), and the number of human intervention is larger (3 times in total). We blame the performance degeneration in two aspects: First, since the traffic jams often appear near the ramp exits and entrances, traffic participants are more likely to conduct aggressive maneuvers leading to more risky scenarios; second, in congested traffic, the noises of sensor data increase since there is more occlusion due to the complex environment, resulting in inaccurate evaluations and improper decisions. Besides, the parameters of our planner are tuned to be more proactive causing more aggressive maneuvers during the experiments. We find nearly half of the interventions are due to improper trigger time for active lane change under noisy observations (3 times), and the rest of them are due to 1) blocked by aggressive drivers (twice), 2) infeasible plan caused by the misestimation of others' intention (once), and 3) drastic deceleration of the leading vehicle (once). Here, we point out that simply reducing the number of human interventions is not a major target of this work since once we tune the planner into a conservative style, metrics such as miles per intervention will increase significantly. At the same time, we expect the planner that produces as flexible behaviors as possible while preserving driving safety.

To show the flexibility and robustness of our planning system in detail, we provide four representative cases collected from our daily road tests.

1) Intelligent Gap Finding and Merging: In Section V-B, we show that by incorporating advanced driving styles into semantic actions, we can fully exploit the power of semantic actions and achieve intelligent and flexible maneuvers. As shown in Fig. 15(a), (A) the user indicates a lane change requirement by a stick signal, and (B) the controlled vehicle moves forward and automatically finds the appropriate gap to merge. In (C), the controlled vehicle finally finds an appropriate gap and starts merging, and in (D) the controlled vehicle completes the merging. From this example, we observe that by introducing driving knowledge through intelligent driving controllers, the lane-change semantic action can be much more flexible and automatically adapts to the traffic configuration. Compared to MPDM [26], the maneuverability of the controlled vehicle is more flexible and intelligent, while compared to pure rule-based decision making methods, EPSILON is more general and does not rely on complex handcrafted logics.

2) Interaction-Aware Overtaking: In Section V-D and V-E, we present the multiagent forward simulation and safety mechanism to consider interaction in a safe way. As illustrated in Fig. 15(b), we show how interaction-aware overtaking can be achieved. In (A), the user indicates a lane change requirement. However, we find that the following vehicle in the neighboring lane is not cooperative. What is worse, it tries to compete with the controlled vehicle and prevent it from changing lanes (B). Actually, this is a common driving style in the area where the field tests are conducted. In (C), the ego vehicle further increases the speed and from the dynamic profile, we can see that the acceleration is even larger compared to the first phase, to increase the gap with respect to the following vehicle. This acceleration is due to the active safety mechanism conducting a longitudinal evasive maneuver so that even in the case of a bump-to-rear collision, the controlled vehicle would not be at fault since it would have already completed the lane change. The

