TUDelft

Delft University of Technology

Automated Lane Merging via Game Theory and Branch Model Predictive Control

Zhang, Luyao; Han, Shaohang; Grammatico, Sergio

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Automated Lane Merging via Game Theory and Branch Model Predictive Control

Luyao Zhang, Shaohang Han, and Sergio Grammatico, *Senior Member, IEEE*

Abstract— We propose an integrated behavior and motion planning framework for the lane-merging problem. The behavior planner combines search-based planning with game theory to model vehicle interactions and plan multivehicle trajectories. Inspired by human drivers, we model the lane-merging problem as a gap selection process and determine the appropriate gap by solving a matrix game. Moreover, we introduce a branch model predictive control (BMPC) framework to account for the uncertain equilibrium strategies adopted by the surrounding vehicles, including Nash and Stackelberg strategies. A tailored numerical solver is developed to enhance computational efficiency by exploiting the tree structure inherent in BMPC. Finally, we validate our proposed integrated planner using real traffic data and demonstrate its effectiveness in handling interactions in dense traffic scenarios.

Index Terms— Behavior planning, game theory, lane merging, model predictive control, trajectory tree.

I. Introduction

A. The Lane-Merging Problem

THE past two decades have seen prosperous progress in autonomous driving technology. While automated vehicles have demonstrated the ability to operate in various daily traffic scenarios, they still face challenges when navigating through highly interactive environments. This is primarily because automated vehicles often struggle to account for the diverse behaviors exhibited by the surrounding vehicles. In this article, we focus on the lane-merging scenario, where the automated vehicle needs to promptly find a suitable gap and understand whether the surrounding vehicles are willing to yield or not. In traditional methods, the motion predictor provides the predicted trajectory of a surrounding vehicle, indicating its driving intention, e.g., yielding. Based on these predicted trajectories, the motion planner generates a safe and comfortable trajectory for the ego vehicle (EV) [1], [2]. Since motion prediction and planning are decoupled in this hierarchical structure, it often overlooks the mutual interactions between the EV and its surroundings. Consequently, the generated motion plans can be overly conservative. Although

recently developed learning-based approaches [3], [4] consider such interactions, they inherently come with issues such as distribution shifts [5], [6] and a lack of interpretability, potentially leading to a loss of safety guarantees. In this article, we aim to develop a model-based interaction-aware planning system that integrates motion planning with motion prediction to handle the uncertain behaviors of the surrounding vehicles.

Interactive environments present two primary challenges: 1) accurately modeling vehicle interactions and 2) effectively handling the diverse behavior modes of the surrounding vehicles. To address the first challenge, researchers have proposed a game-theoretic planning framework to capture the mutual influence among multiple players. Each player in a game has its individual objective function, which depends not only on its own action but also on the actions of other players, and the goal of each player is to optimize its objective function. Previous research has extensively explored equilibrium solutions for autonomous driving. Some studies focus on jointly planning trajectories for all the vehicles by seeking a Nash equilibrium [7], [8]. Other approaches use semantic-level actions as strategies to leverage domain knowledge of the lane-merging problem. Among them, some studies investigate the Stackelberg equilibrium with a leader–follower game structure [9], [10]. In contrast to the leader–follower structure, a Nash game treats all the agents equally [8]. Several methods based on Nash games have been proposed in [11], [12], and [13], assuming that all the players adhere to one specific equilibrium. In practice, however, some players might implement other equilibrium strategies [14], which is reflected in the second challenge discussed above: uncertain behavior modes. Consequently, the EV needs to hold a belief about the strategies used by other vehicles and update the belief based on the observed trajectories. Common approaches [14], [15] in this domain often discretize the action space in line with the so-called partially observable Markov decision process (POMDP) [16]. To consider the continuous action space, branch model predictive control (BMPC) methods [17], [18] have been proposed, which use a trajectory tree with multiple branches to capture various potential behavior modes. Compared with conventional robust motion planning [19], [20], only the shared nodes in the tree need to meet safety requirements across all possible scenarios, resulting in less conservative control inputs.

B. Contribution

To overcome the two main challenges, we propose a two-layer planning framework. At the behavior planning layer,

we approach the lane-merging problem in dense traffic from a game-theoretic perspective for interaction modeling. The behavior planner computes multiple equilibrium strategies and outputs the corresponding multivehicle trajectories. Next, we develop a motion planner in the framework of BMPC to refine the trajectories generated by the behavior planner and handle the various possible equilibrium strategies of other vehicles. Our contributions with respect to the related literature are summarized as follows.

1) We model the interaction between vehicles as a two-player matrix game, where one player is the EV and the second player represents all the surrounding vehicles. Unlike the optimization-based methods in [7] and [8], we introduce a matrix game as an approximation of the original dynamic game to circumvent the issue of local solutions. To make the algorithm efficient, motivated by [21] and [22], we approximate the action space of the dynamic game using semantic-level actions and forward simulating the multivehicle system. Moreover, inspired by POMDPs, we account for the unknown cost functions by incorporating the belief about the semantic-level actions.

2) We systematically evaluate the multivehicle trajectories by solving a matrix game. A similar work EPSILON [21] forward simulates the vehicle system, but it selects the optimal trajectory using handcrafted criteria. Unfortunately, whenever the surrounding vehicles have potentially diverse behavior modes, the rule-based trajectory evaluation in EPSILON might result in overly aggressive or overly conservative trajectories [23].

A preliminary version of this work was presented in [24]. In this extended article, we present the following additional contributions.

1) We develop a BMPC framework to address the strategy misalignment between the EV and other vehicles. On the contrary, some existing methods [10], [25] simply focus on a single type of equilibrium without considering uncertainties in equilibrium strategies.

2) To achieve real-time performance, we implement a customized numerical solver that leverages the parallel structure of the trajectory tree and initialize the trajectory tree using our game-theoretic behavior planner. Instead, most BMPC approaches presented in the literature [17], [26], [27] rely on off-the-shelf nonlinear optimization solvers without exploiting the problem structure.

3) To validate the effectiveness of the proposed game-theoretic planning framework, we conduct extensive numerical simulations on real-traffic data, specifically, on the INTERACTION dataset [28]. Moreover, we compare our newly proposed planner with our previous one and other baseline methods.

C. Related Work

1) Game-Theoretic Planning: Game-theoretic methods have been applied in both motion planning and behavior planning. The game-theoretic motion planners [7], [8] break the traditional predict-and-plan pipeline by jointly planning trajectories for all the vehicles. However, these methods can only find a local equilibrium, and the quality of the solution might heavily depend on the initial guess trajectories due to the nonlinear system dynamics and nonconvex constraints. Instead, other approaches focus on the behavior planning problem, where strategies are represented by discrete semantic-level decisions. Liu et al. [9] design the strategies of the EV as motion primitives and use a leader–follower game to model the surrounding vehicles. Zhang et al. [25] and Wei et al. [10] represent the strategy of the EV as the waiting time before merging and seek a Stackelberg equilibrium. In Stackelberg games, however, determining the relative role of the leader or follower might be difficult in practice [9]. While most of the methods assume the EV to be the leader, the underlying rationale behind this choice is not always clear. In contrast to the leader–follower structure, a Nash game treats all the agents equally [8]. While there are methods based on Nash games [11], [12], [13], our approach differs by incorporating interaction-aware forward simulation. In addition to Nash and Stackelberg games, other approaches use level-k reasoning to model human driving behavior [15], [29]. However, the computational burden of this framework is substantial due to the necessity of modeling the depth of human thinking [30]. Our game-theoretic planner uses semantic-level actions, falling into the category of behavior planning.

2) POMDP: The so-called POMDP provides a mathematical framework for handling incomplete information. One line of research is dedicated to approximating original POMDP problems and enhancing computational efficiency. Online solvers, such as POMCP [31], POMCPOW [32], and DESOPT [33], have been developed, which use sampling techniques to estimate action-value functions. Other heuristic approximations of POMDPs have been introduced for specific autonomous driving applications to facilitate the search process. To generate interactive policies for lane-merging scenarios, Hubmann et al. [34] combine the Monte Carlo sampling algorithm with an A* roll-out heuristic for fast convergence. Fischer et al. [35] propose an interactive lane-merging planner that uses trained policies to guide online belief planning. In [36] and [37], the authors investigate the unsignalized intersection scenario and elaborately design the state and action space to achieve a low-dimensional problem that is solvable in practice. The reader can refer to [16] for a comprehensive survey on POMDP and its applications in robotics. In our setting, we formulate a stochastic optimal control problem that is a continuous-time counterpart to POMDPs.

3) Branch MPC: BMPC is a specific instance of stochastic optimal control. In autonomous driving applications, BMPC is used to account for the uncertain behavior modes of the surrounding vehicles. Methods for BMPC vary in their approaches to modeling the surrounding vehicles. The authors in [27] and [38] present a complete BMPC framework with belief update, enabling active information gathering. Chen et al. [17] and Wang et al. [26] construct the trajectory trees with multiple branches; however, the probability attached to each branch depends only on the current joint state, rather than the observed historical trajectories. Using interactive motion models, their methods still allow the EV to affect the motion of the surrounding ones. A further simplified version of BMPC

is proposed in [18], [39], and [40], where the motion of the surrounding vehicles is determined by the multimodal trajectories from a prediction module. These methods assume that both the future trajectory of each surrounding vehicle and the probability associated with each tree branch are fixed. Our motion planner builds upon these simplified BMPC approaches. Since the behavior planner already accounts for mutual interactions by jointly planning trajectories for all the vehicles, the motion planner, as a downstream module, disregards these interactions to achieve real-time performance.

This article is structured as follows. We introduce the lane-merging scenario and define the dynamic game problem in Section II. Section III presents the game-theoretic behavior planner. In Section IV, we first present a general formulation of the BMPC problem and subsequently elaborate on the design details. The numerical simulation results are showcased in Section V. Finally, Section VI concludes this article.

II. PROBLEM SETTING

A. Lane-Merging Scenario and General Planning Structure

We consider a mixed-traffic scenario where an automated vehicle (EV) interacts with the surrounding vehicles, as shown in Fig. 1(a). The EV attempts to execute a lane change in response to the high-level command from either the human driver [21] or the lane-selection planner [41]. To this end, its decision-making system must account for various factors, including the merging gap, the timing for lane changing, and the diverse behaviors of the surrounding vehicles. For example, in Fig. 1(a), the EV can select to merge ahead of or after SV1. If SV1 yields, then the gap enlarges and the EV merges ahead of SV1; otherwise, if the gap is not sufficiently wide, the EV might slow down and merge after SV1.

To tackle the lane-merging problem, we propose a game-theoretic planning framework, as illustrated in Fig. 2. Unlike traditional behavior planners that usually require a motion predictor as an upstream module, our approach combines motion prediction and behavior planning. The proposed game-theoretic behavior planner comprises three core modules. Specifically, we begin by enumerating the possible semantic-level decision sequences. For the EV, a semantic-level decision typically involves making a lane change, accelerating or decelerating. Then, we create the action pairs by combining the decision sequences of the EV and surrounding vehicles. For each action pair, the forward simulator generates the multivehicle trajectories, and the trajectory evaluator computes the costs associated with these trajectories. Finally, we construct a matrix game and seek multiple equilibrium strategies. The trajectory evaluator outputs the corresponding multivehicle equilibrium trajectories. Moreover, considering the uncertainty in the equilibrium strategies of the surrounding vehicles, we develop a BMPC planner to handle the multimodal behaviors exhibited by these vehicles, as shown in Fig. 2. Specifically, since the EV (in blue) is uncertain whether the nearby vehicle (in pink) will yield, it plans a trajectory tree with a shared branch that accommodates both possible decisions of the pink vehicle. Beyond the branching point, we assume the EV has full knowledge of the pink vehicle's decision.
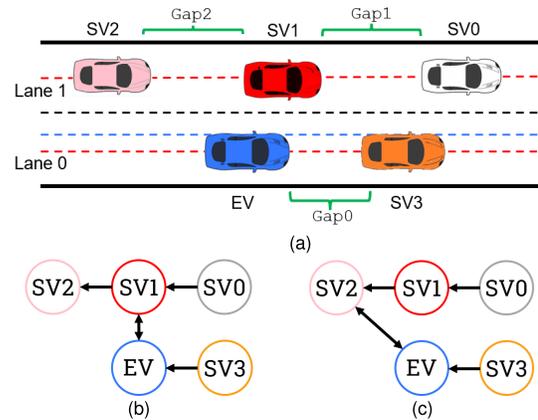

Fig. 1. (a): Lane-merging problem: three gaps are available for the EV to choose from: Gap0, Gap1, and Gap2. The dashed red lines represent the centerlines of the lanes, and the dashed blue line represents the probing line. EV stands for the ego vehicle (blue), while SV0, SV1, SV2, and SV3 denote the surrounding vehicles. (b) and (c) Interaction graphs when EV selects Gap1 and Gap2, respectively. The single-headed arrow indicates one-way influence, while double-headed arrows denote mutual interaction.
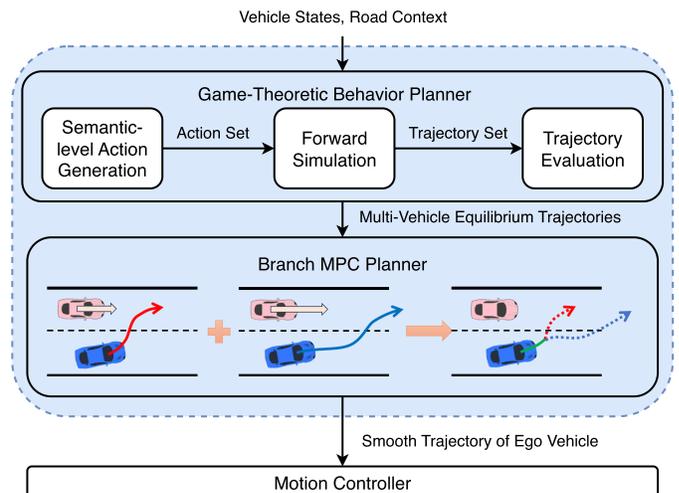
Fig. 2. Structure of the proposed game-theoretic planner. The behavior planner outputs the multivehicle trajectories that represent different equilibrium strategies. The BMPC planner uses the so-called trajectory tree to account for the potential behavior modes of the surrounding vehicles. In this example, the separate tree branches handle two distinct behavior modes (equilibrium strategies) of the IV (pink), whereas the shared part (green) needs to accommodate both the potential behavior modes.

B. Dynamic Game-Theoretic Setting

We model the vehicle interaction in the lane-merging problem using a finite horizon dynamic game with N vehicles. The dynamics of vehicle i is represented by a kinematic bicycle model [42, Section 2.2]:

$$\dot{p}_x^i = v^i \cos(\theta^i), \quad \dot{p}_y^i = v^i \sin(\theta^i), \quad \dot{\theta}^i = \frac{v^i}{l^i} \tan(\delta^i), \quad \dot{v}^i = a^i \tag{1}$$

where (p_x^i, p_y^i), θ^i, and v^i are the position, heading angle, and speed, respectively; a^i and δ^i are the acceleration and steering angle, respectively; and l^i represents the interaxle distance. The state and control input vectors are denoted by $x^i := (p_x^i, p_y^i, \theta^i, v^i)$ and $u^i := (a^i, \delta^i)$, respectively.

Applying the Runge–Kutta method [43, Section 8.2], we derive the following discrete-time dynamics:

$$x_{t+1}^i = f\left(x_t^i, u_t^i\right) \qquad (2)$$

where $t \in [0, T-1] := \{0, \ldots, T-1\}$, and $x_t^i \in \mathbb{R}^n$ and $u_t^i \in \mathbb{R}^m$ denote the state and control input vectors at time step t, respectively. For notational simplicity, the notation without a superscript refers to all the vehicles, while the notation without t refers to all the time steps. For instance, x_t represents the states of all the vehicles at time step t, while x^i denotes the state trajectory of vehicle i. Moreover, x^{-i} and u^{-i} are the stacked state and control input vectors of all the vehicles except vehicle i, respectively. In this open-loop dynamic game, each vehicle selects a control input sequence to optimize its individual objective function. Thus, we formulate the N interdependent optimization problems as follows:

$$\begin{cases} \min_{u^i} \ J^i(\boldsymbol{x}, u^i, \boldsymbol{u}^{-i}) & (3a) \\ \text{s.t. } x_{t+1}^i = f\left(x_t^i, u_t^i\right) \quad \forall t \in [0, T-1] & (3b) \\ \qquad u_t^i \in \mathcal{U}^i \quad \forall t \in [0, T-1] & (3c) \end{cases}$$

where $\mathcal{U}^i \subseteq \mathbb{R}^m$ denotes the feasible control input set, representing the bounds for acceleration and steering angle, T is the horizon, and the cost function J^i typically measures several user-defined metrics, such as progress, comfort, and safety. It depends on both the joint state trajectory and the control input sequences for each vehicle. The solution to (3) is a Nash equilibrium, where no vehicle can further reduce its cost by unilaterally changing its control input sequence. Unfortunately, directly solving (3) is challenging due to the nonconvex cost function (3a) and the nonlinear dynamics (3b). Consequently, if the initial guess is relatively far from an equilibrium point, the solution is likely to get stuck at an undesirable point. To mitigate the aforementioned issues, we use the search-based method, which can explore the solution space more extensively. Specifically, we first discretize the control input space in (3) and then formulate a matrix game with discrete action space. A matrix game can be defined by a tuple $(\mathcal{N}, \Pi, J)$, where $\mathcal{N}$ is the set of players, $\Pi = \times_{i \in \mathcal{N}} \Pi^i$ is the joint action space, and $J = (J^i)_{i \in \mathcal{N}}$ are the cost functions. In a matrix game, we seek equilibria of the following types.