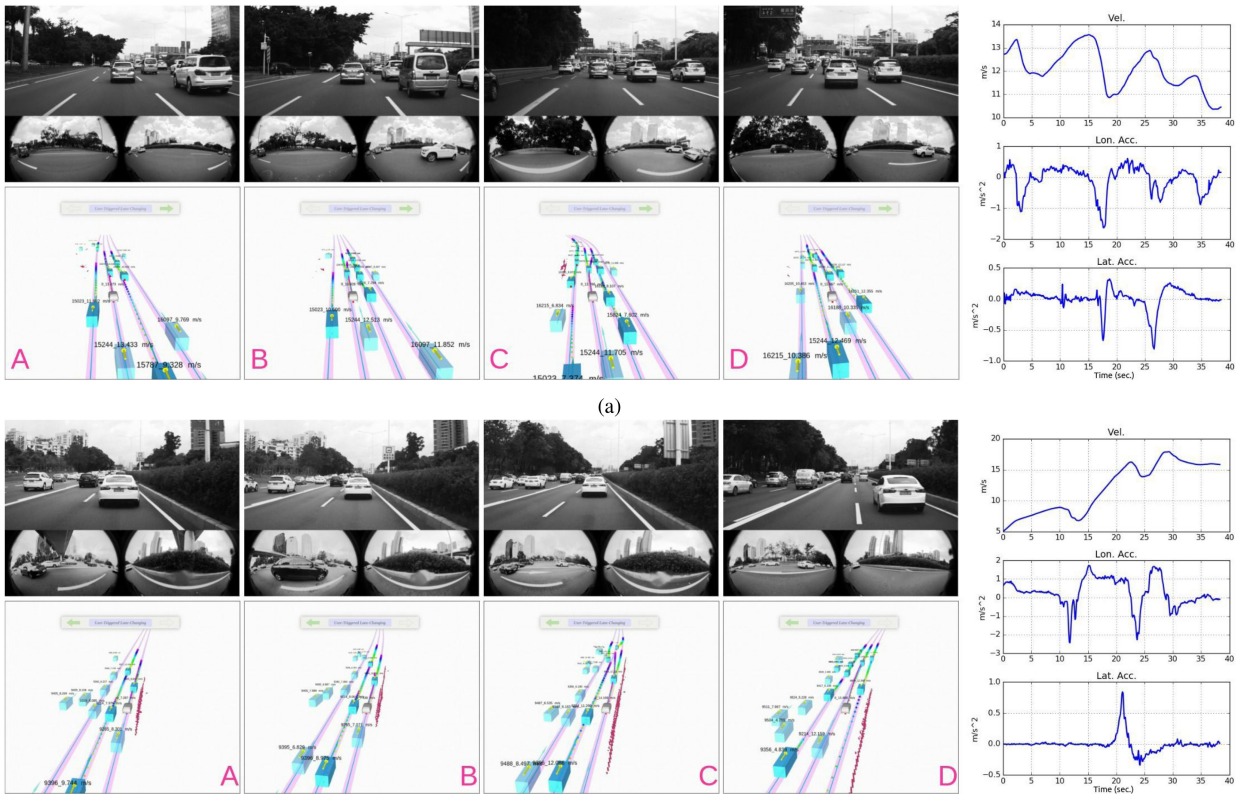
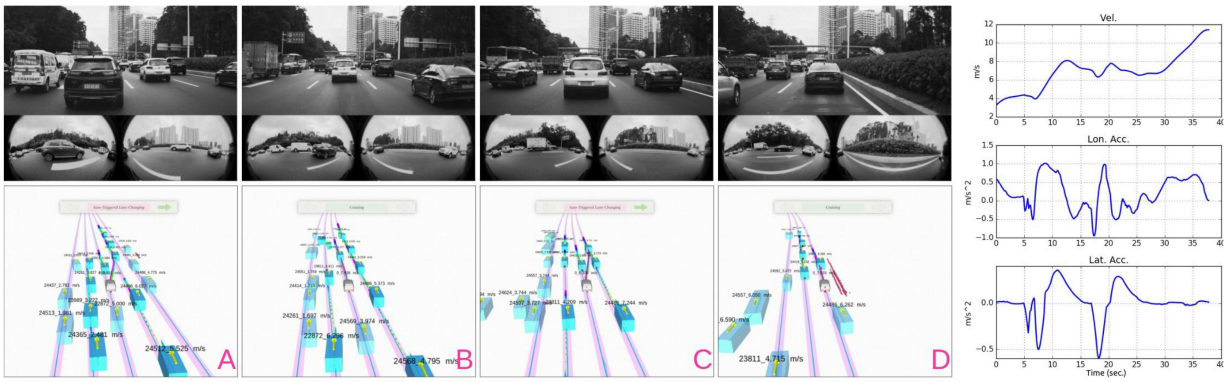


Fig. 15. Illustration of two representative scenarios collected from road tests. The tracklets are represented by *blue* bounding boxes, the static obstacles are marked in *red* and the controlled vehicle is represented using the model vehicle. In the visualization, we illustrate the forward simulation results, both for the controlled vehicle and surrounding vehicles, using *rainbow* dots, where the color reflects the elapsed time. For a clear visualization, we only draw the forward simulation results for the most likely scenario, and the corresponding intention estimations are marked by *yellow* arrows. In these two cases, the user inputs a stick signal, which requires the planning system to conduct a lane change for the user. EPSILON automatically finds the best time and the best gap to smoothly and safely complete the merging requirement. On the right we plot the dynamic profile (velocity, longitudinal and lateral acceleration) during the process. (a) Automatic gap finding and merging. (b) Interaction-aware overtaking.