Definition 1 (Pure-Strategy Nash Equilibrium): A pure-strategy Nash equilibrium is a set of players' actions, $\{\pi^{i*}\}_{i \in \mathcal{N}}$ such that for each player i, it holds that

$$J^i(\pi^{i*}, \pi^{-i*}) \le \inf_{s^i \in \Pi^i} J^i(s^i, \pi^{-i*})$$

where π^{-i} represents the set of actions taken by all the players except player i.

Definition 2 (Stackelberg Equilibrium): A Stackelberg equilibrium is a pair $\{\pi^{L*}, \pi^{F*}(\cdot)\}$ such that

$$\pi^{L*} = \arg \min_{\pi^L \in \Pi^L} J^L(\pi^L, \pi^{F*}(\pi^L))$$
$$\pi^{F*}(\pi^L) = \arg \min_{\pi^F \in \Pi^F} J^F(\pi^L, \pi^F)$$

where the superscripts L and F represent the leader and the follower of the game, respectively.

The idea behind the Stackelberg game is that the leader takes action first, and then the follower plays the best response [44]. The roles of leader and follower are not always fixed on the road. In other words, the EV can switch between being the leader and the follower [45]. Thus, we can consider two Stackelberg equilibria: one with the EV as the leader and the other with the EV as the follower.

The proposed matrix game can be viewed as a discrete approximation of (3). Compared with solving the origin problem, the solution to the matrix game is easy to compute, e.g., by enumeration. Moreover, owing to the context-aware action generation and interactive forward simulation (see Section III), the problem size is moderate.

III. GAME-THEORETIC BEHAVIOR PLANNING

In this section, we provide the technical details of the matrix game, covering aspects such as players, action generation, forward simulation, and trajectory evaluation.

A. Game Players

In principle, we need to consider both the EV and all relevant surrounding vehicles as players. This choice accounts for the interaction between each pair of vehicles but results in a complex multiplayer matrix game. To simplify the matrix game, we make several assumptions based on the inherent properties of the lane-merging problem.

Assumption 1: For each semantic-level decision, the EV mutually interacts with at most one surrounding vehicle. All other vehicle interactions are considered unidirectional.

This assumption is illustrated in Fig. 1(b) and (c). Specifically, in Fig. 1(b), the EV does not influence the vehicle (SV0) ahead of SV1, and SV2 may be affected by SV1 but does not influence the EV. It is important to note that Assumption 1 does not imply that our approach considers only one surrounding vehicle as the interacting vehicle (IV). Instead, conditioned on the semantic-level decision, we select the appropriate IV, as shown in Fig. 1(b) and (c). Two potential IVs are considered in this example. Solving the game allows the EV to decide on the merging gap and the corresponding IV. Moreover, inspired by RSS [46], we assume that the SVs with one-way interactions always take action to avoid colliding with the lead vehicle. Combining this with Assumption 1, we can treat the SVs as a vehicle group (VG). In the group, the IV owns multiple semantic-level actions, while the remaining SVs exhibit car-following behavior. To sum up, we consider the EV and the group of SVs (VG) as two players, $\mathcal{N} := \{\text{EV}, \text{VG}\}$.

Assumption 2: In the behavior planning problem, the surrounding vehicles maintain their lanes and move longitudinally [9, Section 2], [10, Section 3].

B. Semantic-Level Action Generation

One straightforward choice for the action is the control input sequence of each player, which can be obtained by discretizing the control input space. However, this approach results in large action sets and a large-scale matrix game that is potentially challenging to solve. Moreover, if we independently sample the control input space for each player, the potential interaction between players would be ignored,

and the resulting trajectories might not be realistic. Inspired by human drivers, we instead represent the action of player $i \in \mathcal{N}$ by a semantic-level decision sequence, denoted as $\pi^i := \{\pi_0^i, \ldots, \pi_k^i, \ldots, \pi_{H-1}^i\}$, where H is the decision horizon. Following this, we can now discuss the actions of the EV and the surrounding vehicles.

1) Actions of EV: In the lane-merging problem, the semantic-level decision involves selecting a gap and determining a desired lateral position. For example, in Fig. 1(a), the EV has three potential gaps to choose from. To reach the target gap, the EV needs to perform a sequence of lateral decisions. The common lateral decisions are lane changing and lane keeping. In addition, we introduce one additional intermediate lane, represented by the dashed blue line in Fig. 1(a), to enable a probing decision. This allows the EV to gather information and negotiate with the SVs. Overall, the complete lateral decision set can be defined as

$$D^{\text{lat}} := \{\texttt{LaneKeep}, \texttt{LeftChange}, \texttt{LeftProbe}\}.$$

The semantic-level decision at decision step k is denoted by an action pair $\pi_k^{\text{EV}} := (g_k, d_k^{\text{lat}})$, where $g_k \in \{\texttt{Gap0}, \texttt{Gap1}, \texttt{Gap2}\}$ and $d_k^{\text{lat}} \in D^{\text{lat}}(g_k)$. We note that the lateral decision set is conditioned on the gap selection, which reduces the number of action pairs. For example, if the EV chooses Gap0, then the only available lateral action is to keep the current lane.

Then, in line with [21], we construct a decision tree to enumerate all possible decision sequences. Each node in the tree represents a decision pair. The decision tree is rooted in the decision selected in the last planning cycle and branches out at each decision step. Due to the exponential growth of the number of decision sequences with the depth of the tree, it is necessary to prune the decision tree to limit computational complexity. Using semantic-level decisions, we can design some rules to prune the tree. For instance, we can restrict the number of decision changes over the planning horizon because human drivers tend to maintain their current driving decisions for relatively long periods. In addition, we can rule out certain transitions that would not be considered by normal human drivers, such as the transition from (Gap1, LeftChange) to (Gap2, LeftChange).

2) Actions of the Surrounding Vehicles: Next, let us postulate another working assumption.

Assumption 3: In the behavior planning problem, the surrounding vehicles keep their decisions unchanged throughout each forward simulation.

Even though this assumption limits the decision space, it remains reasonable in practice since the behavior planner runs in a receding horizon fashion. In view of Assumptions 2 and 3, we define the action set of the VG as {Assert, Yield}. Assumption 1 indicates that the IV selects an action from the set $\Pi^{\text{VG}} := \{\texttt{Assert}, \texttt{Yield}\}$ while the other vehicles in the group adhere to car-following behavior. For instance, in Fig. 1(a), if the EV selects Gap1, then it mutually interacts with SV1, and SV2 just keeps a safe distance from SV1.
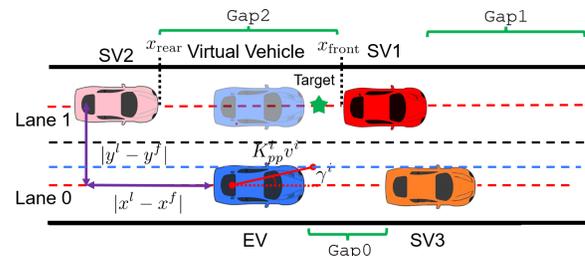

Fig. 3. Multivehicle forward simulation. The $|x^l - x^f|$ and $|y^l - y^f|$ represent the longitudinal and lateral distances between the lane-changing vehicle (leader) and the IV (follower), respectively.

C. Multivehicle Forward Simulation

We generate trajectories for multiple vehicles by simulating their motion from the initial states. Their states evolve according to (2). Next, we discuss the computation of control inputs considering vehicle interactions.

1) Motion of the EV: We apply two separate controllers to generate the longitudinal and lateral motions for the EV. For longitudinal motion, we track the target longitudinal position and the desired speed using a PD controller [21, Section 7.B]. One example of the target longitudinal position within the desired gap is illustrated in Fig. 3. The desired gap is defined based on the positions of the front and rear vehicles, denoted as x_{front} and x_{rear}, respectively. Similar to [21, Section 7.B], [46, Section 3], we determine the target longitudinal position and speed based on the safe distance that the EV and the rear vehicle (shown in pink in Fig. 3) need to keep. The output of the PD controller is represented by a_{pd}. However, this controller does not consider the lead vehicle and the static obstacle. Hence, to avoid a collision, we compute the car-following acceleration a_{idm} using the intelligent driver model (IDM) [47]. Overall, for the sake of safety, we select the minimum acceleration as the command, $a_{\text{cmd}} = \min(a_{\text{pd}}, a_{\text{idm}})$.

As for lateral motion, we adopt a pure pursuit controller that takes the current vehicle speed and the target line as inputs. The steering angle is computed by $\delta_{\text{cmd}}^i = \tan^{-1}(2l^i \sin(\gamma^i)/(K_{\text{pp}}^i v^i))$, where γ^i represents the angle between the heading direction and the lookahead direction, K_{pp} is the feedback gain, and $K_{\text{pp}}^i v^i$ is the lookahead distance.

2) Motion of the Surrounding Vehicles: To model car-following behavior and reactions to lane-changing vehicles, we propose a modified IDM. This model considers lane-changing vehicles by projecting them onto their target lanes, resulting in virtual vehicles as shown in Fig. 3. The only difference between the standard IDM and our modified version lies in the computation of the distance between the leader and the follower, which is given by

$$d_{\text{idm}} = |x^l - x^f|e^{\kappa|y^l - y^f|}, \quad \kappa = 2\log(\beta)/w_{\text{lane}} \quad (4)$$

where w_{lane} is the lane width, and β is a parameter characterizing the level of willingness to yield. By adjusting the value of β, we can represent different actions performed by the VG. Specifically, a large value of β indicates that the vehicle on the target lane is less likely to yield to the lane-changing vehicle because it perceives that the projection is far away. When the lateral distance between two vehicles vanishes, that is,

$|y^l - y^f| = 0$, the virtual distance between them is equivalent to the actual longitudinal distance.

D. Trajectory Evaluation

1) Cost Function: Once multivehicle trajectories are generated, we proceed to select a specific action pair by solving a matrix game. To construct the cost matrix, we introduce the cost function J^i of vehicle i, which is a combination of several user-defined metrics, including safety, efficiency, comfort, navigation, and information cost: $J^i = J^i_{\text{saf}} + J^i_{\text{eff}} + J^i_{\text{com}} + J^i_{\text{nav}} + J^i_{\text{inf}}$. The value of the cost function J^i depends on the simulated trajectories, which are influenced by the semantic-level action of the EV, π^{EV}, and the VG, π^{VG}. We consider the SVs as a VG by calculating the total cost as $J^{\text{VG}} := \sum_{i \neq \text{EV}} J^i$.

We calculate the safety cost by examining vehicle collisions. The footprint of vehicle i is modeled as a rectangle. If the distance $d^{ij}(t)$ between two rectangles is less than a small value $\underline{d}$, indicating a potential collision, we assign a substantial penalty w^{saf}_1 to the corresponding trajectory. In addition, we encourage the vehicle to keep a suitable distance from its surroundings by giving a relatively small penalty w^{saf}_2 when $d^{ij}(t)$ falls within the range of $[\underline{d}, \bar{d}]$. Next, we measure the efficiency of the trajectory by computing the sum of the squares of the differences between the vehicle speed and its desired speed: $J^i_{\text{eff}}(\pi^{\text{EV}}, \pi^{\text{VG}}) := w^{\text{eff}} \sum_{t=0}^{T} (v^i(t) - v^i_{\text{des}})^2$. For the comfort cost, we consider the acceleration change, commonly called jerk. We use the finite difference to approximate the jerk, and subsequently define the comfort cost: $J^i_{\text{com}}(\pi^{\text{EV}}, \pi^{\text{VG}}) := w^{\text{com}} \sum_{t=1}^{T} (a^i(t) - a^i(t-1))^2 / \Delta t^2$. Next, we penalize the differences between the vehicle's lateral position and its desired lateral position to encourage the lane-changing maneuver: $J^i_{\text{nav}}(\pi^{\text{EV}}, \pi^{\text{VG}}) := w^{\text{nav}} \sum_{t=0}^{T} (y^i(t) - y^i_{\text{des}})^2$. In addition, we introduce an information gain metric in the cost function, inspired by [48], to motivate the EV to actively identify the intentions of other vehicles. Informally speaking, the information cost encourages the EV to take the `probing` action.

2) Belief Update: The EV needs to estimate the cost functions of the SVs by observing their trajectories since direct access to these cost functions is impractical. Inspired by POMDP, we account for uncertainty in the aggregate cost of the SVs by integrating the beliefs into the cost function. The modified aggregate cost is computed as follows:

$$\bar{J}^{\text{VG}}_{ij} := \left(1 - b\left(\pi^{\text{VG}}_i\right)\right) J^{\text{VG}}_{ij}, \quad \sum_{i=1}^{M^{\text{VG}}} b\left(\pi^{\text{VG}}_i\right) = 1$$

where i and j represent the indices in the cost matrix, $\pi^{\text{VG}}_i \in \Pi^{\text{VG}}$ is the action of the VG, $M^{\text{VG}} := |\Pi^{\text{VG}}|$ denotes the cardinality of the action set of the VG, and b represents the belief about the VG's action. This design can be understood as incorporating prior knowledge about the behavior of the VG into the aggregate cost. For example, if we have prior knowledge suggesting that the VG is inclined to yield, then we can set the corresponding belief close to 1, which reduces the aggregate cost.

The belief is recursively updated at the beginning of each planning cycle using Bayes filtering [49]. The update rule involves two key components: the transition model $\mathbb{P}(\pi^{\text{VG}} | \pi^{\text{VG}}_-)$ and the observation model $\mathbb{P}(o | \pi^{\text{VG}})$, where the notation $\square_-$ is associated with the previous planning cycle, and o denotes the observed state. Assuming that the IV in the group does not change its action, the transition model can be significantly simplified. In the simplified model, $\mathbb{P}(\pi^{\text{VG}} = \pi^{\text{VG}}_j | \pi^{\text{VG}}_i)$ is equal to 1 if i and j coincide, and for all other cases, $\mathbb{P}(\pi^{\text{VG}} = \pi^{\text{VG}}_j | \pi^{\text{VG}}_i)$ is equal to 0.

In the observation model, we use the dynamics (2) with additive Gaussian noise to predict the IV's state in the group

$$x^{\text{VG}}_{t+1} = f\left(x^{\text{VG}}_t, u^{\text{VG}}_t\left(\boldsymbol{x}_t, \pi^{\text{VG}}_t, \pi^{\text{EV}}_t\right)\right) + w, \quad w \sim \mathcal{N}(0, W)$$

where $u^{\text{VG}}_t(\cdot)$ denotes the control policy discussed in Section III-C2, and w is additive Gaussian noise with zero mean and covariance matrix W. Then, we compute the likelihood of receiving an observation o using the following distribution:

$$\mathbb{P}\left(o | \pi^{\text{VG}}\right) \sim \mathcal{N}\left(f\left(x^{\text{VG}}_-, u^{\text{VG}}_-\left(\boldsymbol{x}_-, \pi^{\text{VG}}_-, \pi^{\text{EV}}_-\right)\right), W\right)$$

where the mean is the predicted state. With the transition model and the observation model, the belief update rule can be expressed as follows:

$$b(\pi^{\text{VG}}) = \frac{\mathbb{P}\left(o | \pi^{\text{VG}}\right) \sum_{\pi^{\text{VG}}_-} \mathbb{P}\left(\pi^{\text{VG}} | \pi^{\text{VG}}_-\right) b\left(\pi^{\text{VG}}_-\right)}{\sum_{\pi^{\text{VG}}} \mathbb{P}\left(o | \pi^{\text{VG}}\right) \sum_{\pi^{\text{VG}}_-} \mathbb{P}\left(\pi^{\text{VG}} | \pi^{\text{VG}}_-\right) b\left(\pi^{\text{VG}}_-\right)}.$$

3) Equilibrium Selection: With the cost matrix, we compute multiple equilibria, such as Nash and Stackelberg equilibria, by enumerating all possible combinations of semantic-level actions. If multiple Nash equilibria exist, we select the equilibrium with the lowest social cost [50], defined as $\mathcal{C}_{ij} := J^{\text{EV}}_{ij} + J^{\text{VG}}_{ij}$. If a pure-strategy Nash equilibrium does not exist, we choose the Stackelberg equilibrium [10], [25] with the EV as the follower as a backup solution.

IV. BRANCH MODEL PREDICTIVE CONTROL

The motion planner aims at refining the coarse trajectory generated by the behavior planner. We model the motion planning problem as a stochastic optimal control problem due to the unknown behavior modes of the surrounding vehicles. In this section, we first introduce the vehicle modeling and then the general formulation of the interaction-aware motion planning problem. Subsequently, we introduce some simplifications to rapidly compute an approximate solution.

A. Vehicle Modeling

1) Dynamics: For simplicity, we consider two vehicles in this section: the EV and one IV, with the states denoted as x^{EV} and x^{IV}, respectively. The joint state is represented as $\boldsymbol{x} := (x^{\text{EV}}, x^{\text{IV}})$. We model the vehicle dynamics by a discrete-time kinematic bicycle, where $i \in \{\text{EV}, \text{IV}\}$, $x^i := (p^i_x, p^i_y, \theta^i, v^i)$ and $u^i := (a^i, \delta^i) \in \mathcal{U}^i$ as in (2). $\mathcal{U}^i$ represents the feasible control input set.

2) Behavior Model: The interactive behavior of the IV can be modeled using the feedback policy $\kappa : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$, where Ω is a finite set with each element corresponding to one behavior mode of the IV. The control input of the IV at time step t is determined by both the joint state $\boldsymbol{x}_t$ and the behavior mode $\omega_t \in \Omega$, that is,

$$u_t^{\text{IV}} = \kappa(\boldsymbol{x}_t, \omega_t), \quad \omega_t \sim b_t := \mathbb{P}(\omega_t | O_t) \tag{5}$$

where b_t, known as the belief state, is the distribution of ω_t conditioned on the observation O_t. The observation is a collection of the observed joint states $O_t := [\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_0]$. Similar to [9], the EV updates the belief state based on Bayes filtering after obtaining a new observation. The belief update, also referred to as belief dynamics [27], can be briefly expressed as $b_{t+1} = g(b_t, \boldsymbol{x}_{t+1}, u_t^{\text{EV}})$. In general, it is commonly believed that forward propagating the belief state analytically is intractable due to the nonlinear vehicle dynamics [27], [38]. Therefore, some approximations become necessary in practice, as shown in Section IV-C.