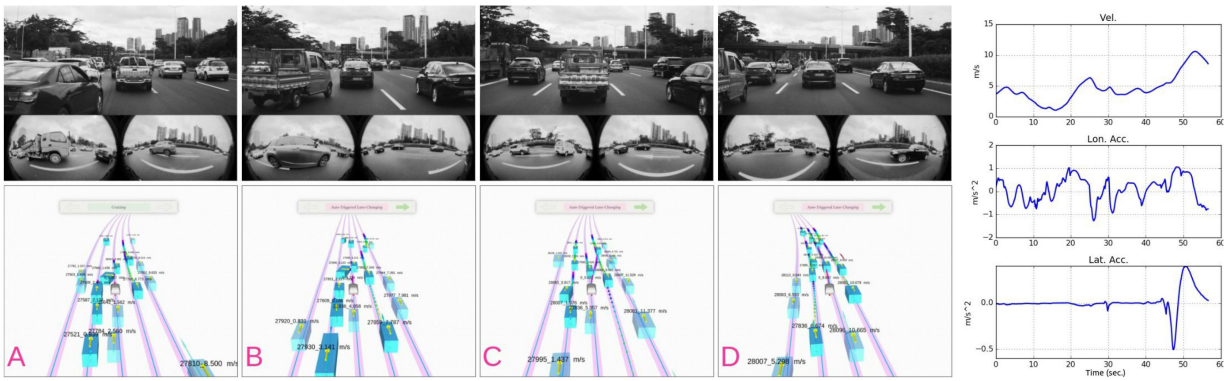
controlled vehicle is confident about conducting the acceleration maneuver since there is enough room for the backup choice to quit overtaking. In (D), the following vehicle does not further accelerate and the ego vehicle finds a safe gap to complete the merge. From this example we find that interaction-aware planning does increases the flexibility of the decision and also avoids overconservative behaviors. However, it is important to consider interaction in the envelope of safety.

3) Low-Speed Automatic Lane Change in Dense Traffic: The previous two cases are lane changes upon user requests. And in this case, we show how automatic lane change can be conducted. Automatic lane change is more challenging in low-speed dense traffic. In Fig. 16(a), the controlled vehicle tries to accelerate and conduct an active merge into the nearby lane, where the traveling efficiency is higher. The controlled vehicle considers the interaction with the following vehicle and finishes the lane change safely in (B). Then the controlled vehicle finds the nearby lane is more efficient than the current lane and proposes another lane change, as shown in (C). The controlled vehicle completes the second lane change safely in (D). From these two consecutive active lane changes, we find that, by considering interaction and equipping it with the safety mechanism, EPSILON can achieve quite flexible maneuvers in this dense traffic. Moreover, there are interesting points in this case concerning the observation noise. The observed velocity of the following vehicle in the nearby lane is subject to large error, especially when the controlled vehicle starts steering (see the attached video for details). Even under this observation noise, EPSILON can guarantee the safety and avoiding overconservative behaviors.

4) Cut-In Handling and Automatic Lane Change: In this case we show how the controlled vehicle handles the aggressive cut-in maneuvers of other traffic participants. In Fig. 16(b), the front vehicle in the nearby lane conducts an aggressive lane-change close to the controlled vehicle (A), and the controlled vehicle encountered another cut-in by a truck (B). Both cut-ins are handled smoothly and the stop-and-go process is comfortable as we can observe from the dynamic profile. Later, the controlled vehicle decides to overtake the vehicles that cut in previously since it is congested in the current lane, as shown in (C). Finally, the controlled vehicle automatically switches to a more efficient lane (D). From this case we can observe that the intention estimation and forward simulation work well in dense traffic. Cut-in maneuvers can be quickly identified and handled smoothly. Moreover, the controlled vehicle is not being too conservative and leaves space for others all the time. It also actively overtakes other vehicles for considering its own travel efficiency.