B. General Formulation

With all these ingredients, we can now formulate the interaction-aware motion planning problem in the framework of stochastic optimal control

$$\min_{(\mu_t^{\text{EV}})_{[0,T-1]}} \mathbb{E}_{\substack{\omega_t \sim b_t \\ t \in [0,T-1]}} \left\{ \sum_{t=0}^{T-1} \ell(\boldsymbol{x}_t, u_t^{\text{EV}}) + \ell_f(\boldsymbol{x}_T) \right\} \tag{6a}$$

$$\text{s.t. } \boldsymbol{x}_0 = \bar{\boldsymbol{x}}, \quad b_0 = \bar{b} \tag{6b}$$

$$\forall t \in [0, T-1]$$

$$x_{t+1}^{\text{EV}} = f(x_t^{\text{EV}}, u_t^{\text{EV}}), \quad u_t^{\text{EV}} = \mu_t^{\text{EV}}(\boldsymbol{x}_t, b_t) \tag{6c}$$

$$x_{t+1}^{\text{IV}} = f(x_t^{\text{IV}}, u_t^{\text{IV}}), \quad u_t^{\text{IV}} = \kappa(\boldsymbol{x}_t, \omega_t) \tag{6d}$$

$$b_{t+1} = g(b_t, \boldsymbol{x}_{t+1}, u_t^{\text{EV}}) \tag{6e}$$

$$u_t^{\text{EV}} \in \mathcal{U}^{\text{EV}} \tag{6f}$$

$$\forall t \in [0, T] : \quad h(\boldsymbol{x}_t) \leq 0 \tag{6g}$$

where $(\mu_t^{\text{EV}})_{[0,T-1]}$ is a sequence of feedback control policies, $\bar{\boldsymbol{x}}$ and $\bar{b}$ are the initial joint state and belief, respectively, $\mathcal{U}^{\text{EV}} \subseteq \mathbb{R}^m$ is the set of feasible control inputs, and (6g) represents the collision avoidance constraints. We defer the discussion on the feasible control input set (6f) and safety constraint (6g) to Section IV-D. An alternative to hard collision avoidance constraints is a chance constraint formulation, which ensures safety in a probabilistic manner and reduces conservatism at the price of higher computational effort [19], [26]. Thus, to achieve a balance between robustness and computational demands, we adopt soft collision avoidance constraints in practice [51] and in turn sacrifice strict probabilistic guarantees.

C. Trajectory Tree

Now, let us present an approximation of the general interaction-aware motion planning problem. In a numerical optimal control problem, we often look for a sequence of control inputs $(u_t^{\text{EV}})_{[0,T-1]}$ rather than a sequence of feedback policies $(\mu_t^{\text{EV}})_{[0,T-1]}$ to reduce the computational complexity.
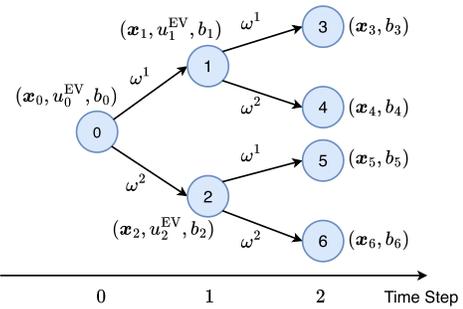

Fig. 4. Trajectory tree with a horizon of $T = 2$. The tree branches out at each time step based on the behavior mode $\omega_t \in \Omega := \{\omega^1, \omega^2\}$. A transition probability P_i pertains to each branch. In this example, $\mathcal{N} = \{0, 1, 2, 3, 4, 5, 6\}$ and $\mathcal{L} = \{3, 4, 5, 6\}$.

Nevertheless, a sequence of control inputs that solve (6) might be overly conservative or even not exist in some cases due to the requirement that constraints must be respected across all realizations of uncertainty. Another primary drawback of this approximation is neglecting the advantages that can be gained from future observations. To resolve this, we instead seek a trajectory tree where all possible realizations of uncertainty are enumerated. As shown in Fig. 4, the tree is rooted at the node of the current joint state and branches out at each time step based on the behavior mode. We represent the set of all the tree nodes as $\mathcal{N}$ and the set of all leaf nodes as $\mathcal{L}$. The transition probability from a parent node p(i) to its child node i is denoted by P_i. In contrast to a single control sequence, distinct control inputs at each predicted time step are derived based on different behavior modes. Using the trajectory tree, we reformulate the stochastic optimal control problem (6) as follows:

$$\min_{(u_i^{\text{EV}})_{i \in \mathcal{N} \setminus \mathcal{L}}} \sum_{i \in \mathcal{L}} P_i \ell_f(\boldsymbol{x}_i) + \sum_{i \in \mathcal{N} \setminus \mathcal{L}} P_i \ell(\boldsymbol{x}_i, u_i^{\text{EV}}) \tag{7a}$$

$$\text{s.t. } \boldsymbol{x}_0 = \bar{\boldsymbol{x}}, \quad b_0 = \bar{b} \tag{7b}$$

$$\forall i \in \mathcal{N} \setminus \{0\} \tag{7c}$$

$$x_i^{\text{EV}} = f(x_{\text{p}(i)}^{\text{EV}}, u_{\text{p}(i)}^{\text{EV}}) \tag{7d}$$

$$x_i^{\text{IV}} = f(x_{\text{p}(i)}^{\text{IV}}, u_{\text{p}(i)}^{\text{IV}}), \quad u_{\text{p}(i)}^{\text{IV}} = \kappa(\boldsymbol{x}_{\text{p}(i)}, \omega_{\text{p}(i)}) \tag{7e}$$

$$b_i = g(b_{\text{p}(i)}, \boldsymbol{x}_i, u_{\text{p}(i)}^{\text{EV}}) \tag{7f}$$

$$\forall i \in \mathcal{N} \setminus \mathcal{L} : \quad u_i^{\text{EV}} \in \mathcal{U}^{\text{EV}} \tag{7g}$$

$$\forall i \in \mathcal{N} : \quad h(\boldsymbol{x}_i) \leq 0 \tag{7h}$$

where objective function (7a) computes the expected cost, and constraints (7d) to (7f) describe the evolution of the states and belief state between two nodes. Constraints (7g) and (7h) are associate with control limits and safety constraints.

While the trajectory tree in (7) is a reasonable approximation of a sequence of feedback policies, it still entails several practical challenges. First, the introduction of additional optimization variables results in an increased computational burden. In addition, as the optimization problem is nonconvex, the quality of the computed solution is heavily dependent on the initial guess. Providing an initial trajectory tree is also a nontrivial task. Taking the aforementioned facts into account, we further simplify the trajectory tree as follows.
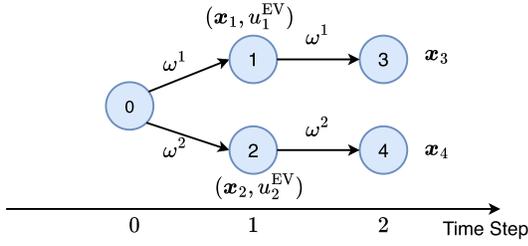
Fig. 5. Simplified trajectory tree. The tree has multiple branches exclusively at the root node, while all other nodes, except for the leaf nodes, have only one single branch.

1) The trajectory tree only branches out once, which assumes that the behavior mode of the IV remains constant over the planning horizon and the EV is fully certain about the behavior mode of the IV after the first branch. As a result, we do not consider the belief update in (7f) over the planning horizon. In addition, our behavior planner can easily initialize such a simplified tree, enhancing the quality of the computed solution.

2) The behavior planner provides the multimodal state trajectories of the IV, which can be viewed as an open-loop control policy. Compared with using feedback control policy in (5), we ignore the mutual interaction in motion planning since it has been taken into account by the behavior planner. We note that the simplified tree shares some similarities with those found in QMDP [49] and contingency planning [18], [52].

Fig. 5 shows an example of the simplified trajectory tree with two branches. We can now discuss the formulation in more detail.

D. Detailed Formulation

1) Branches: Our behavior planner generates multiple multivehicle trajectories associated with distinct equilibrium strategies. Given that the IV might be uncertain in terms of equilibrium strategies, we use the branch MPC formulation to accommodate all the potential equilibrium strategies. In this formulation, each tree branch represents an equilibrium strategy, such as a Nash or Stackelberg strategy.

2) Cost Function: The motion planner tracks the reference trajectories generated by the behavior planner while maximizing the comfort level. Thus, we use the following stage cost:

$$\ell := \ell_{\text{track}} + \ell_{\text{com}}$$

$$\ell_{\text{track}}\left(x_i^{\text{EV}}, u_i^{\text{EV}}\right) := \left\| x_i^{\text{EV}} - x_i^{\text{EV, ref}} \right\|_Q^2 + \left\| u_i^{\text{EV}} - u_i^{\text{EV, ref}} \right\|_R^2$$

$$\ell_{\text{com}}\left(u_i^{\text{EV}}, u_{\text{p}(i)}^{\text{EV}}\right) := \left\| u_i^{\text{EV}} - u_{\text{p}(i)}^{\text{EV}} \right\|_{R_{\text{com}}}^2 \quad \forall i \in \mathcal{N}$$

where $R_{\text{com}} \succ 0$, $Q \succeq 0$, and $R \succ 0$ represent the weight matrices for comfort, state, and control input, respectively; and we define $u_{\text{p}(0)}^{\text{EV}}$ as the executed control command from the last planning cycle. The penalty term on the change rate of the control input plays an important role in enhancing driving comfort. By penalizing $\|u_0^{\text{EV}} - u_{\text{p}(0)}^{\text{EV}}\|$, we reduce the executed control variation between two planning cycles.

3) State and Input Constraints: To ensure physical feasibility, we impose bounds on the state and control input

$$x_{\text{lb}} \le x_i^{\text{EV}} \le x_{\text{ub}} \quad \forall i \in \mathcal{N}, \quad u_{\text{lb}} \le u_i^{\text{EV}} \le u_{\text{ub}} \quad \forall i \in \mathcal{N} \backslash \mathcal{L}.$$

4) Collision Avoidance: The footprint of a vehicle can be naturally represented by a rectangle. Nevertheless, one challenge associated with this representation is the complexity of deriving a closed-form signed distance. Moreover, the gradient of the signed distance function is discontinuous, posing a challenge to the optimization solver. To circumvent these difficulties, we use an overapproximation of the shape of a vehicle that comprises the union of a collection of linked circles [26]. The smooth collision avoidance constraints (7h) can be formulated as follows:

$$h(\boldsymbol{x}) = (r^{\text{EV}} + r^{\text{IV}})^2 - \left\| c_i(x^{\text{EV}}) - c_j(x^{\text{IV}}) \right\|_2^2 \le 0$$
$$i \in \left[1, n_c^{\text{EV}}\right], \quad j \in \left[1, n_c^{\text{IV}}\right] \quad (8)$$

where r^{EV} and r^{IV} represent the disk radii, $c : \mathbb{R}^n \to \mathbb{R}^2$ is a function that computes the center of the disk, and n_c^{EV}, n_c^{IV} denote the number of disks used for approximating the vehicle footprints. In addition to the IV, we enforce the same dynamic collision avoidance constraints on the other vehicles using the simulated trajectories provided by the behavior planner.

V. NUMERICAL SIMULATIONS

A. Benchmarking

We validate our proposed game-theoretic planner using the INTERACTION dataset [28]. We consider the on-ramp merging scenario shown in Fig. 6, where the EV must complete the lane change before reaching the end of the current lane. Considering various factors, such as traffic speeds, vehicle sizes, and behavior modes, we select 100 test scenarios from the dataset and mark the EV for each track. Each test scenario on the dataset has a duration of $T_{\text{tr}} = 4$ s with a discretization step of $\Delta t = 100$ ms, and the number of timestamps is $N_{\text{ts}} = T_{\text{tr}}/\Delta t$. We control the EV using our proposed planner, while the motion of the surrounding vehicles is simulated either by replaying trajectory data (nonreactive mode) or using the IDM (reactive mode). We compare the proposed GT-BMPC with five baseline methods.

1) NE-MPC [24], combining a Nash equilibrium-seeking behavior planner with a traditional MPC motion planner [53]. This method does not consider the strategy misalignment.

2) SE-MPC [10], [25], which searches for the Stackelberg equilibrium in the behavior planner and then uses a traditional MPC for motion planning.

3) Y-MPC, which comprises a behavior planner and a traditional MPC motion planner. The behavior planner assumes that the VG always decides to yield.

4) PGame [54], a finite potential game framework, using sampling-based trajectories [55] as actions.

5) iLQGame [7], which jointly plans Nash equilibrium trajectories for all the vehicles.

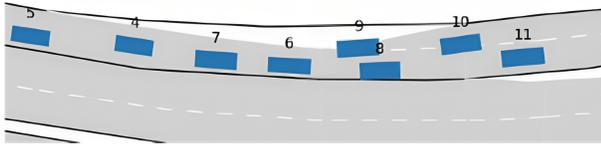The quality of the trajectories is evaluated using the following metrics.

Fig. 6. On-ramp merging scenario: vehicle 9 intends to reach the gap between vehicles 6 and 8. Since the gap is not sufficiently large, vehicle 9 has to identify the intention of vehicle 8 and make an appropriate decision.

Fig. 7. (a) Box plot of TTC_{traj}. We limit TTC_{traj} to 8 s for visualization. TTC_{traj} represents the minimum TTC values among all the timestamps of one pair of trajectories. To obtain it, we first calculate TTC_k at timestamp k. In a lane-merging scenario shown in Fig. 6, the EV (vehicle 9) selects the gap between the leading vehicle (LV, vehicle 6) and the following vehicle (FV, vehicle 8). To consider both the surrounding vehicles, we calculate two TTC values at each timestamp: one between the EV and the LV as $\text{TTC}_{\text{EV-LV},k}$, and another between the FV and the EV as $\text{TTC}_{\text{FV-EV},k}$. Next, we choose $\text{TTC}_k = \min\{\text{TTC}_{\text{EV-LV},k}, \text{TTC}_{\text{FV-EV},k}\}$. After calculating TTC_k at all the timestamps, we select the minimum from $\{\text{TTC}_k\}$ as TTC_{traj}. The triangle marker represents the mean value. (b) Box plot of root-mean-squared heading acceleration.

1) We assess the safety level via collision rate and time-to-collision (TTC) [29], [56]. The collision rate is defined as the proportion of the number of cases, in which a collision happens, to the total number of cases. TTC refers to the time it takes for two vehicles to collide if they maintain their current speed and heading. Note that akin to those in [56] and [57], TTC used here accounts for both the longitudinal and lateral directions.

2) Regarding progress, we project the vehicle position at the final timestamp $p_{N_{\text{ts}}}$ onto the target lane to determine the lateral distances $d_{N_{\text{ts}}}$. A smaller lateral distance indicates better lateral progress.

3) We use the root-mean-squared absolute jerk, maximum absolute jerk, and root-mean-squared angular acceleration as metrics to measure comfort level. With the velocity profile of a track, we numerically approximate jerk at one timestamp as $|(v_{k-1}^{\text{EV}} - 2\,v_k^{\text{EV}} + v_{k+1}^{\text{EV}})/\Delta t^2|$. Similarly, the approximated angular acceleration is calculated by $|(\theta_{k-1}^{\text{EV}} - 2\theta_k^{\text{EV}} + \theta_{k+1}^{\text{EV}})/\Delta t^2|$, where θ is the heading angle.

4) We use average displacement error (ADE) to measure the distance between the generated trajectory and the ground truth [21]. Specifically, $\text{ADE} := \sum_{k=1}^{N_{\text{ts}}} \|p_k^{\text{EV}} - p_k^{\text{EV, GT}}\|/N_{\text{ts}}$, where $p_k^{\text{EV, GT}}$ denotes the ground-truth position of the EV.

B. Implementation Details

1) Game-Theoretic Behavior Planner: We use a planning horizon of $T_{\text{bp}} = 25$, a discretization step of $\Delta t_{\text{bp}} = 0.2$ s, a decision time period of $\Delta h = 1$ s, and a decision horizon of $H = 5$. In other words, the behavior planner looks ahead for 5 s and makes a semantic-level decision every 1 s. We run the behavior planner at 5 Hz in a receding horizon fashion.

We select the vehicle nearest to the EV as the target vehicle (SV1). The vehicles ahead of and behind the target vehicle are denoted as SV0 and SV2, respectively. The gap in front of the target vehicle is denoted as Gap1, while the gap behind is referred to as Gap2, as shown in Fig. 1(a). We label the current lane where the EV stays as Gap0. As discussed in Section III-B, there are three desired lateral positions: two center lines and one probing line.

We consider the vehicle ahead of the EV as a dynamic obstacle, indicating that the EV intends to maintain a safe distance from it. The surrounding vehicles, SV0, SV1, and SV2 are treated as a VG. The potential IVs in this group are SV1 and SV2. We assume that the IV has two longitudinal actions: assert and yield. To represent the actions, we supply distinct parameters to the modified IDM, which
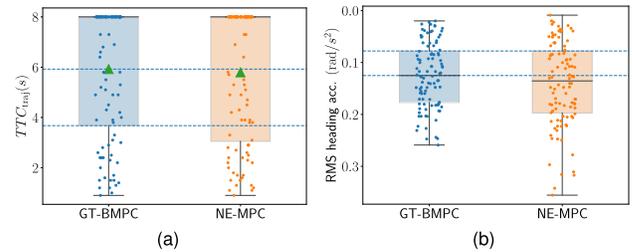
controls the motion of the surrounding vehicle. For instance, to model a yielding vehicle, in addition to the willingness indicator β in (4), we set the minimum spacing and desired time headway to relatively large values.