(a)



(b)

Fig. 16. Illustration of two automatic lane change scenarios in dense traffic. The visualization scheme is the same as in Fig. 15, and the difference is that the lane changes in Fig. 15 are upon user's request, while here, the lane changes are proposed by the planning system. (a) Low-speed automatic lane change in dense traffic. (b) Cut-in handling and automatic lane change.

IX. Conclusion

In this article, we presented EPSILON, an efficient planning system for highly interactive environments. EPSILON achieves flexible maneuverability while preserving computational efficiency via guided branching, inspired by domain knowledge. A safety mechanism was introduced into multiagent forward simulation to consider interaction in the envelope of safety. A novel motion planning technique using an SSC is presented to convert the decision to a safe and comfortable executable trajectory while retaining satisfactory consistency with the behavior planning. EPSILON was validated on a real vehicle in real-world dense city traffic. In real-world driving, EPSILON is neither too conservative like most traditional methods, nor overaggressive considering noisily rational traffic participants.

References

[1] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2997–3004, Jul. 2019.

[2] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, 2017, pp. 1671–1678.

[3] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. IEEE Intell. Conf. Robot. Automat.*, 2012, pp. 2061–2067.

[4] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *Proc. IEEE Intell. Vehicles Symp.*, 2017, pp. 154–159.

[5] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. IEEE Conf. Decis. Control*, 2015, pp. 835–842.

[6] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee, "Focused trajectory planning for autonomous on-road driving," in *Proc. IEEE Intell. Vehicles Symp.*, 2010, pp. 547–552.

[7] J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi, "A behavioral planning framework for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp.*, 2014, pp. 458–464.

[8] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst*, 2018, pp. 4523–4530.

[9] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *Proc. IEEE Inetll. Vehicles Symp.*, 2017, pp. 632–639.

[10] J. Ziegler *et al.*, "Making Bertha drive—An autonomous journey on a historic route," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, Summer 2014, doi: 10.1109/MITS.2014.2306552.

[11] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

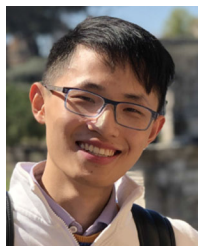
[12] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[13] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," in *Proc. IEEE 9th Int. Conf. Intell. Trans. Syst.*, 2016, pp. 1010–1016.

[14] H. Fan *et al.*, "Baidu Apollo EM motion planner," 2018, *arXiv:1807.08048*.

[15] M. Naumann, L. Sun, W. Zhan, and M. Tomizuka, "Analyzing the suitability of cost functions for explaining and imitating human driving behavior based on inverse reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat*, 2020, pp. 5481–5487.