2) BMPC: The motion planner operates at 10 Hz with a discretization step of 0.1 s and a planning horizon of 4 s. The reference trajectories for the ego vehicle, along with the probability associated with each branch, are provided by the behavior planner. To exploit the sparse structure of the optimal control problem, we develop a tailored BMPC solver by extending the iLQR solver [58] to its trajectory tree version, known as iLQR-tree [23], [39]. We handle state and input constraints using the augmented Lagrangian relaxation [53].

3) Computational Performance: All the simulations are conducted on a laptop with a 2.30-GHz Intel Core i7-11800H processor and 16-GB RAM. We implement both behavior planner and motion planner in C++. The behavior planner requires approximately 10.5 ms to generate the multivehicle trajectories. The real-time performance can be attributed to two factors.

1) Sampling the semantic-level action space, rather than the control input space, resulting in fewer samples.

2) Parallelizing the process of forward simulation using the OpenMP API.

Regarding the motion planner, the average computation time for the BMPC problem is 1.11 ms, while the peak computation time reaches 7.23 ms. We further enhance the computational efficiency by computing the backward sweeps from each leaf node to the shared node in parallel via OpenMP API. As a result, we achieve a reduction of 9.9% in the mean computation time (1.00 ms) and a decrease of 19.2% in the maximum computation time (5.84 ms). The code is publicly available at: https://github.com/SailorBrandon/GT-BMPC.

C. Statistical Results

We compare the proposed planner with the baseline planners across 100 test scenarios. Table I shows the mean values of the considered metrics for these scenarios.

TABLE I
STATISTICAL RESULTS

| Metric	Nonreactive						Reactive					
	GT-BMPC	NE-MPC	SE-MPC	Y-MPC	PGame	iLQGame	GT-BMPC	NE-MPC	SE-MPC	Y-MPC	PGame	iLQGame
Collision Rate (%)	0	0	1	1	6	5	0	0	0	0	2	1
Lateral Progress (m)	1.21	1.20	1.13	1.13	0.78	0.92	1.09	1.07	0.97	0.96	0.55	0.60
RMS Abs. Jerk (m/s^3)	0.21	0.21	0.21	0.20	0.42	1.45	0.24	0.24	0.24	0.24	0.39	1.12
Max Abs. Jerk (m/s^3)	0.52	0.52	0.49	0.48	5.77	2.97	0.60	0.60	0.57	0.56	4.74	2.98
RMS Heading Acc. (rad/s^2)	0.12	0.14	0.13	0.12	0.14	0.21	0.15	0.17	0.15	0.15	0.14	0.16
ADE (m)	0.71	0.70	0.72	0.75	0.91	1.19	N/A	N/A	N/A	N/A	N/A	N/A
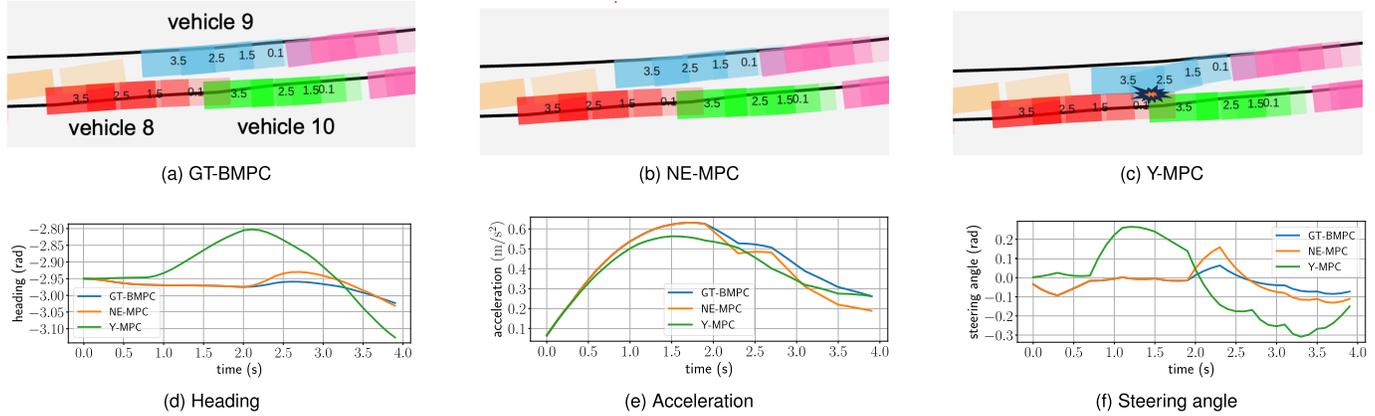
Fig. 8. Simulation results of the selected scenario. (a)–(c): Closed-loop trajectories. The ego vehicle is highlighted in blue, while potential IVs are marked in red and green. The orange vehicle is treated as a dynamic obstacle, and non-IVs are marked in pink. The number on each vehicle represents a simulation timestamp in seconds. (d)–(f): Motion profiles of the ego vehicle. The abrupt changes in the plots are likely attributed to replanning. The control commands generated by GT-BMPC are smoother, leading to a more comfortable trajectory.

1) Nonreactive Simulation: Both GT-BMPC and NE-MPC operate safely across all the test scenarios due to the consideration of diverse driving behaviors, whereas the others do not. Since Y-MPC, SE-MPC, and PGame tend to assume that the IV behaves cooperatively, collisions are likely to occur if the vehicle fails to yield. Moreover, we compute TTC_{traj} to evaluate the safety level further, and the statistical results are illustrated in Fig. 7(a). The mean and lower quartile of GT-BMPC exceed those of NE-MPC, indicating that GT-BMPC generates trajectories with a larger safety margin across the majority of test scenarios. GT-BMPC shows slightly reduced lateral progress because it often plans a probing trajectory to account for the various potential behaviors of the surrounding vehicles in the beginning. Such a motion plan can enhance safety in certain situations (see Section V-D). Regarding comfort assessment, all the planners, except for PGame and iLQGame, achieve comparable performance. Fig. 7(b) illustrates that GT-BMPC outperforms NE-MPC with respect to heading acceleration. This improvement is attributed to the ability of the trajectory tree to mitigate the issue arising from abrupt changes in semantic-level actions between two planning cycles. Finally, the low ADE values achieved by GT-BMPC, NE-MPC, and SE-MPC suggest that the ego vehicle controlled by these planners exhibits more human-like driving behavior. To sum up, our proposed planner balances various metrics and plans safer trajectories, especially when the behavior modes of surrounding vehicles are time-varying, which will be further discussed in Section V-D.

2) Reactive Simulation: Collision rates decrease significantly due to the reactive behavior of the surrounding vehicles, and all the planners achieve improved lateral progress. In addition, Y-MPC and SE-MPC perform well in reactive simulations, as their assumptions about the cooperative behavior of the surrounding vehicles are satisfied.

D. Case Study

We choose one interesting scenario to demonstrate the effectiveness of the integration of the game-theoretic behavior planner and the BMPC framework. The simulation results are illustrated in Fig. 8. The ego vehicle (vehicle 9) faces two major challenges in this scenario. First, the feasible driving space is limited due to the narrow environment. Second, the actual behavior mode of vehicle 10 is variable. Specifically, vehicle 10 decelerates during the initial 1 s but speeds up afterward. Consequently, all the ego vehicles in Fig. 8 decide to change the lane in the beginning but finally cancel the lane change. In Fig. 8(c), as the ego vehicle does not realize that merging is dangerous in time, a collision with vehicle 10 becomes unavoidable. On the contrary, the closed-loop trajectories generated by GT-BMPC and NE-MPC are collision-free due to the consideration of multimodal behaviors. We observe that the distance between the ego vehicle and vehicle 10 at 3.5 s in Fig. 8(a) is larger than that in Fig. 8(b), and the corresponding TTC_{traj} values in Table II further affirm that GT-BMPC outputs a safer

TABLE II
CASE STUDY

Metric	GT-BMPC	NE-MPC	Y-MPC
Collision-free	✓	✓	✗
TTC_{traj} (s)	4.10	2.50	N/A
RMS Abs. Jerk (m/s^3)	0.23	0.25	N/A
Max Abs. Jerk (m/s^3)	0.61	0.61	N/A
RMS Heading Acc. (rad/s^2)	0.09	0.18	N/A

trajectory, as it allows the EV to safely return to the original lane by adhering to another equilibrium strategy. Furthermore, GT-BMPC outperforms NE-MPC in terms of rms absolute jerk and RMS heading acceleration.

VI. CONCLUSION

In this article, we present a game-theoretic planning framework to address lane-merging problems in interactive environments. Our approach explicitly models the interaction between vehicles using a matrix game and tackles the issue caused by the uncertain behaviors of the surrounding vehicles via BMPC. Our validation study on the INTERACTION dataset indicates the necessity of interaction modeling and demonstrates the superior performance of our proposed method compared with the state-of-the-art approaches. In future work, we aim to enhance the computational efficiency and robustness of the BMPC planner. We plan to explore the recursive feasibility of the BMPC planner under certain technical assumptions, such as the existence of controlled invariant sets and the ability to gather information incrementally. In addition, we intend to extend this planning framework to handle intersection-crossing scenarios. Furthermore, we plan to validate the entire planning framework on a hardware platform.