[16] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1/2, pp. 99–134, 1998.
[17] H. Kurniawati and V. Yadav, "An online POMDP solver for uncertainty planning in dynamic environment," in *Robotics Research*. Berlin, Germany: Springer, 2016, pp. 611–629.
[18] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," *J. Artif. Intell. Res.*, vol. 58, pp. 231–266, 2017.
[19] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *Proc. Int. Conf. Intell. Trans. Syst.*, 2018, pp. 1617–1624.
[20] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 454–460.
[21] L. Zhang, W. Ding, J. Chen, and S. Shen, "Efficient uncertainty-aware decision-making for automated driving using guided branching," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3291–3297.
[22] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot., Auton. Syst.*, vol. 1, pp. 187–210, 2018.
[23] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2164–2172.
[24] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty," *Int. J. Robot. Res.*, vol. 40, no. 2/3, pp. 558–573, 2021.
[25] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 1, pp. 5–17, Mar. 2018.
[26] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 1670–1677.
[27] 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.
[28] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.
[29] 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, Mar. 2016.
[30] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1879–1884.
[31] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *Proc. IEEE Int. Conf. Robot. Automat*. IEEE, 2010, pp. 3071–3077.
[32] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4889–4895.
[33] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, 2012.
[34] M. T. Wolf and J. W. Burdick, "Artificial potential functions for high-way driving with collision avoidance," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 3731–3736.
[35] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha—A local, continuous method," in *Proc. IEEE Int. Vehicles Symp.*, 2014, pp. 450–457.
[36] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 250–256.
[37] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.
[38] S. Vaskov, U. Sharma, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, "Guaranteed safe reachability-based trajectory design for a high-fidelity model of an autonomous passenger vehicle," in *Proc. Amer. Control Conf.*, 2019, pp. 705–710.
[39] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 1758–1765.
[40] K. Leung *et al.*, "On infusing reachability-based safety assurance within planning frameworks for human-robot vehicle interactions," *Int. J. Robot. Res.*, vol. 39, no. 10/11, pp. 1326–1345, 2020.
[41] J. Zhao *et al.*, "Fp-Stereo: Hardware-efficient stereo vision for embedded applications," 2020, *arXiv:2006.03250*.
[42] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[43] P. Li, X. Chen, and S. Shen, "Stereo R-CNN based 3D object detection for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 7644–7652.
[44] M. Treiber and A. Kesting, "Traffic flow dynamics," in *Traffic Flow Dynamics: Data, Models and Simulation*, Berlin, Germany: Springer-Verlag2013.
[45] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robot. Inst., Carnegie-Mellon Univ., Pittsburgh PA, USA, CMU-RI-TR-92-01, 1992.
[46] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "PRECOG: Prediction conditioned on goals in visual multi-agent settings," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019, pp. 2821–2830.
[47] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "PiP: Planning-informed trajectory prediction for autonomous driving," in *Proc. Eur. Conf. Comput. Vision*, 2020, pp. 598–614.
[48] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017, *arXiv:1708.06374*.
[49] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. Math. Softw.*, vol. 29, no. 1, pp. 58–81, 2003.
[50] 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.*, 2015, pp. 1094–1099.
[51] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "TPNet: Trajectory proposal network for motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 6797–6806.
[52] H. He, H. Dai, and N. Wang, "UST: Unifying spatio-temporal context for trajectory prediction in autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5962–5969.
[53] M.-F. Chang *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 8748–8757.
[54] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.
[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



Wenchao Ding received the B.Eng. degree in electronic and information engineering from the Huazhong University of Science and Technology,Wuhan, China, in 2015, the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2020.

He is currently working as a Research Engineer with Huawei Technology Co., Ltd, Shenzhen, China. His research interests include decision-making, prediction, motion planning and autonomous navigation for aerial robots and autonomous vehicles.

Lu Zhang (Student Member, IEEE) received the B.Eng. and M.Eng. degrees in automation from the Beijing Institute of Technology, Beijing, China, in 2015 and 2018, respectively.

He then joined Hong Kong University of Science and Technology (HKUST) Aerial Robotics Group with the HKUST, Hong Kong, under the supervision of Prof. Shaojie Shen. His research interests cover decision-making, motion planning, and motion prediction for autonomous vehicles.

Jing Chen received the B.Eng. degree in computer science and technology from Harbin Institute of Technology, Harbin, China, in 2014, and the M.Phil. degree in robotics from the Hong Kong University of Science and Technology, Hong Kong, in 2016.

He is currently working as an Algorithm Engineer in DJI. His research interests include planning, optimization programming, mobile robot navigation.

Shaojie Shen (Student Member, IEEE) received the B.Eng. degree in electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2009, and the M.S. degree in robotics and the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2011 and 2014, respectively.

He joined the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong, in September 2014, as an Assistant Professor, and was promoted to Associate Professor, in July 2020. His research interests include robotics and unmanned aerial vehicles, with focus on state estimation, sensor fusion, computer vision, localization and mapping, and autonomous navigation in complex environments.