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. Autom. Lett.*, vol. 4, no. 3, pp. 2997–3004, Jul. 2019.

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

[3] W. Zeng et al., "End-to-end interpretable neural motion planner," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 8652–8661.

[4] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 17201–17210.

[5] J. Thagaard, S. Hauberg, B. van der Vegt, T. Ebstrup, J. D. Hansen, and A. B. Dahl, "Can you trust predictive uncertainty under real dataset shifts in digital pathology?" in *Proc. 23rd Int. Conf. Med. Image Comput. Comput. Assist. Intervent. (MICCAI)*, Lima, Peru. Cham, Switzerland: Springer, Oct. 2020, pp. 824–833. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-59710-8_80#citeas

[6] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3145–3153.

[7] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multiplayer general-sum differential games," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 1475–1481.

[8] S. Le Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: A fast solver for constrained dynamic games," in *Proc. Robot., Sci. Syst. XVI*, Jul. 2020.

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

[10] C. Wei, Y. He, H. Tian, and Y. Lv, "Game theoretic merging behavior control for autonomous vehicle at highway on-ramp," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 21127–21136, Nov. 2022.

[11] V. G. Lopez, F. L. Lewis, M. Liu, Y. Wan, S. Nageshrao, and D. Filev, "Game-theoretic lane-changing decision making and payoff learning for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 3609–3620, Apr. 2022.

[12] P. Hang, C. Lv, Y. Xing, C. Huang, and Z. Hu, "Human-like decision making for autonomous driving: A noncooperative game theoretic approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2076–2087, Apr. 2021.

[13] S. Coskun, Q. Zhang, and R. Langari, "Receding horizon Markov game autonomous driving strategy," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 1367–1374.

[14] L. Sun, M. Cai, W. Zhan, and M. Tomizuka, "A game-theoretic strategy-aware interaction algorithm with validation on real traffic data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 11038–11044.

[15] R. Tian, L. Sun, M. Tomizuka, and D. Isele, "Anytime game-theoretic planning with active reasoning about humans' latent states for human-centered robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 4509–4515.

[16] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable Markov decision processes in robotics: A survey," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 21–40, Feb. 2023.

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

[18] J. Hardy and M. Campbell, "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles," *IEEE Trans. Robot.*, vol. 29, no. 4, pp. 913–929, Aug. 2013.

[19] H. Ahn, C. Chen, I. M. Mitchell, and M. Kamgarpour, "Safe motion planning against multimodal distributions based on a scenario approach," *IEEE Control Syst. Lett.*, vol. 6, pp. 1142–1147, 2022.

[20] A. Wang, A. Jasour, and B. C. Williams, "Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6041–6048, Jul. 2020.

[21] W. Ding, L. Zhang, J. Chen, and S. Shen, "EPSILON: An efficient planning system for automated vehicles in highly interactive environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1118–1138, Apr. 2022.

[22] 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. Autom. (ICRA)*, May 2015, pp. 1670–1677.

[23] T. Li, L. Zhang, S. Liu, and S. Shen, "MARC: Multipolicy and risk-aware contingency planning for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6587–6594, Oct. 2023.

[24] L. Zhang, S. Han, and S. Grammatico, "An efficient game-theoretic planner for automated lane merging with multi-modal behavior understanding," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, vol. 62, Sep. 2023, pp. 3085–3090.

[25] Q. Zhang, R. Langari, H. E. Tseng, D. Filev, S. Szwabowski, and S. Coskun, "A game theoretic model predictive controller with aggressiveness estimation for mandatory lane change," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 1, pp. 75–89, Mar. 2020.

[26] R. Wang, M. Schuurmans, and P. Patrinos, "Interaction-aware model predictive control for autonomous driving," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2023, pp. 1–6.

[27] H. Hu and J. F. Fisac, "Active uncertainty reduction for human–robot interaction: An implicit dual control approach," in *Algorithmic Foundations of Robotics XV* (Springer Proceedings in Advanced Robotics). Cham, Switzerland: Springer, 2023, pp. 385–401. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-21090-7_23#citeas

[28] W. Zhan et al., "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," Sep. 2019, *arXiv:1910.03088*.

[29] D. Li et al., "Social-aware decision algorithm for on-ramp merging based on level-k gaming," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2022, pp. 1753–1758.

[30] K. Ji, N. Li, M. Orsag, and K. Han, "Hierarchical and game-theoretic decision-making for connected and automated vehicles in overtaking scenarios," *Transp. Res. C, Emerg. Technol.*, vol. 150, May 2023, Art. no. 104109.

[31] D. Silver and J. Veness, "Monte–Carlo planning in large POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23. Red Hook, NY, USA: Curran Associates, 2010.

[32] Z. Sunberg and M. J. Kochenderfer, "Improving automated driving through POMDP planning with human internal states," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20073–20083, Nov. 2022.

[33] 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, nos. 2–3, pp. 558–573, Feb. 2021.

[34] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1617–1624.

[35] J. Fischer, E. Bührle, D. Kamran, and C. Stiller, "Guiding belief space planning with learned models for interactive merging," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 2542–2549.

[36] 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.

[37] B. Zhou, W. Schwarting, D. Rus, and J. Alonso-Mora, "Joint multi-policy behavior estimation and receding-horizon trajectory planning for automated urban driving," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2388–2394.

[38] D. Qiu, Y. Zhao, and C. Baker, "Latent belief space motion planning under cost, dynamics, and intent uncertainty," in *Proc. Robot., Sci. Syst. XVI*, Jul. 2020.

[39] F. Da, "Comprehensive reactive safety: No need for a trajectory if you have a strategy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 2903–2910.

[40] I. Batkovic, U. Rosolia, M. Zanon, and P. Falcone, "A robust scenario MPC approach for uncertain multi-modal obstacles," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 947–952, Jul. 2021.

[41] S. Bae, D. Isele, K. Fujimura, and S. J. Moura, "Risk-aware lane selection on highway with dynamic obstacles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jul. 2021, pp. 652–659.

[42] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, Dec. 2011. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4614-1433-9

[43] J. B. Rawlings et al., *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Madison, WI, USA: Nob Hill Publishing, 2017.

[44] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed., Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Jan. 1998. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611971132

[45] W. Wang, L. Wang, C. Zhang, C. Liu, and L. Sun, "Social interactions for autonomous driving: A review and perspectives," *Found. Trends Robot.*, vol. 10, nos. 3–4, pp. 198–376, 2022.

[46] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," Oct. 2018, *arXiv:1708.06374*.

[47] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.

[48] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state," *Auto. Robots*, vol. 42, no. 7, pp. 1405–1426, Oct. 2018.

[49] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[50] A. Zanardi, E. Mion, M. Bruschetta, S. Bolognani, A. Censi, and E. Frazzoli, "Urban driving games with lexicographic preferences and socially efficient Nash equilibria," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4978–4985, Jul. 2021.

[51] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Sep. 2015.

[52] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *Proc. IEEE Amer. Control Conf. (ACC)*, Jul. 2019, pp. 717–722.

[53] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7674–7679.

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

[55] 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, Mar. 2012.

[56] H. Caesar et al., "NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," 2022, *arXiv:2106.11810*.

[57] Y. Jiao, S. C. Calvert, H. Van Lint, and S. Van Cranenburgh, "Identifying vehicle interaction at urban intersections: A comparison of proximity resistance, time-to-collision, and post-encroachment-time," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, vol. 44, Sep. 2023, pp. 973–978.

[58] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4906–4913.

Luyao Zhang received the B.Eng. degree in electronics engineering from the University of Nottingham, Nottingham, U.K., in 2019, and the M.Sc. degree in control systems from Imperial College London, London, U.K., in 2020. He is currently the Ph.D. degree with Delft Center for Systems and Control, TU Delft, Delft, The Netherlands.

He was a Research Assistant at the Southern University of Science and Technology, Shenzhen, China, in 2021. His research interests include game theory, interaction-aware planning, and numerical optimization.

Shaohang Han received the B.S. degree in mechanical engineering from Beihang University, Beijing, China, in 2021, and the M.S. degree in robotics from Delft University of Technology, Delft, The Netherlands, in 2023. He is currently the Ph.D. degree with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, Stockholm, Sweden.

His research interests include risk-aware motion planning and control.

Sergio Grammatico (Senior Member, IEEE) received the bachelor's degree in computer engineering and the master's and Ph.D. degrees in automatic control from the University of Pisa, Pisa, Italy, in 2008, 2009, and 2013, respectively.

He is currently an Associate Professor with Delft Center for Systems and Control, TU Delft, Delft, The Netherlands. From 2012 to 2017, he held research positions at the University of California Santa Barbara, Santa Barbara, CA, USA, and ETH Zürich, Zürich, Switzerland, and TU Eindhoven, Eindhoven, The Netherlands.

Dr. Grammatico has been awarded the 2021 Roberto Tempo Best Paper Award. He was the recipient of the Best Paper Award at the 2016 ISDG International Conference on Network Games, Control and Optimization. He is currently an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL and IFAC *Automatica*.